Reinforcement Learning: Tutorial 2

Introduction & MDPs

Week 1
University of Amsterdam

Milena Kapralova
September 2024

Check-in

How is it going?

Outline

1. Admin
2. Tutorial 2
   - Introduction
   - Exploration
   - Markov Decision Processes

Admin

- Have you started looking for a HW buddy?
- Any questions?

Tutorial 2 Overview

1. Introduction
2. Exploration
3. Markov decision processes

Tutorial 2 Overview

1. Introduction
   - Questions 1.1.1 - 1.1.4
2. Exploration
3. Markov decision processes

Q 1.1 Introduction

1. Explain what is meant by the 'curse of dimensionality'.

Q 1.1 Introduction

1. Explain what is meant by the 'curse of dimensionality'.

From the book: 'the computational requirements grow exponentially with the number of state variables'.

Q 1.1 Introduction

2. Suppose you are trying to design a predator agent that can learn to catch a randomly moving prey on a 5×5 **toroidal** grid. You have been given the (x, y)-coordinates of the predator and the (x, y)-coordinates of the prey to use as the state.

Q 1.1 Introduction

2. Suppose you are trying to design a predator agent that can learn to catch a randomly moving prey on a 5×5 **toroidal** grid. You have been given the (x, y)-coordinates of the predator and the (x, y)-coordinates of the prey to use as the state.
   a. How many possible states are there in this naive approach?

Q 1.1 Introduction

2. Suppose you are trying to design a predator agent that can learn to catch a randomly moving prey on a 5×5 **toroidal** grid. You have been given the (x, y)-coordinates of the predator and the (x, y)-coordinates of the prey to use as the state.
   a. How many possible states are there in this naive approach?

5^4.

Q 1.1 Introduction

2. Suppose you are trying to design a predator agent that can learn to catch a randomly moving prey on a 5×5 **toroidal** grid. You have been given the (x, y)-coordinates of the predator and the (x, y)-coordinates of the prey to use as the state.
   - b There is a way of reducing the state space considerably a priori. Write down how you would adapt the given state representation to reduce the size of the state space. Note that you only care about a representation that you can use to solve the problem.

Q 1.1 Introduction

2. Suppose you are trying to design a predator agent that can learn to catch a randomly moving prey on a 5×5 **toroidal** grid. You have been given the (x, y)-coordinates of the predator and the (x, y)-coordinates of the prey to use as the state.
   b. There is a way of reducing the state space considerably a priori. Write down how you would adapt the given state representation to reduce the size of the state space. Note that you only care about a representation that you can use to solve the problem.

Instead of using both (x, y)-coordinates, use the point-wise distance between the predator and the prey. So if predator is at (x, y) and prey is at (x', y') then the state is (for instance) $(x - x', y - y')$. Note that in this representation something like $(-2, -2)$ is the same state as $(3, 3)$, due to the toroidal symmetry (the maximum distance between predator and prey in any direction is 4). This means we can choose to represent the distances as numbers between 0 and 4 (inclusive).

Q 1.1 Introduction

2. Suppose you are trying to design a predator agent that can learn to catch a randomly moving prey on a 5×5 **toroidal** grid. You have been given the (x, y)-coordinates of the predator and the (x, y)-coordinates of the prey to use as the state.
   c How many possible states are there now?

Q 1.1 Introduction

2. Suppose you are trying to design a predator agent that can learn to catch a randomly moving prey on a 5×5 **toroidal** grid. You have been given the (x, y)-coordinates of the predator and the (x, y)-coordinates of the prey to use as the state.
   - c How many possible states are there now?

In the new representation, there are 5^2 states.

Q 1.1 Introduction

2. Suppose you are trying to design a predator agent that can learn to catch a randomly moving prey on a 5×5 **toroidal** grid. You have been given the (x, y)-coordinates of the predator and the (x, y)-coordinates of the prey to use as the state.
   d What is the advantage of doing this?

