Foundations of Reinforcement Learning

From Markov Decision Processes to Optimal Value Functions

Matthia Sabatelli

November 9, 2021

Today's Agenda

1. Course Information
2. What is Reinforcement Learning
3. Mathematical Framework
4. Value Functions

Course Information

- Coordinator: Matthia Sabatelli
- Lecturers: Matthia Sabatelli (m.sabatelli@rug.nl) and Nicole Orzan (n.orzan@rug.nl)
- Classroom: Your bedroom
- Theoretical Lectures: Monday morning from 9:00-11:00
- Computer Labs: Monday afternoon from 15:00-17:00

Course Information

- Lecture 1: Foundations of Reinforcement Learning (Matthia)
- Lecture 2: Exploration and Bandit Problems (Nicole)
- Lecture 3: Dynamic Programming (Nicole)
- Lecture 4: Model-Free Reinforcement Learning (Matthia)
- Lecture 5: Function Approximators (Matthia)
- Lecture 6: Beyond Model-Free Reinforcement Learning (Matthia)
- Lecture 7: *What does it mean to do research in RL?* (Matthia & Nicole)

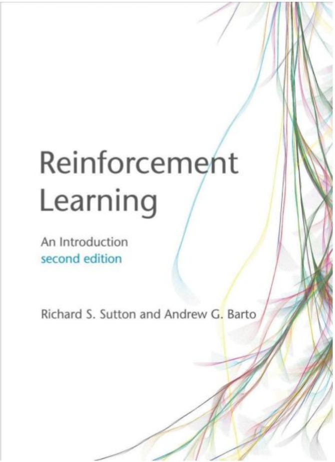
Course Information

All course material will be made available

- Nestor
- Github: `https://github.com/paintception/reinforcement-learning-practical`

Course Information

Textbook: Reinforcement Learning: An Introduction by Sutton & Barto



Course Information

Final course assessement:

- There is **no exam**
- Students should handle in three deliverables:
  1. Assignment 1: 25% of the grade (coding)
  2. Assignment 2: 25% of the grade (mathematics)
  3. Report: 50% of the grade (final project)
- Students can work alone or in groups of a maximum of **2** people

Reinforcement Learning

Machine Learning is typically divided into three branches:

1. Supervised Learning: learning from labeled data
2. Unsupervised Learning: learning from unlabeled data
3. Reinforcement Learning: learning from experience

Reinforcement Learning

A reminder of supervised learning:

- Input space: $\mathcal{X}$
- Output space: $\mathcal{Y}$
- Probability distribution $p(x, y)$

The goal

We want to build a function $f : \mathcal{X} \to \mathcal{Y}$ that minimizes the expectation of a given loss ℓ

$$\mathbb{E}_{(x,y)\sim p(x,y)}\{\ell(y, f(x))\}$$

through leaning samples $LS = \{(x_i, y_i)|i = 1, ..., N\}$ of input-ouput pairs drawn from $p(x, y)$.

Reinforcement Learning

Supervised learning is therefore:

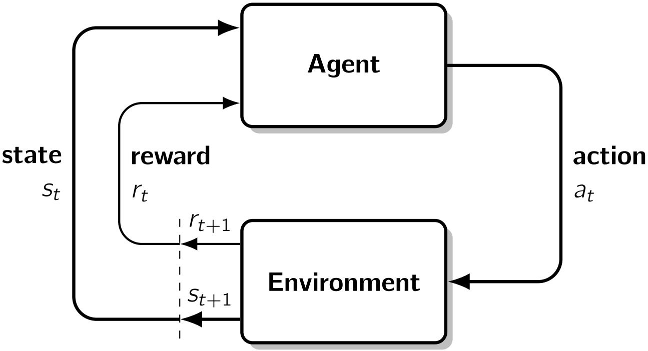
- A learning paradigm which is static
- No interaction happens between the learner and $p(x, y)$
- Assumes we have access to a knowledgable supervisor

In Reinforcement Learning however ...

Reinforcement Learning

- We would like to learn how to interact with an environment
- We do not assume any sort of supervision but only a reward signal
- The component of time plays a crucial role
- The learning process is therefore dynamic and uncertain
- Agents are goal oriented

Reinforcement Learning



Reinforcement Learning

Some examples of Reinforcement Learning:

- https://www.youtube.com/watch?v=eG1Ed8PTJ18
- https://www.youtube.com/watch?v=n2gE7n11h1Y
- https://www.youtube.com/watch?v=_qLs2K4UXXk

Reinforcement Learning

The mathematical framework of Reinforcement Learning:

- A set of possible states $\mathcal{S}$ where $s_t \in \mathcal{S}$ is the current state
- A set of possible actions $\mathcal{A}$ where $a_t \in \mathcal{A}$ is the current action
- A transition function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$
- A reward function $\Re : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ which returns r_t

Markov Decision Processes (MDPs)

These components allows us to define a Markov Decision Process $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r \rangle$.

Markov Decision Processes

What is so special about MDPs?

