Artificial Intelligence
5 – Markov Decision Processes (MDP)

Arthur Bit-Monnot

INSA 4IR

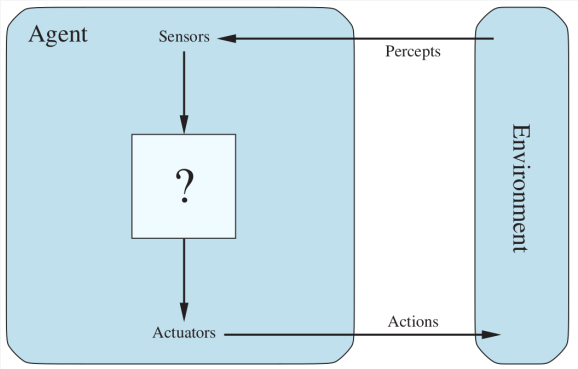
Section 1

Previously

Agent (course #2)

An **agent** is anything that can be viewed as:

- **perceiving** its environment through **sensors**, and
- **acting** upon that environment through **actuators**



Rational agents select their actions in order to maximize a **performance measure**

Environments (course #2)

- **environment** is characterized along several dimensions
  - **fully observable** vs **partially observable** (partial characterization of current state)
  - **deterministic** vs **stochastic** (non-predictable state evolution)
  - **episodic** vs **sequential** (several non-independent actions required)
  - **static** vs **dynamic** (autonomous state evolution with time)
  - **single-agent** vs **multi-agent** (other agents qith non-independent objectives)

Goal-based agent (course #3 / lab 1)

Environment: fully observable / deterministic / "sequential" / static / single-agent

- Goals describe desirable situations (*happy* states)
- The agent selects actions that lead to the goal
- requires following a **plan**: sequence of actions leading from the current state to a **goal state**
- Producing a (high quality?) plan is a computationnally hard decision problem
  - best-first search: dijkstra, A*, greedy best-first search (course #3, lab1)
  - many other methods: greedy algorithms, mathematical programming, local-search (metaheuristics course)
- In such environment (evolving predictably only through the agent's action) a plan is guaranteed to work

Utility & Decision Theory (course #4)

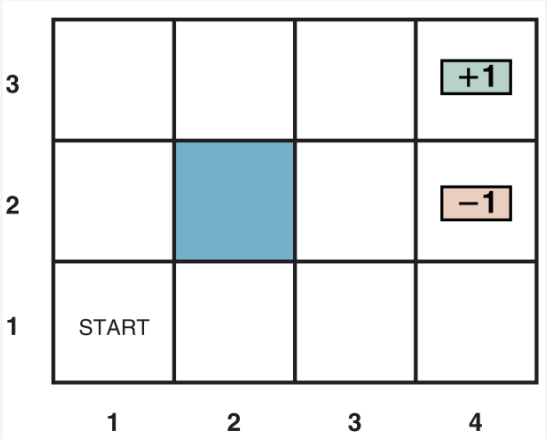
Environment: fully observable / **stochastic** / **episodic** / static / single-agent

- **probability theory** describes what an agent believes
  - probability distribution on the current state
  - probability distribution on the state after taking one action
- **utility theory** describes what an agent wants
- **decision theory** combines the two to describe what an agent should do
- an agent that shows **consistent preferences** possesses a **utility function**
  - numeric measure of "happiness" for each state
- a rational agent can act by selecting the action that **maximizes the expected utility**

Section 2

Markov Decision Processes

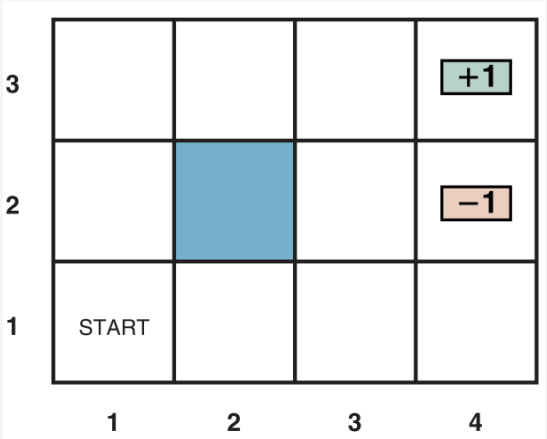
Sequential Decision processes



Grid environment:

