Introduction to reinforcement learning

July 9, 2019

Introduction

- This morning we will study an important AI paradigm : **Reinforcement learning (RL)**

Applications of RL

- RL has many applications and is quite a hot topic.

Applications of RL

- RL has many applications and is quite a hot topic.
- Especially **Deep Reinforcement Learning** has received a lot of attention recently.

Applications of Deep Reinforcement Learning I

- Atari games

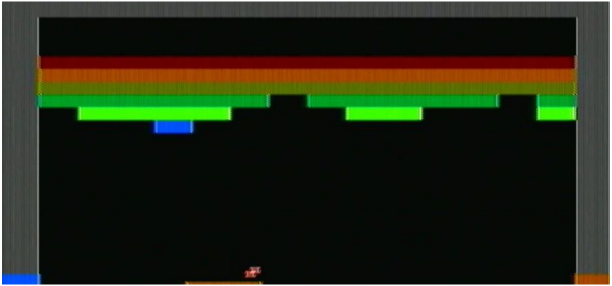


Figure: Atari game

Applications of Deep Reinforcement Learning II

- AlphaGo

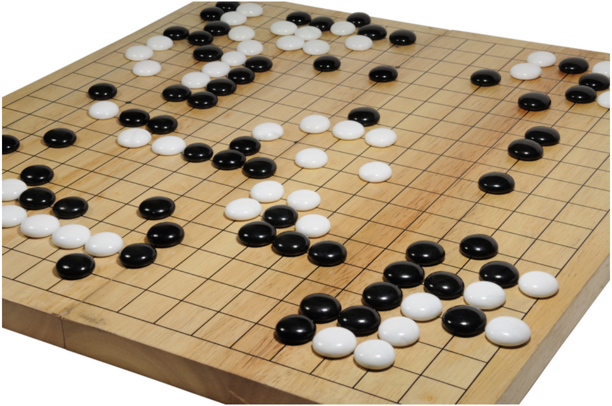


Figure: Go game, beaten by AlphaGo in 2017 [Silver et al., 2016]

Applications of Reinforcement Learning III

- Reinforcement Learning is also begin used in the community of **Computationnal neuroscience.**

Ressources

- `https://github.com/nlehir/summerschool` contains our slides and exercices.
- When doing exercises, we will be using **python 3**

References

- [Andrew and Sutton, 1998]

Overview

Supervised learning

- Let us first recall what supervised learning is.

Supervised learning

- Let us first recall what supervised learning is.
- MNIST demo

Supervised learning and Correction

- In **supervised learning**, the supervisor indicates the **expected answer** the agent should answer.
- With our mnist digit classificatino example, the actin of the agent is the **prediction of the class**.

Supervised learning and Correction

- In **supervised learning**, the supervisor indicates the **expected answer** the agent should answer.
- The feedback does not depend on the action performed by the agent (for instance the prediction from the agent)

Supervised learning and Correction

- In **supervised learning**, the supervisor indicates the **expected answer** the agent should answer.
- The feedback does not depend on the action performed by the agent (for instance the prediction from the agent)
- We say that the agent receives an **instructive feedback**

Supervised learning Correction

- In **supervised learning**, the supervisor indicates the **expected answer** the agent should answer.
- The agent must then **correct its model** based on this answer.

Cost sensitive learning

- In **Cost sensitive learning**, the situation is different.
- The agent receives an **evaluative feedback**. The feedack depends on the action performed by the agent.

Cost sensitive learning

- In **Cost sensitive learning**, the situation is different.
- The agent receives an **evaluative feedback**. The feedack depends on the action performed by the agent.
- **Examples :**
  - AI playing a game and receiving "victory" or "defeat" as a feedback.
  - Child playing with an animal.

Reinforcement learning

- **Reinforcement learning** is a particular case of cost-sensitive learning.

Reinforcement learning

- **Reinforcement learning** is a particular case of cost-sensitive learning.
- In reinforcement learning, the feedback is a **real number**

Reinforcement learning

- **Reinforcement learning** is a particular case of cost-sensitive learning.
- In reinforcement learning, the feedback is a **real number**
- **Example :** amount of coins won after a poker turn.

Reinforcement learning

- First, the agent does not know if a reward is good or bad per se
- A reward of -10 good be good or bad depending on the other rewards that are possible to obtain.

