Course 5: Reinforcement Learning

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Summary

Last session

1. Combinatorial game theory
2. Definition of a game
3. Proof of determined games

Today's session

1. Reinforcement Learning
2. Value and Policy Functions
3. Q-Learning

Note: reinforcement learning and combinatorial game theory share a common mathematical framework. But to ease access to online resources, we will adopt a new vocabulary.

Outline of the course

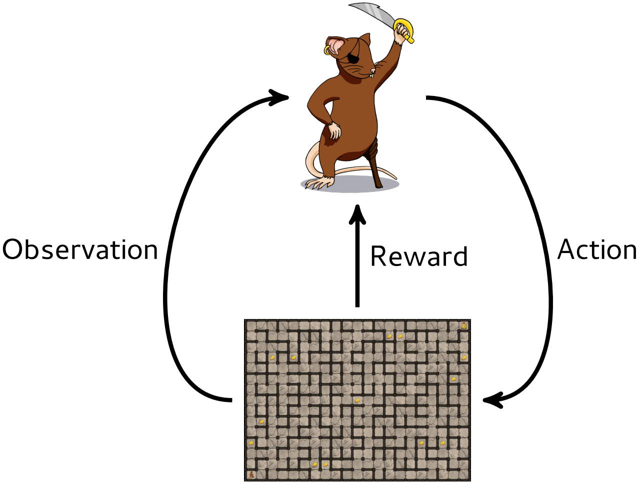
1. Definitions of Reinforcement Learning (RL)
    - Fundamentals
    - Example: PyRat
    - Policy and values
2. Q-learning
    - Q-learning definitions
    - Example
    - Approximate Q-learning
    - Exploration/Exploitation

Plan

1. Definitions of Reinforcement Learning (RL)
   - Fundamentals
   - Example: PyRat
   - Policy and values
2. Q-learning
   - Q-learning definitions
   - Example
   - Approximate Q-learning
   - Exploration/Exploitation

Agent and environment

Our objective is to train an agent to maximize its reward through actions that affect an environment.



Fundamentals

Reward hypothesis

- *All goals can be described by the maximization of excepted cumulated reward over time.*

Specificities of reinforcement learning

- No supervision, only a reward signal,
- Delayed feedback, the reward can come (much) later,
- Importance of the temporal dimension,
- Agent's actions affect the subsequent data it receives.

Fundamentals

Reward hypothesis

- *All goals can be described by the maximization of excepted cumulated reward over time.*

Specificities of reinforcement learning

- No supervision, only a reward signal,
- Delayed feedback, the reward can come (much) later,
- Importance of the temporal dimension,
- Agent's actions affect the subsequent data it receives.

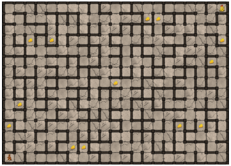
Agent and environment

Agent state representation $s_t^\alpha \in S^\alpha$

Observation $o_t \in O$

$r_t \in \mathbb{R}$ Reward

Action $a_t \in A$



Environment state representation $s_t^e \in S^e$

Definitions

- The agent α...
  1. analyzes previous actions, states, rewards and observations,
  2. computes action a_t,
  3. obtains reward r_t,
  4. obtains an observation o_{t+1},
  5. deduce a new state s_{t+1}^{α}.
- The environment...
  1. receives action a_t,
  2. produces reward r_t,
  3. deduce a new state s_{t+1}^{e},
  4. produces o_{t+1}.
- Observability:
  - Perfect: $s_t^{\alpha} = s_t^{e} = o_t$
  - Imperfect: no access to full environment state:
    - The agent indirectly observes the environment through o_t,
    - s_t^{α} is estimated by the agent and may differ from s_t^{e}.

Definitions

- The agent α...
  1. analyzes previous actions, states, rewards and observations,
  2. computes action a_t,
  3. obtains reward r_t,
  4. obtains an observation o_{t+1},
  5. deduce a new state s^{α}_{t+1}.
- The environment...
  1. receives action a_t,
  2. produces reward r_t,
  3. deduce a new state s^{e}_{t+1},
  4. produces o_{t+1}.

- Observability:
  - Perfect: $s^{\alpha}_t = s^{e}_t = o_t$
  - Imperfect: no access to full environment state:
    - The agent indirectly observes the environment through o_t,
    - s^{α}_t is estimated by the agent and may differ from s^{e}_t.