Q 1.1 Introduction

2. Suppose you are trying to design a predator agent that can learn to catch a randomly moving prey on a 5×5 **toroidal** grid. You have been given the (x, y)-coordinates of the predator and the (x, y)-coordinates of the prey to use as the state.
   d What is the advantage of doing this?

Alleviating the curse of dimensionality by reducing the number of state variables.

Q 1.1 Introduction

2. Suppose you are trying to design a predator agent that can learn to catch a randomly moving prey on a 5×5 **toroidal** grid. You have been given the (x, y)-coordinates of the predator and the (x, y)-coordinates of the prey to use as the state.
   e. Consider the Tic-Tac-Toe example in Chapter 1.5 of the book. Here, too, we can exploit certain properties of the problem to reduce the size of the state space. Give an example of a property you can exploit.

Q 1.1 Introduction

2. Suppose you are trying to design a predator agent that can learn to catch a randomly moving prey on a 5×5 **toroidal** grid. You have been given the (x, y)-coordinates of the predator and the (x, y)-coordinates of the prey to use as the state.
   e. Consider the Tic-Tac-Toe example in Chapter 1.5 of the book. Here, too, we can exploit certain properties of the problem to reduce the size of the state space. Give an example of a property you can exploit.

By exploiting rotation invariance in the value function.

Q 1.1 Introduction

3. Suppose you want to implement a Reinforcement Learning agent that learns to play Tic-Tac-Toe, as outlined in Chapter 1 of the book.

Q 1.1 Introduction

3. Suppose you want to implement a Reinforcement Learning agent that learns to play Tic-Tac-Toe, as outlined in Chapter 1 of the book.
   a. Which agent do you think would learn a better policy in the end: a greedy agent that always chooses the action it currently believes is best, or a non-greedy agent that sometimes tries new actions? Why?

Q 1.1 Introduction

3. Suppose you want to implement a Reinforcement Learning agent that learns to play Tic-Tac-Toe, as outlined in Chapter 1 of the book.
   a. Which agent do you think would learn a better policy in the end: a greedy agent that always chooses the action it currently believes is best, or a non-greedy agent that sometimes tries new actions? Why?

We expect the curious agent to perform better: it will be able to discover strategies that the greedy agent may miss.

Q 1.1 Introduction

4. Assume we start with an exploration rate of ϵ, meaning that whenever the agent chooses an action, it has a probability of ϵ to pick an action at random, and a probability of $1 - \epsilon$ to pick the greedy action. If we assume the environment has been sufficiently explored, we may want to reduce the amount of exploration after some time.

Q 1.1 Introduction

4. Assume we start with an exploration rate of ϵ, meaning that whenever the agent chooses an action, it has a probability of ϵ to pick an action at random, and a probability of $1 - \epsilon$ to pick the greedy action. If we assume the environment has been sufficiently explored, we may want to reduce the amount of exploration after some time.
    a. Write down how you would do this.

Q 1.1 Introduction

4. Assume we start with an exploration rate of ϵ, meaning that whenever the agent chooses an action, it has a probability of ϵ to pick an action at random, and a probability of $1 - \epsilon$ to pick the greedy action. If we assume the environment has been sufficiently explored, we may want to reduce the amount of exploration after some time.
   a. Write down how you would do this.

Straightforward: annealing ϵ over time.

Q 1.1 Introduction

4. Assume we start with an exploration rate of ϵ, meaning that whenever the agent chooses an action, it has a probability of ϵ to pick an action at random, and a probability of $1 - \epsilon$ to pick the greedy action. If we assume the environment has been sufficiently explored, we may want to reduce the amount of exploration after some time.
   - b Does your method work if the opponent changes strategies? Why/why not? If not, provide suggestions on a heuristic that can adapt to changes in the opponent's strategy.

Q 1.1 Introduction

4. Assume we start with an exploration rate of ϵ, meaning that whenever the agent chooses an action, it has a probability of ϵ to pick an action at random, and a probability of $1 - \epsilon$ to pick the greedy action. If we assume the environment has been sufficiently explored, we may want to reduce the amount of exploration after some time.
   b Does your method work if the opponent changes strategies? Why/why not? If not, provide suggestions on a heuristic that can adapt to changes in the opponent's strategy.