Reinforcement learning

- First, the agent does not know if a reward is good or bad per se.
- A reward of -10 good be good or bad depending on the other rewards that are possible to obtain.
- The objective of the agent will be to optimize the **agregation of rewards**

Reinforcement learning

- The idea is that the agent lives in a world E, and can be in several states s. The agent performs **actions** a and receives rewards r.

Reinforcement learning

- The idea is that the agent lives in a world E, and can be in several states s. The agent performs **actions** a and receives rewards r.
- **Examples :**
  - world $= \mathbb{R}^2$
  - state $=$ position
  - actions $=$ moving somewhere
  - reward $=$ amount of food found

Formalization

- There are many aspects of the problem that we need to formalize. Depending on the situation, the formalization could have some differences.

Formalization

- There are many aspects of the problem that we need to formalize. Depending on the situation, the formalization could have some differences.
- We will consider **discrete spaces** :
  - the time will be discrete
  - the number of possible states will be **finite**
  - the number of possible actions will be **finite**

Formalization

- There are many aspects of the problem that we need to formalize. Depending on the situation, the formalization could have some differences.
- We will consider **discrete spaces** :
  - the time will be discrete
  - the number of possible states will be **finite**
  - the number of possible actions will be **finite**

Question

- We will consider **discrete spaces** :
  - the time will be discrete
  - the number of possible states will be **finite**
  - the number of possible actions will be **finite**
- Are these hypothesis valid in the case of AlphaGo ?

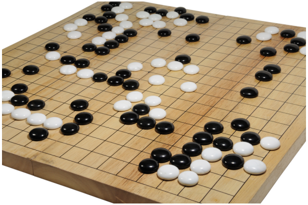


Figure: Go game, beaten by AlphaGo in 2017 [Silver et al., 2016]

Question

- We will consider **discrete spaces** :
  - the time will be discrete
  - the number of possible states will be **finite**
  - the number of possible actions will be **finite**
- Are these hypothesis valid in the case of AlphaGo ?

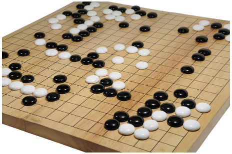


Figure: Go game, beaten by AlphaGo in 2017 [Silver et al., 2016]

- Yes ! So you can do crazy stuff in discrete spaces.

Question

- Are these hypothesis valid in the case of AlphaGo ?

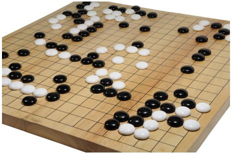


Figure: Go game, beaten by AlphaGo in 2017 [Silver et al., 2016]

- Yes ! So you can do crazy stuff in discrete spaces.
- However, please note that this is not always the case. Sometimes the possible actions are continuous, the available psitions are continuous, etc.

Let us continue with the formalization

- we will write :
  - s_t : state at time t
  - a_t : action performed at time t
  - r_t : reward received at time t
- how is the action chosen ?

Let us continue with the formalization

- we will write :
  - s_t : state at time t
  - a_t : action performed at time t
  - r_t : reward received at time t
- how is the action chosen ?

Let us continue with the formalization

- we will write :
  - s_t : state at time t
  - a_t : action performed at time t
  - r_t : reward received at time t
- the actions are chosen according to a **policy** π

Policies

- The policy π depends on the current state.

Policies

- The policy π depends on the current state.
- It can be **deterministic :** the action chosen is chosen with probability 1

Policies

- The policy π depends on the current state.
- It can be **deterministic :** the action chosen is chosen with probability 1
- Or **stochastic :** the action performed in a given state is drawn from a **distribution**

Two levels of randomness

- The policy can be deterministic or stochastic.
- But the result of an action could also be stochastic ! This is called a **stochastic transition function**.

Two levels of randomness

- The policy can be deterministic or stochastic.
- But the result of an action could also be stochastic ! This is called a **stochastic transition function**.

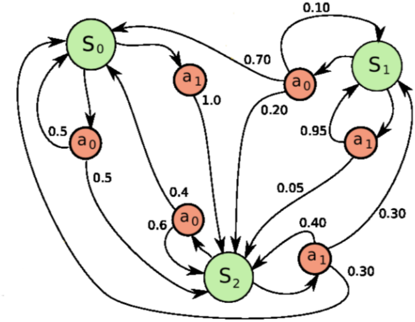


Figure: A stochastic policy with a stochastic transition function.

Exercise

- What is the probability of staying in state S_0 when performing an action from S_0 ? and from S_1 and S_2 ?