- each cell is a state
- agent can move up, down, left, right
- each action has a small cost of 0.04
- two terminal states, with respective rewards of $+1$ and -1

Sequential Decision processes



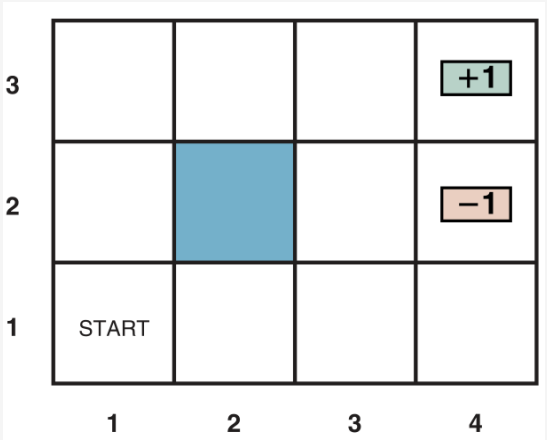
Grid environment:

- each cell is a state
- agent can move up, down, left, right
- each action has a small cost of 0.04
- two terminal states, with respective rewards of $+1$ and -1

Optimal plan (Shortest action sequence to goal):

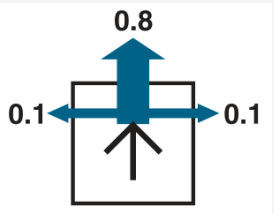
- [Up, Up, Right, Right, Right]

Sequential Decision processes, with stochastic actions

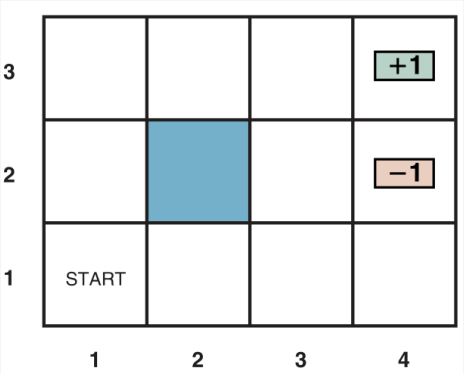


What if the actions are stochastic?

- 80% chance of going in the intended direction
- 10% chance of going in each of sideways directions



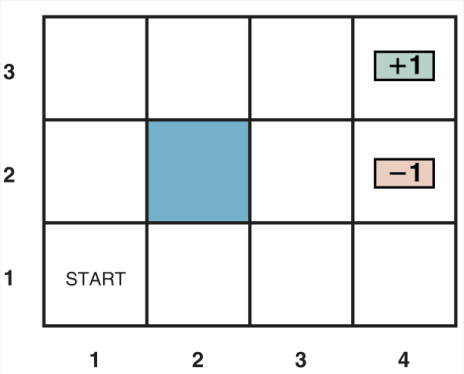
Sequential Decision processes, with stochastic actions



Action sequence [Up, Up, Right, Right, Right]

- has probability $0.8^5 = 0.32768$ of working has intended

Sequential Decision processes, with stochastic actions

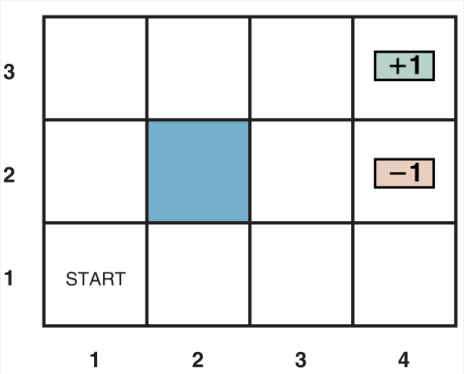


Action sequence [Up, Up, Right, Right, Right]

- has probability $0.8^5 = 0.32768$ of working has intended

Could it get us to the goal through another (non-intended) path? With what probability?

Sequential Decision processes, with stochastic actions



Action sequence [Up, Up, Right, Right, Right]

- has probability $0.8^5 = 0.32768$ of working has intended

Could it get us to the goal through another (non-intended) path? With what probability?