Definitions

- The agent α...
  1. analyzes previous actions, states, rewards and observations,
  2. computes action a_t,
  3. obtains reward r_t,
  4. obtains an observation o_{t+1},
  5. deduce a new state s^{α}_{t+1}.
- The environment...
  1. receives action a_t,
  2. produces reward r_t,
  3. deduce a new state s^{e}_{t+1},
  4. produces o_{t+1}.

- Observability:
  - Perfect: $s^{\alpha}_t = s^{e}_t = o_t$
  - Imperfect: no access to full environment state:
    - The agent indirectly observes the environment through o_t,
    - s^{α}_t is estimated by the agent and may differ from s^{e}_t.

Definitions

- The agent α...
  1. analyzes previous actions, states, rewards and observations,
  2. computes action a_t,
  3. obtains reward r_t,
  4. obtains an observation o_{t+1},
  5. deduce a new state s^α_{t+1}.
- The environment...
  1. receives action a_t,
  2. produces reward r_t,
  3. deduce a new state s^e_{t+1},
  4. produces o_{t+1}.

- Observability:
  - Perfect: $s^\alpha_t = s^e_t = o_t$
  - Imperfect: no access to full environment state:
    - The agent indirectly observes the environment through o_t,
    - s^α_t is estimated by the agent and may differ from s^e_t.

Example: PyRat

Definitions

- The agent is either the Rat or the Python,
- The opponent becomes part of the environment,
  - Note that the game can be with perfect observability if the opponent strategy is known,
- Seen this way, the game becomes sequential.

RL-based PyRat versus supervised approach

- Reward signal: number of picked up pieces of cheese,
- Delayed feedback: several moves required to reach a reward,
- Character's moves affect subsequent data it receives,
- Importance of the temporal dimension.

Example: PyRat

Definitions

- The agent is either the Rat or the Python,
- The opponent becomes part of the environment,
  - Note that the game can be with perfect observability if the opponent strategy is known,
- Seen this way, the game becomes sequential.

RL-based PyRat versus supervised approach

- Reward signal: number of picked up pieces of cheese,
- Delayed feedback: several moves required to reach a reward,
- Character's moves affect subsequent data it receives,
- Importance of the temporal dimension.

Example: PyRat

Observability examples

- Perfect: $s_t^{rat} = s_t^o = o_t$
  - a_t: Last move of the rat,
  - o_t: The entire maze with all cheese locations and python position,
  - r_t: Binary variable which is 1 if the rat just got a piece of cheese.
- Imperfect:
  - a_t: Last move of the rat,
  - o_t: Neighboring cells of the rat,
  - r_t: Binary variable which is 1 if the rat just got a piece of cheese.

To represent the strategy of the rat, we use a policy function.

Example: PyRat

Observability examples

- Perfect: $s_t^{rat} = s_t^o = o_t$
  - a_t: Last move of the rat,
  - o_t: The entire maze with all cheese locations and python position,
  - r_t: Binary variable which is 1 if the rat just got a piece of cheese.
- Imperfect:
  - a_t: Last move of the rat,
  - o_t: Neighboring cells of the rat,
  - r_t: Binary variable which is 1 if the rat just got a piece of cheese.

To represent the strategy of the rat, we use a policy function.

Example: PyRat

Observability examples

- Perfect: $s_t^{rat} = s_t^o = o_t$
  - a_t: Last move of the rat,
  - o_t: The entire maze with all cheese locations and python position,
  - r_t: Binary variable which is 1 if the rat just got a piece of cheese.
- Imperfect:
  - a_t: Last move of the rat,
  - o_t: Neighboring cells of the rat,
  - r_t: Binary variable which is 1 if the rat just got a piece of cheese.

To represent the strategy of the rat, we use a policy function.

Policy Function

Definition

The policy function of an agent α is:

$$\pi : \begin{cases} S^\alpha & \rightarrow & A \\ s_t^\alpha & \mapsto & a_t^\alpha \end{cases}$$

- π can be deterministic or stochastic.

Playout

The playout $(s_t^{\alpha,\pi})_{t\in\mathbb{N}}$ associated with a policy π and initial state s_0, is defined by considering agent α takes his/her actions using π.

