Reinforcement Learning

An Introduction

Instructor

Hussam ATOUI

- Software Engineer at Valeo Créteil (France) since Nov 2022
- PhD-Cifre RENAULT & Grenoble-Alpes University (2019-2022)
- Specialities: Automated Driving, Reinforcement Learning, Automatic Control, Optimization

Instructor

Victor MORAND
morand@isir.upmc.fr

- **PhD Student at ISIR (Sorbonne - CNRS)**
  - Explaining how LLMs manipulate Knowledge
  - *towards AIs that know what they know*
- **Also out of a School of Engineering !**

Feel free to reach out at the end of our sessions !

Course Content

1. Introduction to Reinforcement Learning
2. Markov Decision Processes (MDPs)
3. Policy and Value Functions
4. Dynamic Programming (DP) for RL
5. Model-Free methods
6. Value Function Approximation
7. Policy-Gradient and Actor-Critic Methods
8. Deep RL
9. TP Project

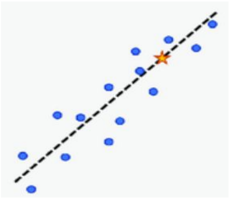
Introduction to Reinforcement Learning (RL)

Types of Learning

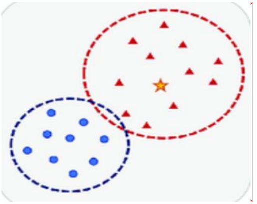
Supervised Learning

Train with labeled data

Regression



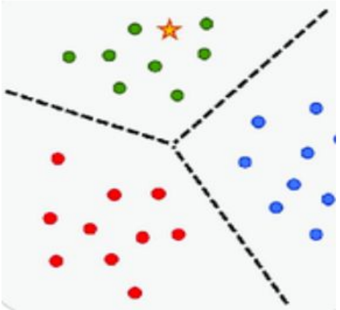
Classification



Unsupervised Learning

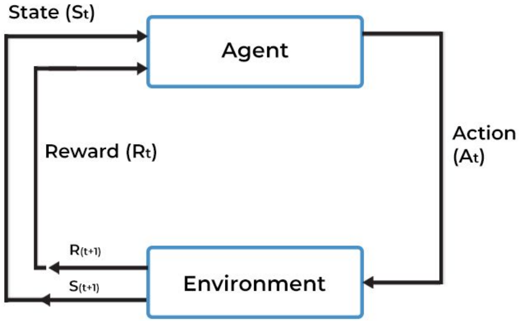
Train with unlabeled data

Clustering

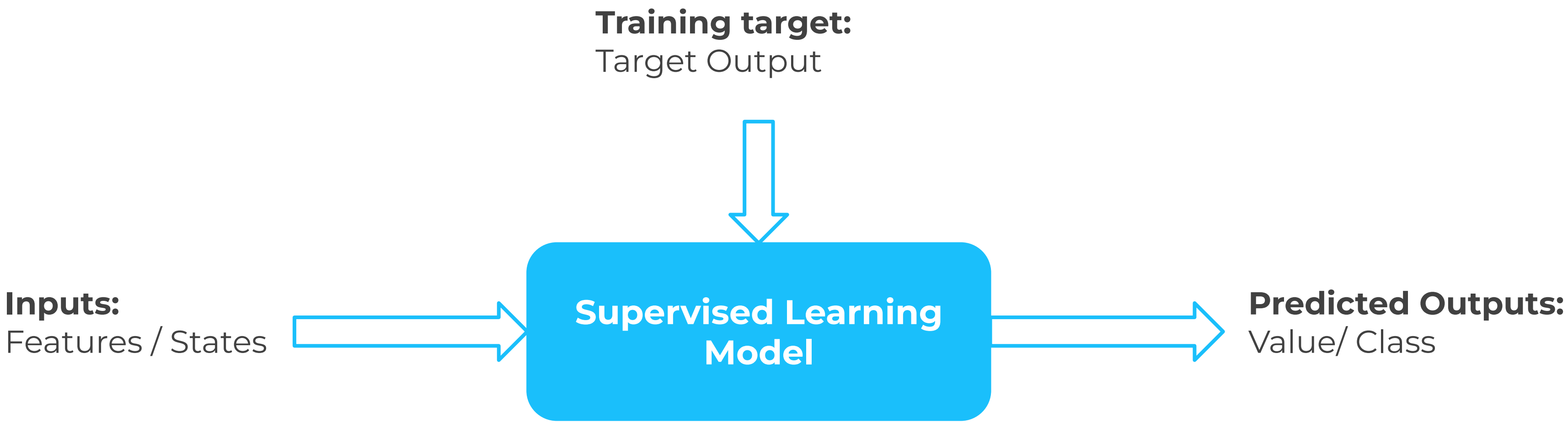


Reinforcement Learning

Train with environment experience



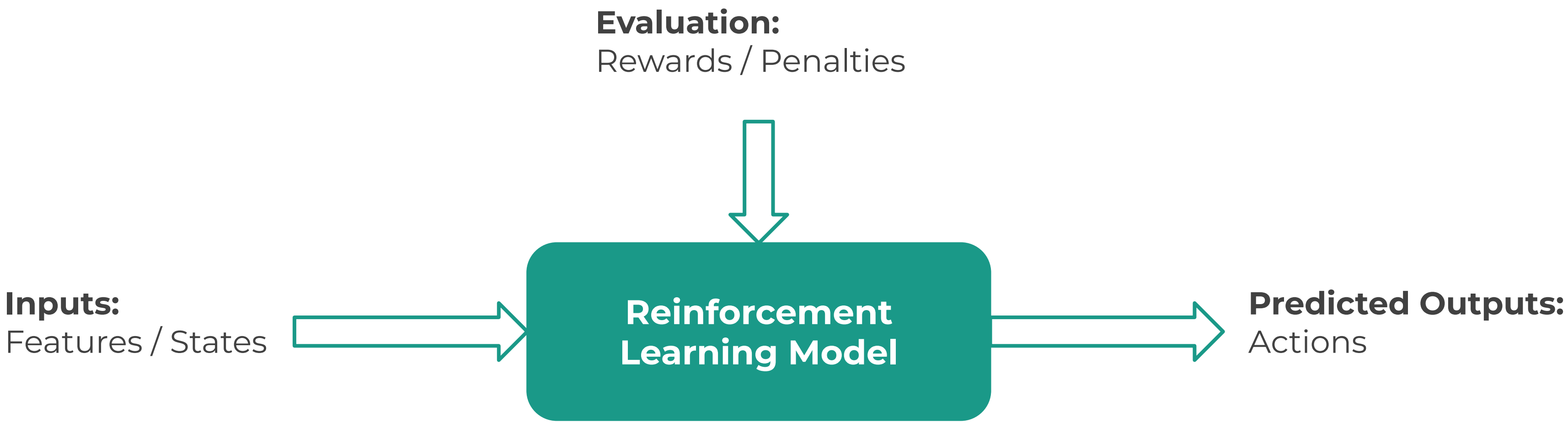
Supervised Learning



- **Error:** Target Output - Predicted Output
- **Objective:** Minimize the error between the target and the predicted output

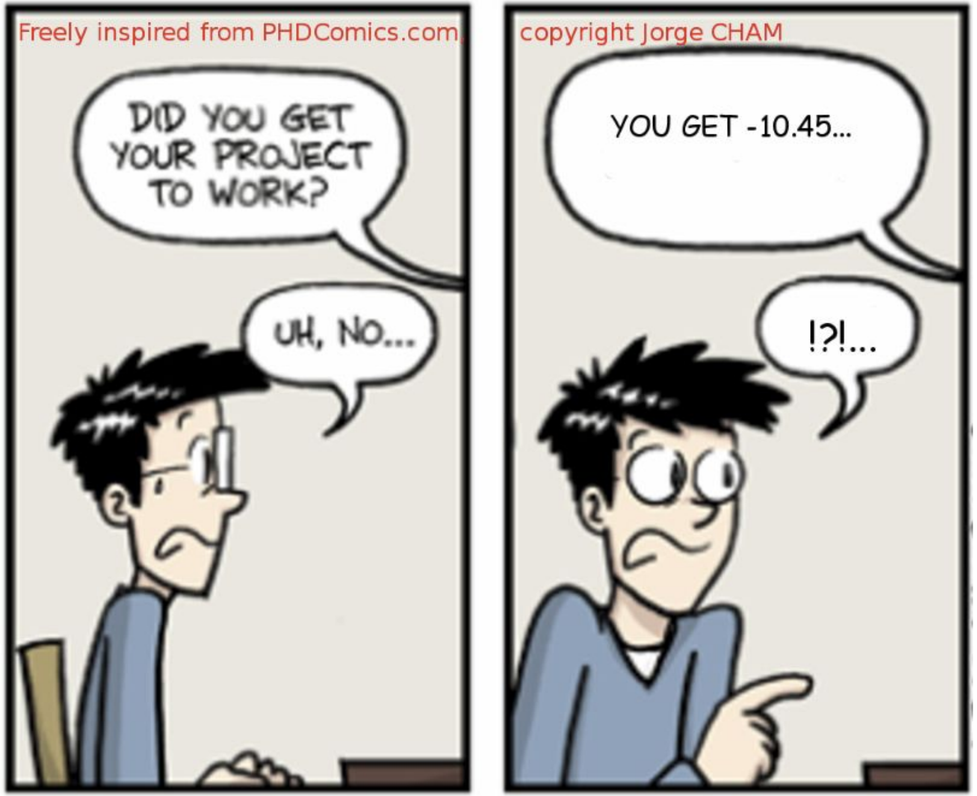
Supervised Learning

Reinforcement Learning



- **Error:** Awards - Penalties
- **Objective:** Maximize the awards and decrease penalties as much as possible