Time-based: No. Heuristics that would work: temporal difference (TD) error, curiosity.


WELL DONE!

Tutorial 2 Overview

1. Introduction
2. Exploration
   - Questions 1.2.1 - 1.2.6
3. Markov decision processes

Q 1.2 Exploration

1. In ϵ-greedy action-selection for the case of n actions, what is the probability of selecting the greedy action?

Q 1.2 Exploration

1. In ϵ-greedy action-selection for the case of n actions, what is the probability of selecting the greedy action?

We have probability $1 - \epsilon$ of selecting a greedy action, and ϵ of selecting a random action uniformly. Thus, each individual action has base probability of selection of $\frac{\epsilon}{n}$. The probability of selecting a greedy action is thus $1 - \epsilon + \frac{\epsilon}{n}$.

Q 1.2 Exploration

2. Consider a 3-armed bandit problem with actions $1, 2, 3$. If we use ϵ-greedy action-selection, initialization at 0, and **sample-average** action-value estimates, which of the following sequence of actions are certain to be the result of exploration?
$A_0 = 1, R_1 = -1, A_1 = 2, R_2 = 1, A_2 = 2, R_3 = -2,$
$A_3 = 2, R_4 = 2, A_4 = 3, R_5 = 1.$

Q 1.2 Exploration

2. Consider a 3-armed bandit problem with actions $1, 2, 3$. If we use ϵ-greedy action-selection, initialization at 0, and **sample-average** action-value estimates, which of the following sequence of actions are certain to be the result of exploration?
   $A_0 = 1, R_1 = -1, A_1 = 2, R_2 = 1, A_2 = 2, R_3 = -2,$
   $A_3 = 2, R_4 = 2, A_4 = 3, R_5 = 1.$

Start of Q-values: [0, 0, 0].
After $A_0 = 1$: [-1, 0, 0].
After $A_1 = 2$: [-1, 1, 0].
After $A_2 = 2$: [-1, -0.5, 0].
After $A_3 = 2$: [-1, 0.333, 0].
After $A_4 = 3$: [-1, 0.333, 1].
Actions that were non-greedy: A_3, A_4.

Q 1.2 Exploration

2. You are trying to find the optimal policy for a two-armed bandit. You try two approaches: in the pessimistic approach, you initialize all action-values at -5, and in the optimistic approach you initialize all action-values at $+5$. One arm gives a reward of $+1$, one arm gives a reward of -1. Using a greedy policy to choose actions, compute the resulting Q-values for both actions after three interactions with the environment. In case of a tie between two Q-values, break the tie at random. *Note*: the initialization is *not* a sample.

Q 1.2 Exploration

2. You are trying to find the optimal policy for a two-armed bandit. You try two approaches: in the pessimistic approach, you initialize all action-values at -5, and in the optimistic approach you initialize all action-values at +5. One arm gives a reward of +1, one arm gives a reward of -1. Using a greedy policy to choose actions, compute the resulting Q-values for both actions after three interactions with the environment. In case of a tie between two Q-values, break the tie at random. *Note*: the initialization is *not* a sample.

a. Start of Q-values: [5, 5]. After $A_0 = 1$: [1, 5]. After $A_1 = 2$: [1, -1] (these two can be flipped with the same end result). After $A_2 = 1$: [1, -1]. Total return: 1+-1+1=1

Q 1.2 Exploration

2. You are trying to find the optimal policy for a two-armed bandit. You try two approaches: in the pessimistic approach, you initialize all action-values at -5, and in the optimistic approach you initialize all action-values at +5. One arm gives a reward of +1, one arm gives a reward of -1. Using a greedy policy to choose actions, compute the resulting Q-values for both actions after three interactions with the environment. In case of a tie between two Q-values, break the tie at random. *Note*: the initialization is *not* a sample.