Policy Function

Definition

The policy function of an agent α is:

$$\pi : \begin{cases} S^{\alpha} & \rightarrow & A \\ s_t^{\alpha} & \mapsto & a_t^{\alpha} \end{cases}$$

- π can be deterministic or stochastic.

Playout

The playout $(s_t^{\alpha,\pi})_{t\in\mathbb{N}}$ associated with a policy π and initial state s_0, is defined by considering agent α takes his/her actions using π.

Value Function

Definition

Fix $\gamma \in [0, 1[$, the value function v^π is defined as:

$$v^\pi : \begin{cases} S^\alpha & \to \quad \mathbb{R} \\ s_{t_0}^{\alpha,\pi} & \mapsto \quad \sum_{t=t_0}^{+\infty} \gamma^{t-t_0} r_t \end{cases}$$

- The value of a policy function is thus an expectation of cumulative future rewards, weakened by the geometrical coefficient γ to avoid divergence,
- The best possible policy $\pi^*(s)$ is defined as:

$$\forall s \in S^\alpha, \forall \pi, V^{\pi^*}(s) \geq V^\pi(s).$$

Value Function

Definition

Fix $\gamma \in [0, 1[$, the value function v^π is defined as:

$$v^\pi : \begin{cases} S^\alpha & \rightarrow \quad \mathbb{R} \\ s_{t_0}^{\alpha,\pi} & \mapsto \quad \sum_{t=t_0}^{+\infty} \gamma^{t-t_0} r_t \end{cases}$$

- The value of a policy function is thus an expectation of cumulative future rewards, weakened by the geometrical coefficient γ to avoid divergence,
- The best possible policy $\pi^*(s)$ is defined as:

$$\forall s \in S^\alpha, \forall \pi, V^{\pi^*}(s) \geq V^\pi(s).$$

Plan

Q-learning

Definition

In Q-learning, we aim to find V^{π^*} as a solution to the recursive system of equations (Bellman equation):

$$\forall s \in S^\alpha, \forall a \in A, Q(s,a) = r_{s,a} + \gamma \max_{a'} Q(s(a), a'),$$

where $r_{s,a}$ is the reward agent α performs action a in state s and $s(a)$ is the state observed by agent α after performing action a.

Pros and cons

Pros:

- Can be learned even if the agent is not following any specific π,
- Self training is possible,

Cons:

- Scalability issues when S^α is large.

Q-learning

Definition

In Q-learning, we aim to find V^{π^*} as a solution to the recursive system of equations (Bellman equation):

$$\forall s \in S^\alpha, \forall a \in A\,,\, Q(s,a) = r_{s,a} + \gamma \max_{a'} Q(s(a), a'),$$

where $r_{s,a}$ is the reward agent α performs action a in state s and $s(a)$ is the state observed by agent α after performing action a.

Pros and cons

Pros:

- Can be learned even if the agent is not following any specific π,
- Self training is possible,

Cons:

- Scalability issues when S^α is large.

Q-learning

Definition

In Q-learning, we aim to find V^{π^*} as a solution to the recursive system of equations (Bellman equation):

$$\forall s \in S^\alpha, \forall a \in A, Q(s,a) = r_{s,a} + \gamma \max_{a'} Q(s(a), a'),$$

where $r_{s,a}$ is the reward agent α performs action a in state s and $s(a)$ is the state observed by agent α after performing action a.

Pros and cons

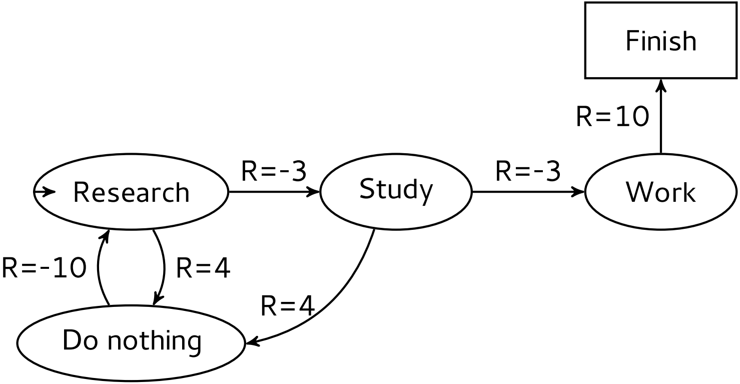
Pros:

