Introduction to reinforcement learning

July 10, 2019

Value Iteration

- We will start by finishing our work on **value iteration** that we started yesterday.

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

Value Iteration

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

Value Iteration

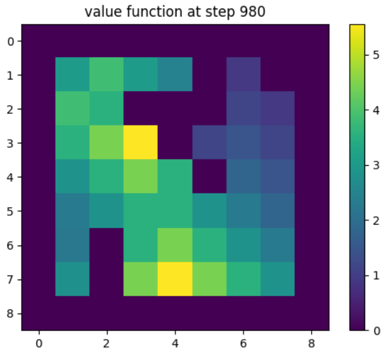
- First, the initial Value function for all the states is 0.
- Then we propagate the information about the rewards between the states, in order to **update the value function**
- We can find an optimal policy in the following way :

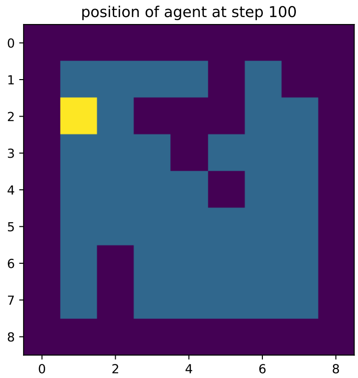
$$\forall s \in V(s_t) \leftarrow \max_{a_t} \left(r_{s_t} + \gamma V(s_{t+1}) \right) \tag{1}$$

(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

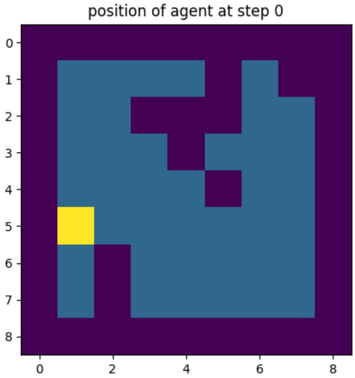


Figure: After learning, the agent can go to the reward.

Optimal policy

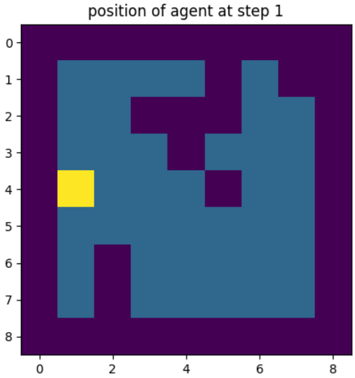


Figure: After learning, the agent can go to the reward.

Optimal policy

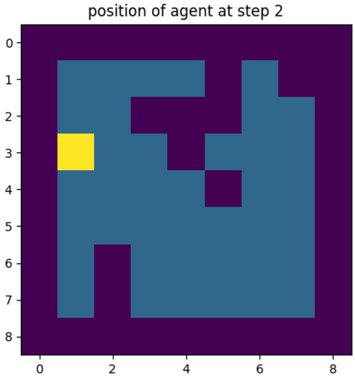


Figure: After learning, the agent can go to the reward.

Optimal policy

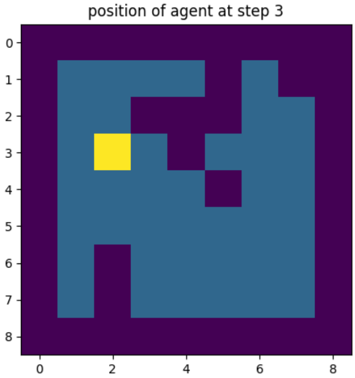


Figure: After learning, the agent can go to the reward.

Optimal policy

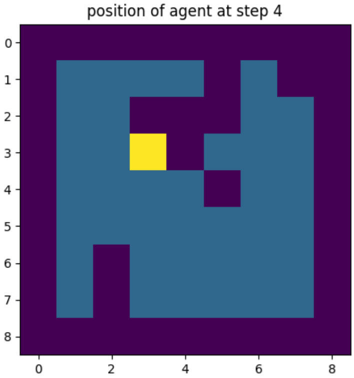


Figure: After learning, the agent can go to the reward.

Optimal policy

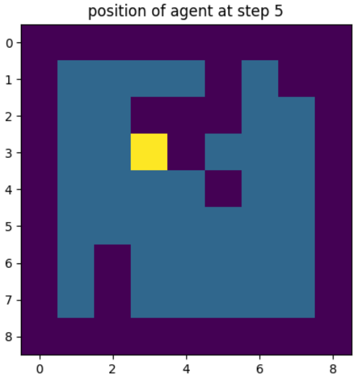


Figure: After learning, the agent can go to the reward.

Remark

- Before going closer to RL, let us do another example of **dynamic programming.**

Policy Iteration

- **Policy iteration** is another method that is slightly different.

Policy Iteration

- **Policy iteration** is another method that is slightly different.
- It consists in two steps :
  - **Policy evaluation**

Policy Iteration

- **Policy iteration** is another method that is slightly different.
- It consists in two steps :
  - **Policy evaluation**
  - **Policy improvement**

Exercise 4A

- Pease use the file **policy_iteration.py** in order to perform the algorithm.

Exercise 4B

- Pease use the file **policy_iteration.py** in order to perform the algorithm.
- Add randomness to the actions of the agent to **guarantee exploration** .

Multiple paradigms

- Reinforcement learning has many variants.
- In the ones we studied, a model of the effect of our actions were known.
- This is not always de case.

Temporal difference learning

- In temporal difference learning, the agent does not know a **model** of its world.

Temporal difference learning

- In temporal difference learning, the agent does not know a **model** of its world.
- But it can still learn the value function with the **TD updates**

Temporal difference learning

- In temporal difference learning, the agent does not know a **model** of its world.
- But it can still learn the value function with the **TD updates**

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] \quad (2)$$

Monte Carlo methods

- Monte Carlo methods can be used in Reinforcement Learning.

Monte Carlo methods

- Monte Carlo methods can be used in Reinforcement Learning.
- For instance in episodic games, we can do statistics on the values of the states.

Actor critic methods

- Sometimes you can use **two** policies

Actor critic methods

- Sometimes you can use **two** policies
  - the **behavior policy** provides actions and guarantees exploration
  - the **target polivy** is the optimal policy learned in parallel by the agent, that would be used in exploitation mode.

Bias variance compromise

- Very generally speaking, the complexity of your model influences the biais and the variance.

Bias variance compromise

- Very generally speaking, the complexity of your model influences the biais and the variance.
  - more complex : less bias, more variance
  - less complex : more bias, less variance

Tabular case and continous case

- We studied **finite** (and thus discrete situations).

Tabular case and continous case

- We studied **finite** (and thus discrete situations).

Tabular case and continous case

- We studied **finite** (and thus discrete situations).
- However, RL can also be applied to continuous state / discrete action spaces (DQN)

Tabular case and continous case

- We studied **finite** (and thus discrete situations).
- However, RL can also be applied to continuous state / discrete action spaces (DQN) [**?**]
- And even to continous state / continous action spaces (DDPG) [Bengio, 2009] .

References I

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

Bengio, Y. (2009).
CONTINUOUS CONTROL WITH DEEP REINFORCEMENT LEARNING.
Foundations and Trends® in Machine Learning.