a Start of Q-values: [5, 5]. After $A_0 = 1$: [1, 5]. After $A_1 = 2$: [1, -1] (these two can be flipped with the same end result). After $A_2 = 1$: [1, -1]. Total return: 1+-1+1=1

b Start of Q-values: [-5, -5]. After $A_0 = 1$: [1, -5]. After $A_1 = 1$: [1, -5]. After $A_2 = 1$: [1, -5]. Total return: 1+1+1=3. If the tie is broken differently: $A_0 = 2$: [-5, -1]. $A_1 = 2$: [-5, -1]. $A_2 = 2$: [-5, -1]. Total return: -1+-1+-1=-3

Q 1.2 Exploration

2. Which initialization leads to a higher (undiscounted) return? What if you had broken the tie differently?

Q 1.2 Exploration

2. Which initialization leads to a higher (undiscounted) return? What if you had broken the tie differently?

If the tie is broken one way: pessimistic has higher return (3 vs $+1$).
If it's broken the other way, optimistic has higher return (-3 vs $+1$).

Q 1.2 Exploration

2. Which initialization leads to a better estimation of the Q-values?

Q 1.2 Exploration

2. Which initialization leads to a better estimation of the Q-values?

   The optimistic initialization.

Q 1.2 Exploration

2. Explain why one of the two initialization methods is better for exploration.

Q 1.2 Exploration

2. Explain why one of the two initialization methods is better for exploration.

Looking for an answer along the lines that the optimistic one is better due to unexplored options having a very high value and thus higher chance of actually being selected.

Tutorial 2 Overview

1. Introduction
2. Exploration
3. Markov decision processes
   - Questions 1.3.1 - 1.3.2

Q 1.3 Markov Decision Processes

1. a For the first four examples outlined in Section 1.2 of the book, describe the state space, action space and reward signal.

Q 1.3 Markov Decision Processes

1. a For the first four examples outlined in Section 1.2 of the book, describe the state space, action space and reward signal.

   1 State space: all possible configurations of chess board. Action space: all allowed moves of one piece at a time. Reward signal: win/lose/draw

   2 State space: all possible configurations of parameters. Action space: change of parameters. Reward signal: marginal cost

   3 State space: limb configurations. Action space: change angles of limbs. Reward signal: penalizes falling, reward acceleration

   4 State space: battery level, location of charger. Action space: enter/not enter. Reward signal: penalize running out of battery, reward collecting trash

Q 1.3 Markov Decision Processes

1. b Come up with an example of your own that you might model as an MDP. State the action space, state space and reward signal.

Q 1.3 Markov Decision Processes

1. b Come up with an example of your own that you might model as an MDP. State the action space, state space and reward signal.

?

Q 1.3 Markov Decision Processes

1. c Come up with an example for a problem that you might have trouble solving with an MDP. Why doesn't this fit the framework?

Q 1.3 Markov Decision Processes

1. c Come up with an example for a problem that you might have trouble solving with an MDP. Why doesn't this fit the framework?

For example, non-Markov states (biological processes).

Q 1.3 Markov Decision Processes

1. d In mazes, the agent's position is often seen as the state. However, the agent's position alone in not always a sufficient description. Come up with an example where the state consists of the agent's location and one or more other variables.

Q 1.3 Markov Decision Processes

1. d In mazes, the agent's position is often seen as the state. However, the agent's position alone in not always a sufficient description. Come up with an example where the state consists of the agent's location and one or more other variables.

For example, in Pacman the state consists of the agent's location, as well as the location of the ghosts and food pellets. In a driving tasks, the state consists of the agent's location as well as the state of other vehicles and context variables such as time of day, season, traffic light status. In a maze, the possession of a key might be another state variable next to the agent's location.

Q 1.3 Markov Decision Processes

1. e Consider the example in exercise 3.3 of the book. Why might you choose to view the actions as handling the accelerator, brake and steering wheel? What is the disadvantage of doing this?

Q 1.3 Markov Decision Processes

1. e Consider the example in exercise 3.3 of the book. Why might you choose to view the actions as handling the accelerator, brake and steering wheel? What is the disadvantage of doing this?