Markov Property

In a Markovian environment the conditional distribution of the next state of the process only depends from the current state of the process.

$$p(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, \ldots) = p(s_{t+1}|s_t, a_t).$$

Interestingly, the same property also holds for the reward that the agent will get:

$$p(r_t|s_t, a_t, \ldots, s_1, a_1) = p(r_t|s_t, a_t).$$

Markov Decision Processes

How does an agent interact with its environment?

Policy π

Through a probability distribution over $a \in \mathcal{A}(s)$ for each $s \in \mathcal{S}$:

$$\pi(a|s) = \Pr\{a_t = a | s_t = s\}, \text{ for all } s \in \mathcal{S} \text{ and } a \in \mathcal{A}.$$

which more simply can be seen as a mapping from states to actions that determines the behaviour of the agent:

$$\pi : \mathcal{S} \to \mathcal{A}$$

Episodes

Once we have policy π we can let the agent interact with the environment, which results in an episode:

$$\langle (s_t, a_t, r_t, s_{t+1}) \rangle, t = 0, \ldots, T-1$$

where T is a random variable defining the length of the episode

One $\langle s_t, a_t, r_t, s_{t+1} \rangle$ is called a trajectory τ

Goals and Returns

Why do we want an agent to interact with the environment?

- We would like it to master a certain task
- Ideally it could discover solutions that are unknown to us
- Biologically plausible form of learning

How do we formalize this mathematically?

Goal

In its easiest form we can define such goal as maximizing the sequence of r_t returned by $\Re$

$$G_t = r_t + r_{t+1} + r_{t+2}, \dots, r_T.$$

Goals and Returns

However the definition of G_t has some limitations:

- Assumes that episodes are finite
- Many real world applications are instead continuous, therefore $T = \infty$
- As a result G_t can be infinite as well

We need a slightly more complex definition of G_t based on the concept of discounting modeled by γ

Goals and Returns

Discounted Return

γ allows us to define the notion of discounted cumulative reward

$$\begin{aligned} G_t &= r_t + \gamma r_{t+1}, \gamma^2 r_{t+2} + \ldots \\ &= \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}. \end{aligned}$$

- γ determines the value of future rewards as $0 \leq \gamma \leq 1$
- Makes G_t finite as long as $\gamma < 1$ and r_t is constant

$$G_t = \sum_{k=0}^{\infty} \gamma^k = \frac{1}{1-\gamma}$$

Value Functions

One of the most concepts of Reinforcement Learning: value

- Directly connected to the notion of G_t
- We can define the value of a state s, of an action a, and even of a policy π
- Allows the introduction of value functions
  - state-value function $V(s)$
  - state-action value function $Q(s, a)$

Value Functions

The state-value function $V^{\pi}(s)$

$$\begin{aligned} V^{\pi}(s) &= \mathbb{E}\left[G_t \;\middle|\; s_t = s, \pi \right] \\ &= \mathbb{E}\left[\sum_{k}^{\infty} \gamma^k r_{t+k+1} \;\middle|\; s_t = s, \pi \right] \end{aligned}$$

- The easiest value function to learn
- Intuitively it tells us *"how good/bad"* every state s visited by policy π is
- This *"goodness"* is expressed with respect to G_t

Value Functions

The state-value function is only conditioned on the states that are being visited

The state-action value function $Q^\pi(s, a)$

$$
\begin{aligned}
Q^\pi(s,a) &= \mathbb{E}\Big[G_t \;\Big|\; s_t = s,\ a_t = a, \pi \Big] \\
&= \mathbb{E}\Big[\sum_{k}^{\infty} \gamma^k r_{t+k+1} \;\Big|\; s_t = s,\ a_t = a, \pi \Big]
\end{aligned}
$$

- Is much more informative compared to $V^\pi(s)$
- Intuitively tells us *how good/bad* taking action a in state s is
- Plays a crucial role in the development of Reinforcement Learning algorithms

Value Functions

An interesting property of these value functions is that they satisfy a recursive equality

$$\begin{aligned} V^{\pi}(s) &= \mathbb{E}\left[\sum_{k}^{\infty} \gamma^{k} r_{t+k+1} \,\middle|\, s_t = s, \pi \right] \\ &= \sum_{a} \pi(s, a) \sum_{s_{t+1}} p(s_{t+1}|s, a) \left[\Re(s_t, a, s_{t+1}) + \gamma V^{\pi}(s_{t+1}) \right] \end{aligned}$$

Value Functions

An interesting property of these value functions is that they satisfy a recursive equality

$$\begin{aligned} V^{\pi}(s) &= \mathbb{E}\left[\sum_{k}^{\infty} \gamma^k r_{t+k+1} \,\middle|\, s_t = s, \pi \right] \\ &= \sum_a \pi(s,a) \sum_{s_{t+1}} p(s_{t+1}|s,a) \left[\Re(s_t, a, s_{t+1}) + \gamma V^{\pi}(s_{t+1}) \right] \end{aligned}$$

Value Functions

An interesting property of these value functions is that they satisfy a recursive equality

$$
\begin{aligned}
V^{\pi}(s) &= \mathbb{E}\left[\sum_{k}^{\infty} \gamma^{k} r_{t+k+1} \,\middle|\, s_t = s, \pi\right] \\
&= \sum_{a} \pi(s,a) \sum_{s_{t+1}} p(s_{t+1}|s,a)\left[\Re(s_t, a, s_{t+1}) + \gamma V^{\pi}(s_{t+1})\right]
\end{aligned}
$$

which is key for the creation of Reinforcement Learning algorithms (Lectures 3 and 4)!