- yes, if the action results in the move sequence [Right, Right, Up, Up, Right] with probability $0.1^4 \times 0.8 = 0.00008$

Markov Decision Processes

States and actions:

- S: set of states
- $Actions(s)$: set of actions available in state s
  - a state with no available actions is a **terminal** state

Transition model:

- $P(s'|s,a)$: probability of reaching state s' after taking action a in state s
  - **markovian**: only depends on current state and action (and not on history)

Reward model:[1]

- $R(s,a,s')$: reward received after taking action a in state s and reaching state s'
  - 1 for reaching the goal, -1 for reaching the negative goal
  - -0.04 for each other transition (action cost, encouraging to reach the goal quickly)

[1] a negative reward can be interpreted as a cost

2048 as an MDP

- **States**: all possible board configurations
- **actions**: subset of {up, down, left, right} (the ones that change the board)
- **Transition model**:
  - first move all tiles in the direction of the action (merging identical tiles)
  - the uniformely select an empty time
  - place a 2 or 4 tile with respective probabilities 0.9 and 0.1
- **reward**: the value of any newly merged tile

Policies

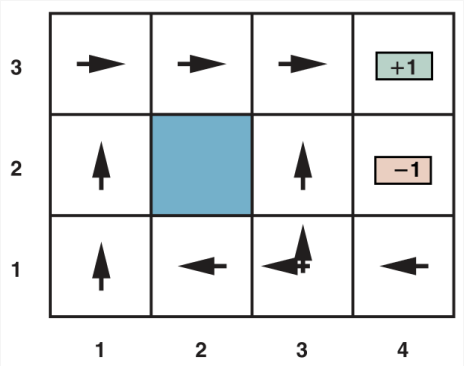
A finite sequence of actions cannot solve the problem in the stochastic case.

A solution must be a **policy** π that specifies the action to take in any state it may reach.

- $\pi(s)$: action to take in state s
  - where s is determined from the last percept

An **optimal policy** π^* is one that maximizes the **expected utility** of the agent.

Optimal Policies (example)



Two optimal policies for the grid environment, stating which actions to do in each state.

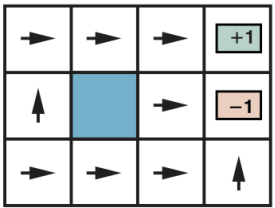
The two policies differ only in the action to in the $(1, 3)$ state:

- *Left*: Longer path, but safer
- *Right*: Shorter path, but riskier

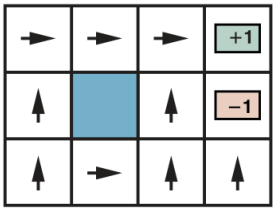
Note: policies a computed for

- a utility defined as the sum of rewards
- a transition reward of -0.04 for each transition into a non-terminal state

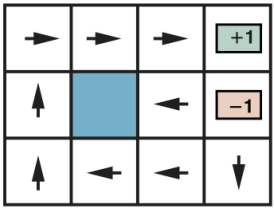
Optimal Policies (example with variable rewards)



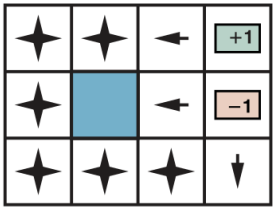
$r < -1.6497$



$-0.7311 < r < -0.4526$



$-0.0274 < r < 0$



$r > 0$

Optimal policies when changing the transition reward r (previously -0.04)

- top-left: very high cost of transition, agents desperately tries to exit
- top-right: high cost, agent attempts for the goal but is ok with the (-1) state if goal to far
- bottom-left: low cost, agent avoids any risk
- bottom-right: positive reward for any action, agent avoid terminal state to gather rewards indefinitely

Utilities over time

What is the utility of an an history: $U_h([s_0, a_0, s_1, a_1, s_2, \ldots, s_n])$?

Over this history we would collects the rewards:

- $R(s_0, a_0, s_1)$
- $R(s_1, a_1, s_2)$
- $\ldots$
- $R(s_{n-1}, a_{n-1}, s_n)$

How do we aggregate them?

