Assignment 8

Subject

Topics of this session :

1. Decision making under uncertainty.
2. Markov Decision Processes.
3. Backwards Induction.

This assignement is graded and must be submitted (individually) on Moodle before next week's class.

For each exercise, detail your reflexion steps :
- We are mostly interested in your actual thinking process.
- Even if you are unable to solve an exercise, write out what were you reflexion steps.
- For each attempted exercise, a written feedback will be provided (if time alllows it).

For coding :
- Noto (Online Jupyter NoteBook).
- Any other python coding environment you prefer using.

Exercise 1

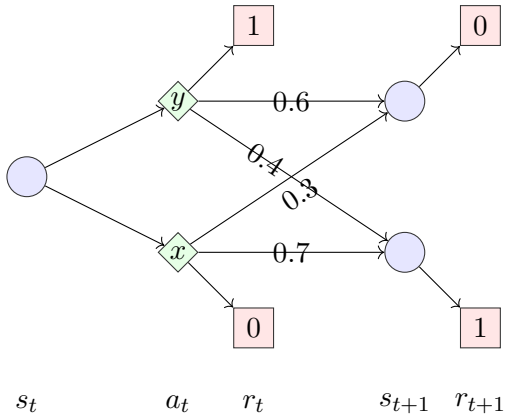
For each of the following problems, tell if they can be modeled as a Markov Process, Markov Decision Process, Non-Markovian Process or Non-Markovian Decision Process :
- Car Driving.
- Playing Poker.
- Stock Values in the Stock Market.

Explain your answers.

Exercise 2

Consider the following one-step MDP. You are in state s_t and can choose between two actions x and y. Each action leads probabilistically to a next state and gives a deterministic immediate reward.

Questions ? victor.villin@unine.ch



Assume that this MDP ends after s_{t+1} (i.e., horizon is 2). Let the value of each terminal state be its immediate reward.

Reminders. For any Markov policy $\pi(a_t|s_t)$, the definition of the state value function is :

$$V_t^\pi(s) = \mathbb{E}^\pi[U_t|s_t = s],$$

where $U_t = \sum_{k=t}^{T} r_t$ is the utility from timestep t, or total remaining reward until time T.

Recall also the definition of the state-action value function :

$$Q_t^\pi(s,a) = \mathbb{E}^\pi[U_t \mid s_t = s, a_t = a].$$

If a policy π^* is **optimal**, then :

$$\forall(s,a), \quad \pi^*(s,a) = \arg\max_a Q^*(s,a).$$

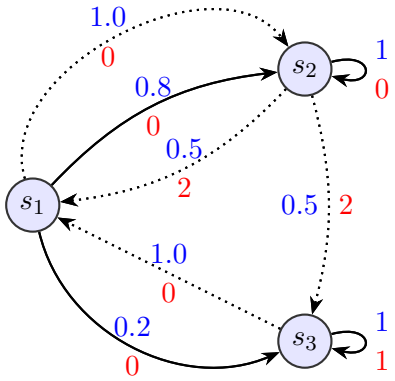
Note : All of these definitions will be provided at the exam.

---

1. Compute the expected utility from s_t if you take action x.
2. Compute the expected utility from s_t if you take action y.
3. What is the value of the policy π at state s_t, $V_t^\pi(s_t)$, given : $\pi(x|s_t) = 0.2$ and $\pi(y|s_t) = 0.8$.
4. Which action is optimal at s_t ?

Exercise 3

Consider the following MDP with three states : s_1, s_2, and s_3. At each state, you can choose between two actions : a_1 (solid edges) and a_2 (dotted edges). Transitions probabilities and rewards associated to actions are given in each edges, in blue and red respectively. The process ends after $T = 3$ steps.