- Can be learned even if the agent is not following any specific π,
- Self training is possible,

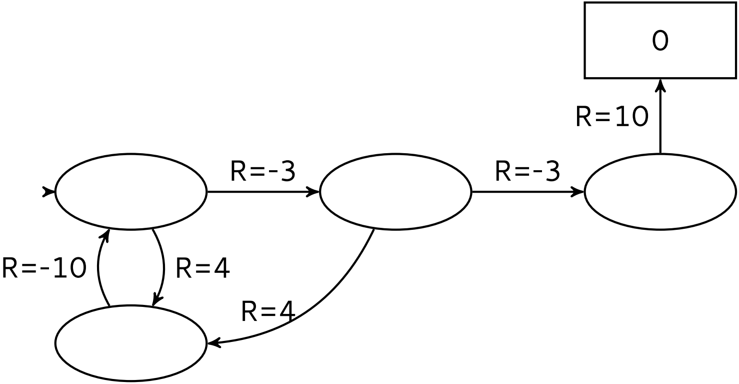
Cons:

- Scalability issues when S^α is large.

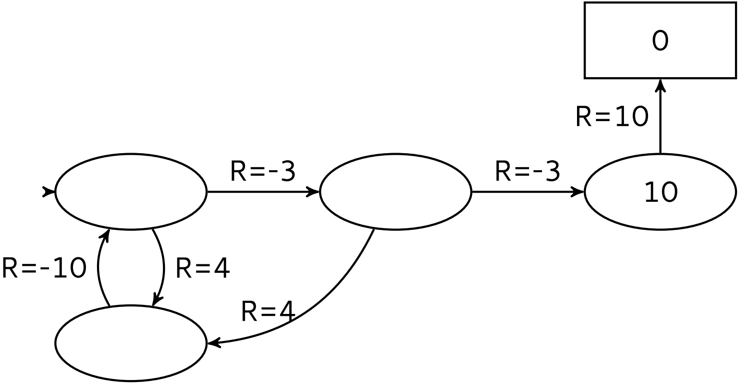
Q-learning example



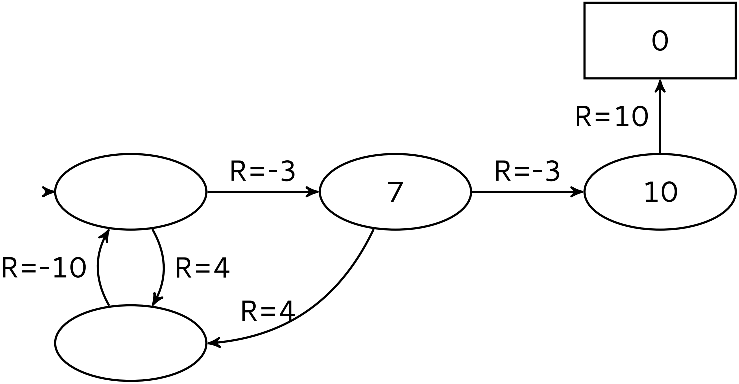
Q-learning example



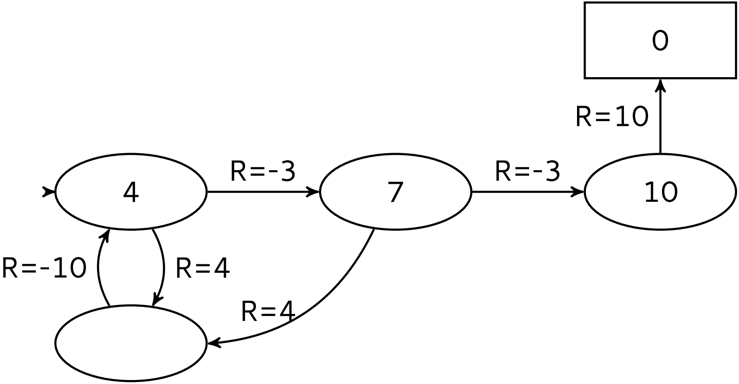
Q-learning example



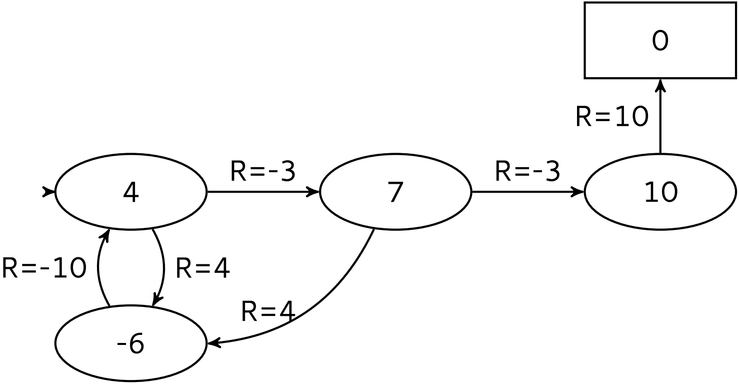
Q-learning example



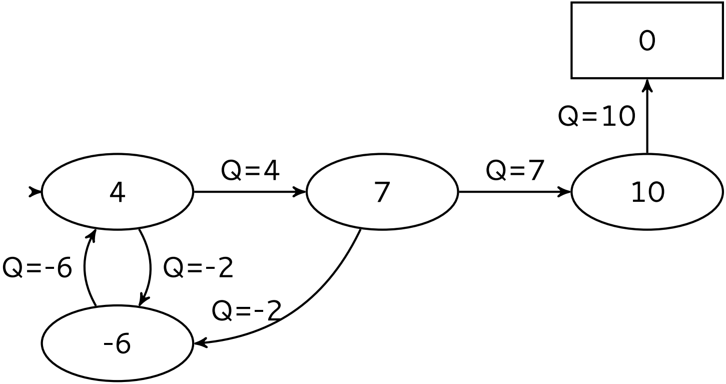
Q-learning example



Q-learning example



Q-learning example



Approximated Q-learning

Definition

- Train a model to approximate Q,
  - Input is a state s and output is made of values of $Q(s, \cdot)$,
  - Representation learning can be used to compress S^α.

Problems

- Almost always needs a simulator for the game,
- Game duration can be bottleneck for training,
- Catastrophic forgetting and adversary specialization,
  - These effects can be alliviated by training using experience replay.

Experience replay

- Instead of using only the last decision to train, sample at random from the m previous decisions,
- Decisions taken before should remain considered now.

Approximated Q-learning

Definition

- Train a model to approximate Q,
  - Input is a state s and output is made of values of $Q(s, \cdot)$,
  - Representation learning can be used to compress S^α.

Problems

- Almost always needs a simulator for the game,
- Game duration can be bottleneck for training,
- Catastrophic forgetting and adversary specialization,
  - These effects can be alliviated by training using experience replay.

Experience replay

- Instead of using only the last decision to train, sample at random from the m previous decisions,
- Decisions taken before should remain considered now.

Approximated Q-learning