Additive discounted rewards

$$
\begin{aligned}
U_h([s_0, a_0, s_1, a_1, s_2, \ldots, s_n]) = & R(s_0, a_0, s_1) \\
& + \gamma R(s_1, a_1, s_2) \\
& + \gamma^2 R(s_2, a_2, s_3) \\
& + \ldots \\
& + \gamma^{n-1} R(s_{n-1}, a_{n-1}, s_n)
\end{aligned}
$$

where γ is the **discount factor** $(0 \leq \gamma \leq 1)$

Why add a discount factor?

- empirical: humans and animals tend to prefer immediate rewards
- economics: money now is worth more than money later (it would produce interests and its value may decrease due to inflation)
- uncertainty about the true rewards: our model is not perfect, and the more time passes the most likely we are to be wrong
- convenience: the sum of discounted rewards converges to a finite value, even with infinite action sequences

Section 3

Exercise: discount factor for interest rates

Exercise: discount factor for interest rates

I am interested in maximizing the money I have in my bank account in the long term.

The interest rate is 5% per time-step:

What should be the discount factor γ to represent this situation?

Exercise: discount factor for interest rates

I am interested in maximizing the money I have in my bank account in the long term.

The interest rate is 5% per time-step:

What should be the discount factor γ to represent this situation?

- If I have 100 euros in my account now, I will have 105 euros in one time-step.
- I should value 105 euros in the next step as much as 100 euros now.

Exercise: discount factor for interest rates

I am interested in maximizing the money I have in my bank account in the long term.

The interest rate is 5% per time-step:

What should be the discount factor γ to represent this situation?

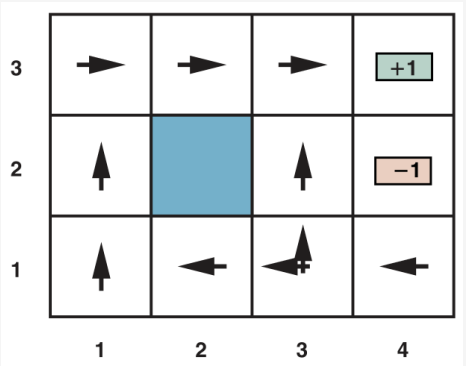
- If I have 100 euros in my account now, I will have 105 euros in one time-step.
- I should value 105 euros in the next step as much as 100 euros now.

Let M be the amount of money I have now, and I the interest rate (e.g. 0.05).

- $M = \gamma \times (M \times (1 + I))$
- $\gamma = \frac{1}{1+I} = \frac{1}{1.05} \approx 0.9524$

Optimal policies and the utilities of states

Given an initial state (1,1) and a policy π, I can determine my probability of being in any state after n steps.



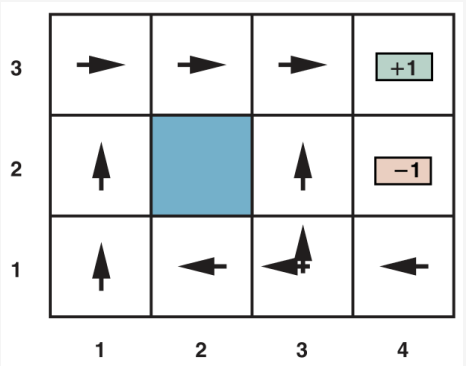
Step 0:

- $p((1,1)) = 1$

Step 1:

Optimal policies and the utilities of states

Given an initial state (1,1) and a policy π, I can determine my probability of being in any state after n steps.



Step 0:

- $p((1,1)) = 1$

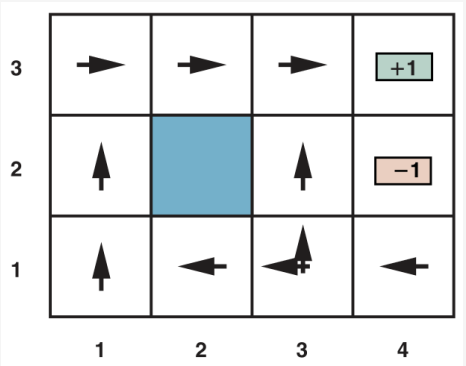
Step 1:

- $p((2,1)) = 0.8$
- $p((1,2)) = 0.1$
- $p((1,1)) = 0.1$

Step 2:

Optimal policies and the utilities of states

Given an initial state (1,1) and a policy π, I can determine my probability of being in any state after n steps.