Reminder. The **Backwards Induction algorithm** calculates the optimal value function through the following recursion, initialised as $Q_T^*(s,a) = r(s,a)$ (or setting $V_{T+1}^*(s,a) = 0$).

$$Q_t^*(s,a) = r(s,a) + \sum_{s'} P(s'|s,a) V_{t+1}^*(s')$$
$$V_t^*(s) = \max_a Q_t^*(s,a).$$

---

1. Define the transition probabilities for every state s, action a and next state s' : $P(s'|s,a)$.
2. Define the reward function for every state s and action a : $r(s,a)$.
3. Find the optimal value function for $T = 3$ steps of backwards induction, where, at the last step, $Q_T^*(s,a) = r(s,a)$, filling it in the table below.

t	$V_t^*(s_1)$	$V_t^*(s_2)$	$V_t^*(s_3)$	$Q_t^*(s_1,a_1)$	$Q_t^*(s_1,a_2)$	$Q_t^*(s_2,a_1)$	$Q_t^*(s_2,a_2)$	$Q_t^*(s_3,a_1)$	$Q_t^*(s_3,a_2)$
3									
2									
1									

4. Deduce the optimal action for each state and timestep.
5. What is the expected utility of the optimal policy, if we let it start at s_1 (at $t = 1$). Can you interpret that value ?

Project – Part 5

This whole section must be done in groups of 2-3 people.

This week, we will model our problem into an MDP.

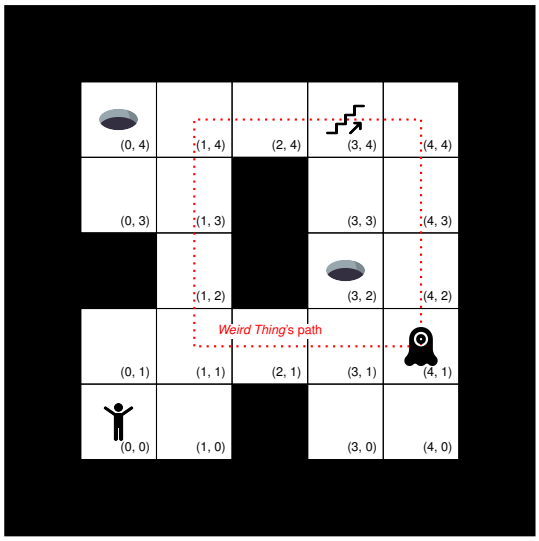


FIGURE 1 – A simple dungeon room with holes. The agent starts at position (0, 0). The *Weird Thing* starts at (4, 1).

1 Guided Project

Dungeon Gridworld To not make things too complicated. We consider the following setting :

- **Stochastic transitions**.
- **Full observability**.
- **Single-agent** (If there is other agents, their policy is fixed, and known).

Aligning with above, we will assume for our problem that :

- The goal is to reach the stairs **as fast as possible**.
- Falling into a hole causes the game to end immediately.
- The agent **has full observability**.
- If you walk over a *hole* (that is, when you move to a hole tile), there is a 50% chance that you fall into the hole. Otherwise, you manage to avoid it.
- There is a *Weird Thing* walking around the dungeon. If the agent ends on the same tile as the *Weird Thing*, it gets instantly dissolved into goo... We know that the *Weird Thing* never deviates from its path (see Figure 1), with the following policy :
  - It moves clockwise 75% of the time.
  - It moves counter-clockwise 20% of the time.
  - It does not move 5% of the time.
- The agent gets penalised by -1 for every step it takes in the maze. It also gets penalized by -100 if it falls into a hole or gets dissolved into goo. If the agent gets to the stairs and that the *weird thing* is also located at the stairs, the agent loses.

Task :

1. Using `dungeon_mdp.ipynb` as a template, complete the code.
2. Propose an implementation of Backwards Induction, leaving comments highlighting your understanding of the algorithm, and use it to solve the problem, with a maximum horizon of $T = 25$.
3. Interpret the policy you end up computing.

2 Personal Project

Task :

1. Model your problem as an MDP. It should have stochastic transitions, full observability. If there is other agents in your game (such as ennemies you cannot control), their policy should be fixed and known by our agent.
2. Implement your problem as an MDP, using the guided project code as a template, and solve it using Backwards Induction.
3. Interpet the policy you get by running Backwards Induction.