Value Functions

Where does this recursive property come from?

Derivation Time!

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

.

Value Functions

Where does this recursive property come from?

Derivation Time!

$$
\begin{aligned}
V^{\pi}(s) &= \mathbb{E}\Big[\sum_{k=0}^{\infty} \gamma^{k} r_{t+k} \Big| s_t = s, \pi\Big] \\
&= \mathbb{E}\Big[r_t + \gamma r_{t+1} + \gamma^{2} r_{t+2} + \ldots \Big| s_t = s, \pi\Big]
\end{aligned}
$$

.

Value Functions

Where does this recursive property come from?

Derivation Time!

$$\begin{aligned} V^{\pi}(s) &= \mathbb{E}\Big[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \Big| s_t = s, \pi\Big] \\ &= \mathbb{E}\Big[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots \Big| s_t = s, \pi\Big] \\ &= \mathbb{E}\Big[r_t + \gamma(r_{t+1} + \gamma r_{t+2} + \ldots)\Big| s_t = s, \pi\Big] \end{aligned}$$

.

Value Functions

Where does this recursive property come from?

Derivation Time!

$$
\begin{aligned}
V^{\pi}(s) &= \mathbb{E}\Big[\sum_{k=0}^{\infty}\gamma^{k} r_{t+k}\Big| s_t = s, \pi\Big] \\
&= \mathbb{E}\Big[r_t + \gamma r_{t+1} + \gamma^{2} r_{t+2} + \ldots \Big| s_t = s, \pi\Big] \\
&= \mathbb{E}\Big[r_t + \gamma(r_{t+1} + \gamma r_{t+2} + \ldots)\Big| s_t = s, \pi\Big] \\
&= \mathbb{E}\Big[r_t + \gamma V^{\pi}(s_{t+1})\Big| s_t = s, \pi\Big]
\end{aligned}
$$

.

Value Functions

Where does this recursive property come from?

Derivation Time!

$$\begin{aligned}
V^{\pi}(s) &= \mathbb{E}\Big[\sum_{k=0}^{\infty} \gamma^{k} r_{t+k} \Big| s_t = s, \pi\Big] \\
&= \mathbb{E}\Big[r_t + \gamma r_{t+1} + \gamma^{2} r_{t+2} + \dots \Big| s_t = s, \pi\Big] \\
&= \mathbb{E}\Big[r_t + \gamma(r_{t+1} + \gamma r_{t+2} + \dots) \Big| s_t = s, \pi\Big] \\
&= \mathbb{E}\Big[r_t + \gamma V^{\pi}(s_{t+1}) \Big| s_t = s, \pi\Big] \\
&= \sum_{a} \pi(s, a) \sum_{s_{t+1}} p(s_{t+1}|s, a)\Big[\Re(s_t, a, s_{t+1}) + \gamma V^{\pi}(s_{t+1})\Big].
\end{aligned}$$

Optimal Value Functions

Solving a Reinforcement Learning problem intuitively means finding a policy π that achieves a lot of reward:

- What makes a certain policy π better than another policy π'?
- How can we rank different policies?

We can answer these questions thanks to the $V(s)$ and the $Q(s, a)$ functions:

Optimal Policy π^*

There is always one policy that is better or equal than all other policies

$$\pi \geq \pi' \text{ iff } V^{\pi}(s) \geq V^{\pi'}(s) \text{ for all } s \in \mathcal{S}$$

The best possible policy is the optimal policy π^*

Optimal Value Functions

- How do we find the optimal policy π^*?
- By maximizing the state-value and state-action value functions results in the optimal value functions

$$V^*(s) = \max_{\pi} V^{\pi}(s) \qquad\qquad Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a)$$

which, if expressed in a recursive form, result in the Bellman optimality equations (for the coming lectures!)

Final Slide!

We aim to teach an agent how to interact with its environment:

Lecture Takeaway

1. The environment is modelled as a Markov Decision Process $\mathcal{M}\langle\mathcal{S}, \mathcal{A}, \mathcal{P}, r\rangle$
2. The interaction is governed by the agent's policy π
3. The goal of the agent is measured wrt discounted cumulative reward G_t
4. G_t allows us to quantify how good a certain state s is, and how good a certain action a in a certain state is: $V^{\pi}(s)$ and $Q^{\pi}(s, a)$