Step 0:

- $p((1,1)) = 1$

Step 1:

- $p((2,1)) = 0.8$
- $p((1,2)) = 0.1$
- $p((1,1)) = 0.1$

Step 2:

- $p((3,1)) = 0.8^2$
  - moving from (2,1)
- $p((2,1)) = 0.8 \times 0.1 + 0.8 \times 0.1 + 0.1 \times 0.8$
  - not moving from (2,1) and moving from (1,1)
- ...

Utilities of policy π in state s

The utility of a state s under a policy π is the expected utility of the history starting from s and following π.

$$U^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(S_t, \pi(S_t), S_{t+1})\right]$$

where the *expectation* E is taken with respect to the probability distribution over all possible histories starting from s and following π.

Optimal Policy and Utility of States

Of all the possible policies the agent could choose to execute in s, one (or more) will have higher expected utility than all the others.

$$\pi_s^* = \arg\max_\pi U^\pi(s)$$

Under the common assumptions[2] the optimal policy is independent of the initial state.

- from the markovian property, the optimal policy is a function of the current state only: optimal policies have no reason to disagree on the action to take in a given state
- noted as π^* (optimal policy)

The **utility of a state** is the utility the agent would get by following the optimal policy from that state.

$$U(s) = U^{\pi^*}(s)$$

[2] discounted utilities and infinite horizon, (i.e. no fixed deadline for the agent)

Characterizing Utility: Bellman Equation

The optimal policy is the one that maximizes the expected utility of the agent, computed from immediate rewards and the utility of successor states.

$$\pi^*(s) = \underset{a \in Actions(s)}{\arg\max} \sum_{s'} P(s'|s,a) \left[R(s,a,s') + \gamma U(s') \right]$$

The **Bellman equation** characterizes the utility of a state in terms of the utility of its successor states (when the agent follows an optimal policy).

$$U(s) = \max_{a \in Actions(s)} \sum_{s'} P(s'|s,a) \left[R(s,a,s') + \gamma U(s') \right]$$

Action Utility

The **action utility function** (or Q-function) $Q(s, a)$ is the expected utility of doing action a in state s and then following the optimal policy.

$$Q(s, a) = \sum_{s'} P(s'|s, a) \left[R(s, a, s') + \gamma U(s')\right]$$

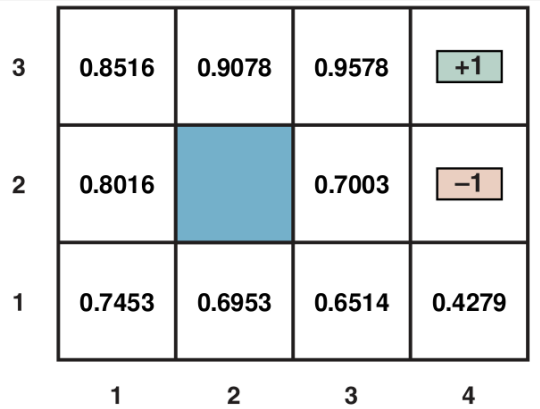
We can redefine the state utility and optimal policy in terms of the action utility:

$$U(s) = \max_{a \in Actions(s)} Q(s, a)$$

$$\pi^*(s) = \arg\max_{a \in Actions(s)} Q(s, a)$$

Utility: Solution to the Bellman Equation

The utilities of the states are the solution to the Bellman equation.



(Approximated) Utility of states with $\gamma = 1$ and $r = -0.04$.

Reward shaping (example)

Exercise: I want my pet to learn to do a sequence of 3 actions.

- I can give it a reward of 3 sweets when it does the right sequence
- I can give it a reward of 1 sweet for each action it does correctly

Are the two MDP equivalent?

- No, the utilities would be different
- But, the optimal policy would be the same

Reward shaping theorem

Let $\Phi(s)$ be *any* function of the state.