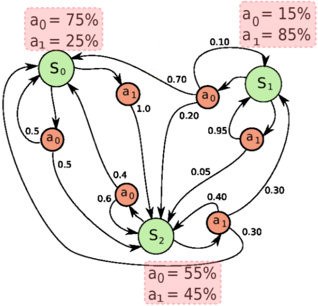


Figure: A stochastic policy with a stochastic transition function.

Agregation of rewards

- Remember that our agent want to optimize the **agregation of the rewards.**
- However, what exactly does the agent maximise ?

Agregation of rewards

- Remember that our agent want to optimize the **agregation of the rewards.**
- However, what exactly does the agent maximise ?
- There are several ways to agregate the rewards.

Agregation of rewards

- If the horizon is finite, we can take the sum

$$V^{\pi}(s_0) = r_0 + \cdots + r_N \tag{1}$$

Agregation of rewards

- If the horizon is finite, we can take the sum

$$V^{\pi}(s_0) = r_0 + \cdots + r_N \tag{2}$$

- We could also average a window. For instance a window of size 3 :

$$V^{\pi}(s_0) = \frac{r_0 + r_1 + r_2}{3} \tag{3}$$

Agregation of rewards : discount factor

- the **discount factor** $\gamma \in [0, 1]$ allows you to weight the rewards r_k

$$V^{\pi}(s_0) = \sum_{t=t_0}^{+\infty} \gamma^{t-t_0} r_t \tag{4}$$

More considerations

- The Markov hypothesis
- Exploitation exploration compromise

Art

"RL is a science, but dealing with the exploration-exploitation compromise is an art" (Sutton)

Dynamic programming

- Today we will study a simple case of Reinforcement learning
- In that case, the result of our actions is deterministic.

World

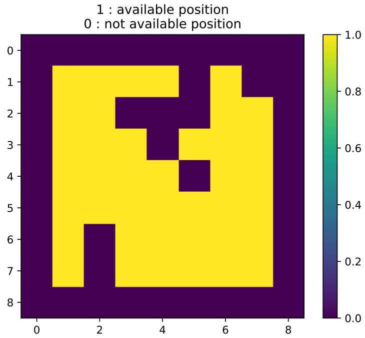


Figure: 2 dimensional world.

Reward

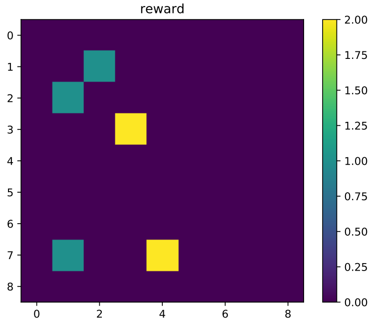


Figure: Reward function.

2D world

- Our agent can move in the 4 directions, one step at a time.
- We will progressively build an agent that learns to evaluate the states and then learns how to go to the best state.

Value function

- For each state (=position in the 2D world), we want to compute the **value function**.
-
$$V(s_0) = r_0 + \gamma r_1 + \gamma^2 r_2 \dots \tag{5}$$

Exercise

- For each state (=position in the 2D world), we want to compute the **value function**.
- $$V(s_0) = r_0 + \gamma r_1 + \gamma^2 r_2 \dots \tag{6}$$
- Can you express $V(s_0)$ as a function of $V(s_1)$?

Bellman equation

- This equation is the Bellman equation.

Value Iteration

- First, the initial Value function for all the states is 0.

Value Iteration

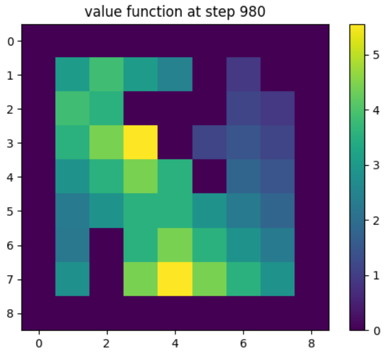
- First, the initial Value function for all the states is 0.
- Then we propagate the information about the rewards between the states, in order to **update the value function**
- We can find an optimal policy in the following way :

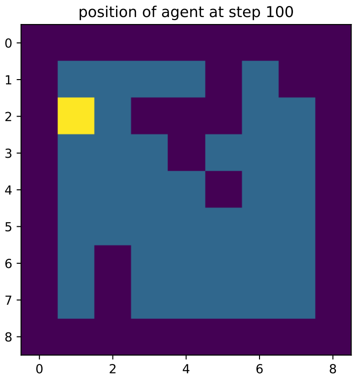
$$\forall s \in V(s_t) \leftarrow \max_{a_t} \left(r_{s_t} + \gamma V(s_{t+1}) \right) \tag{7}$$

(s_{t+1} depends on a_t).