Definition

- Train a model to approximate Q,
  - Input is a state s and output is made of values of $Q(s, \cdot)$,
  - Representation learning can be used to compress S^{α}.

Problems

- Almost always needs a simulator for the game,
- Game duration can be bottleneck for training,
- Catastrophic forgetting and adversary specialization,
  - These effects can be alliviated by training using experience replay.

Experience replay

- Instead of using only the last decision to train, sample at random from the m previous decisions,
- Decisions taken before should remain considered now.

Dilemma

- Repeat with existing strategy (Exploitation)...
- ... or try a new strategy (Exploration)?

Example

- Always eating in restaurants that you know is exploitation,
- While that is a good heuristic, you have no way of knowing if you have the maximum reward possible,
- So exploring new restaurants from time to time may be needed to find the maximum reward.

Exploration/Exploitation

Dilemma

- Repeat with existing strategy (Exploitation)...
- ... or try a new strategy (Exploration)?

Example

- Always eating in restaurants that you know is exploitation,
- While that is a good heuristic, you have no way of knowing if you have the maximum reward possible,
- So exploring new restaurants from time to time may be needed to find the maximum reward.

Lab Session 5

TP4 - PyRat with reinforcement learning

- Approximate Q-learning algorithm using experience replay and linear regression to beat the greedy algorithm,
- Approximation method (linear regression) and experience replay routine are given,
- Assemble all the primitives to perform Reinforcement Learning.

Challenge

You can continue working in the challenge after finishing TP4. You can now integrate reinforcement learning in your solution.