The optimal policy is the same for the MDP with reward $R(s, a, s')$ and the MDP with reward

$$R'(s, a, s') = R(s, a, s') + \gamma\Phi(s') - \Phi(s)$$

- The utilities are just a mean to an end: the optimal policy.
- Reward shaping can be used to make the problem easier to solve, by making the optimal policy easier to find.
  - critical in reinforcement learning (immediate rewards guide the learning process)

Section 4

Algorithms for MDPs

Algorithms for MDPs

- offline algorithms: compute the optimal policy before the agent starts acting
  - generate the optimal policy for all states
  - value iteration, policy iteration, linear programming, ...
- online algorithms: approximate the optimal policy while the agent is acting
  - expectimax, Monte-Carlo Tree Search, Q-learning, ...

Solving MDPs offline

For each state s, we have the Bellman equation:

$$U(s) = \max_{a \in Actions(s)} \sum_{s'} P(s'|s,a) \left[R(s,a,s') + \gamma U(s') \right]$$

- With n states, we have n equations with n unknowns (the utilities of the n states).
- problem: the equations are
  - non-linear (max operator)
  - coupled: $U(s)$ may appear in the equation for $U(s')$ and vice-versa

Solving MDPs offline: Value Iteration

Value iteration is an iterative algorithm that approximates the utilities of the states.

- start with an initial guess for the utilities (typicaly $U(s) = 0 \forall s$)
- iteratively update the utilities of the states by incorporating the utilities of the successor states

Let $U_i(s)$ be the utility of state s at the i-th iteration.

$$U_0(s) = 0$$
$$U_{i+1}(s) = \max_{a \in Actions(s)} \sum_{s'} P(s'|s,a) \left[R(s,a,s') + \gamma U_i(s') \right] \quad (Bellmanupdate)$$

- At step 1: $U_1(s)$ is the utility of the state if the agent could only do one action.
- At step 2: $U_2(s)$ is the utility of the state if the agent could do two actions, etc.

Solving MDPs offline: Value Iteration

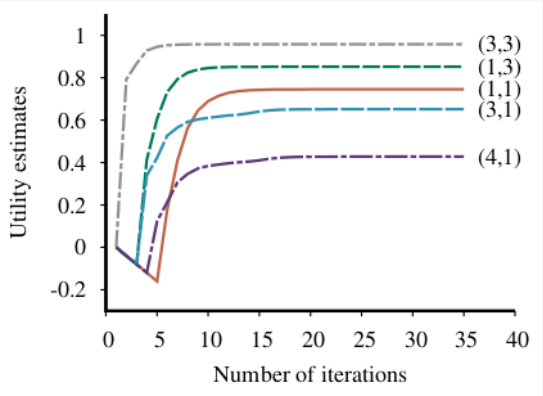
function VALUE-ITERATION(mdp, ϵ) **returns** a utility function
  inputs: mdp, an MDP with states S, actions $A(s)$, transition model $P(s' \mid s, a)$,
    rewards $R(s, a, s')$, discount γ
    ϵ, the maximum error allowed in the utility of any state
  local variables: U, U', vectors of utilities for states in S, initially zero
    δ, the maximum relative change in the utility of any state

  repeat
    $U \leftarrow U'$; $\delta \leftarrow 0$
    for each state s **in** S **do**
      $U'[s] \leftarrow \max_{a \in A(s)}$ Q-VALUE(mdp, s, a, U)
      if $|U'[s] - U[s]| > \delta$ **then** $\delta \leftarrow |U'[s] - U[s]|$
  until $\delta \leq \epsilon(1-\gamma)/\gamma$
  return U

Solving MDPs offline: Value Iteration

Some properties of the value iteration algorithm:

- the error will decrease at each iteration
- **convergence**: the utilities of the states converge to the optimal utilities
  - under the assumption that $\gamma < 1$



Conclusion

- **Markov Decision Processes (MDPs)** are a formalism to model
  - sequential decision processes
  - with stochastic actions
  - and rewards
- The **optimal policy** is the one that maximizes the expected utility of the agent
- The **Bellman equation** characterizes the utility of a state in terms of the utility of its successor states
- Offline algorithms like **value iteration**
  - can be used to compute the optimal policy before the agent starts acting
  - can be hard to compute for large problems (requires the utility function for every state)
- Online algorithms (next course)
  - approximate the optimal policy while the agent is acting (i.e. under computation constraints)