Value iteration

- After learning, we will obtain a value function




position of agent at step 100
0
1
2
3
4
5
6
7
8
0
2
4
6
8

Exercise 2A

- in the **reinforcement learning folder**
- Please use the file **create_world.py** in order to generate your own environment.
- You can use the one that is already there if you prefer.
- We store the data about the world in **.npy** files.

Exercise 2B

- In **value_iteration.py**, modify the function **move_agent** so that the agent is randomly moved.

Exercise 2C

- In **value_iteration.py**, modify the function **update_value_function** in order to modify the value function according to the Bellman equation.

Exercise 2D

- Finally, make the alrogithm run in order to **converge to the optimal value function.**

Exercise 3

- Please use the file **value_iteration_policy** in order to design an optimal policy for our agent.

Optimal policy

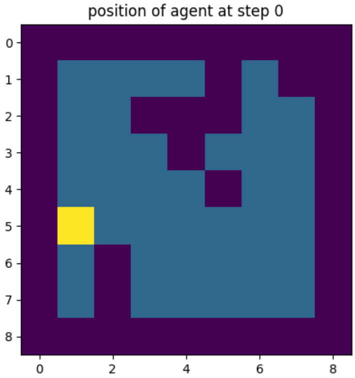


Figure: After learning, the agent can go to the reward.

Optimal policy

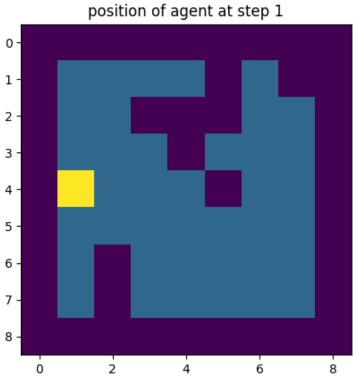


Figure: After learning, the agent can go to the reward.

Optimal policy

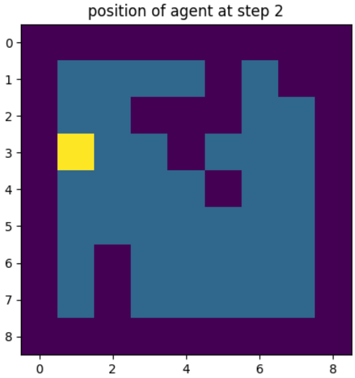


Figure: After learning, the agent can go to the reward.

Optimal policy

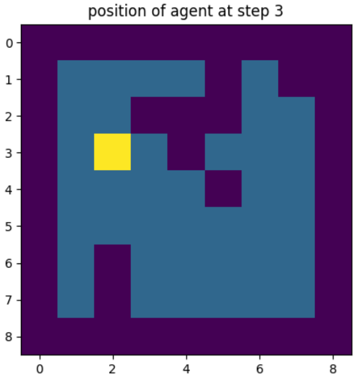


Figure: After learning, the agent can go to the reward.

Optimal policy

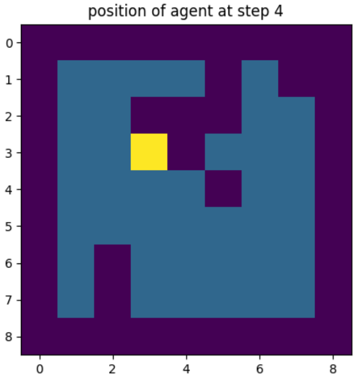


Figure: After learning, the agent can go to the reward.

Optimal policy

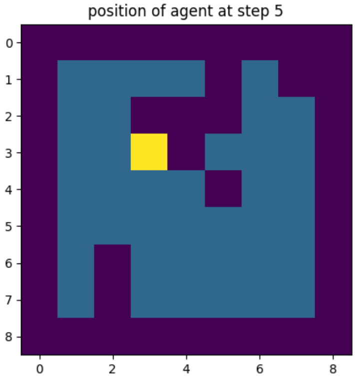


Figure: After learning, the agent can go to the reward.

References I

Andrew, A. M. and Sutton, R. S. (1998).
Reinforcement Learning: An Introduction.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016).
Mastering the Game of {Go} with Deep Neural Networks and Tree Search.
Nature, 529(7587):484–489.