This low-level representation allows you to learn *how* to drive. The disadvantage is that this approach makes it very hard to navigate anywhere, since our representation is too fine-grained.

Q 1.3 Markov Decision Processes

1. f Why might you choose to view the actions as choosing where to drive? What is the disadvantage of doing this?

Q 1.3 Markov Decision Processes

1. f Why might you choose to view the actions as choosing where to drive? What is the disadvantage of doing this?

This high-level representation is much better suited to learning how to navigate somewhere, and to reach high-level goals such as *go to the supermarket*. The disadvantage is that we have to assume the agent already knows how to drive a car.

Q 1.3 Markov Decision Processes

1. g Can you think of some way to combine both approaches?

Q 1.3 Markov Decision Processes

1. g Can you think of some way to combine both approaches?

   Two-level policies, hierarchical RL.

Q 1.3 Markov Decision Processes

2. a. Eq. 3.8 in the book gives the discounted return for the continuing case. Write down the formula for the discounted return in the episodic case.

Q 1.3 Markov Decision Processes

2. a Eq. 3.8 in the book gives the discounted return for the continuing case. Write down the formula for the discounted return in the episodic case.

$$G_t = \sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1}$$

Q 1.3 Markov Decision Processes

2 b Show that $\sum_{k=0}^{\infty} \gamma^k = \frac{1}{1-\gamma}$ if $0 \leq \gamma < 1$. (Hint: if you're stuck, have a look at the Wikipedia page on *geometric series*)

Q 1.3 Markov Decision Processes

2 b Show that $\sum_{k=0}^{\infty} \gamma^k = \frac{1}{1-\gamma}$ if $0 \leq \gamma < 1$. (Hint: if you're stuck, have a look at the Wikipedia page on *geometric series*)

$$\sum_{k=0}^{\infty} \gamma^k = \lim_{n \to \infty} (1 + \gamma + \gamma^2 + \cdots + \gamma^n)$$

$$\begin{aligned}(1-\gamma) \sum_{k=0}^{\infty} \gamma^k &= \lim_{n \to \infty} (1-\gamma)(1 + \gamma + \gamma^2 + \cdots + \gamma^n) \\ &= \lim_{n \to \infty} \left((1-\gamma) + (\gamma - \gamma^2) + (\gamma^2 - \gamma^3) + \cdots + (\gamma^n - \gamma^{n+1})\right) \\ &= \lim_{n \to \infty} \left(1 - \gamma^{n+1}\right) \Rightarrow \sum_{k=0}^{\infty} \gamma^k = \lim_{n \to \infty} \frac{1 - \gamma^{n+1}}{1 - \gamma}\end{aligned}$$

If $\gamma < 1$, $\lim_{n \to \infty} \gamma^n \to 0$, so $\sum_{k=0}^{\infty} \gamma^k = \frac{1}{1-\gamma}$.

Q 1.3 Markov Decision Processes

2. c Consider exercise 3.7 in the book. Why is there no improvement in the agent?

Q 1.3 Markov Decision Processes

2. c Consider exercise 3.7 in the book. Why is there no improvement in the agent?

The return is always $0 + 0 + \cdots + 1 = 1$, regardless of how many time steps the agent takes.

Q 1.3 Markov Decision Processes

2. d How would adding a discount factor of $\gamma < 1$ help solve this problem?

Q 1.3 Markov Decision Processes

2. d How would adding a discount factor of $\gamma < 1$ help solve this problem?

By adding γ, the return is now γ^K, with K the number of time steps until the robot escapes. Since $\gamma < 1$, the return is bigger if the robot escapes faster.

Q 1.3 Markov Decision Processes

2. e How might changing the reward function help solve this problem?

Q 1.3 Markov Decision Processes

2. e How might changing the reward function help solve this problem?

By changing the reward to give a small penalty of e.g. $R_t = -0.01$ on every time step, the return is now $G_t = -0.01(T - t)$, which means the return is bigger if the robot escapes faster.

That's it!

See you next Monday