Reinforcement Learning



Examples of Rewards [1]

- **Fly stunt manoeuvres in a helicopter**
  - +ve reward for following desired trajectory
  - −ve reward for crashing
- **Manage an investment portfolio**
  - +ve reward for each $ in bank
- **Control a power station**
  - +ve reward for producing power
  - −ve reward for exceeding safety thresholds
- **Make a humanoid robot walk**
  - +ve reward for forward motion
  - −ve reward for falling over
- **Play many different Atari games better than humans**
  - +/−ve reward for increasing/decreasing score

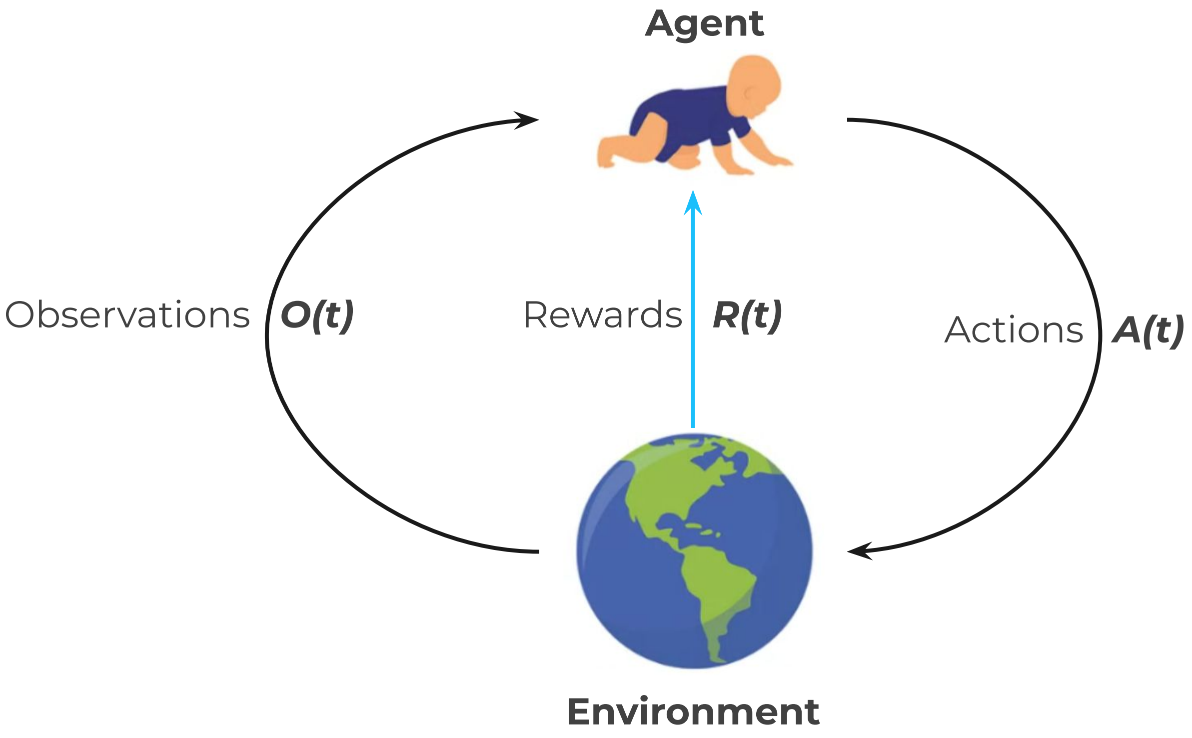
Agent and Environment

At step ***t***:

The **Agent:**
- Receives ***O(t)***
- Receives ***R(t)***
- Executes ***A(t)***

The **Environment:**
- Receives ***A(t)***
- Emits ***O(t+1)***
- Emits ***R(t+1)***

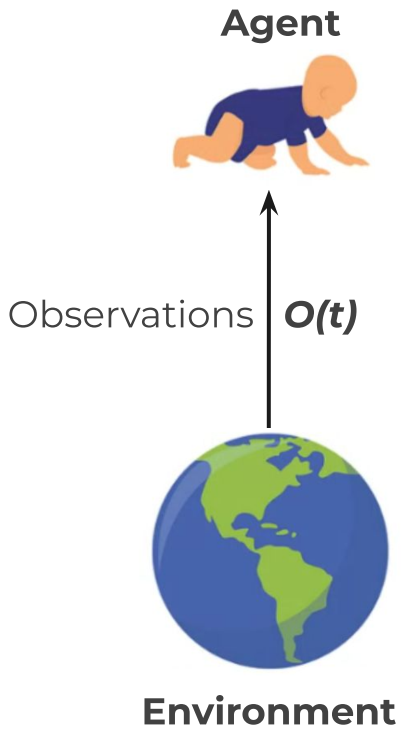
t++



Fully Observable Environment

- Environment observations = Agent state

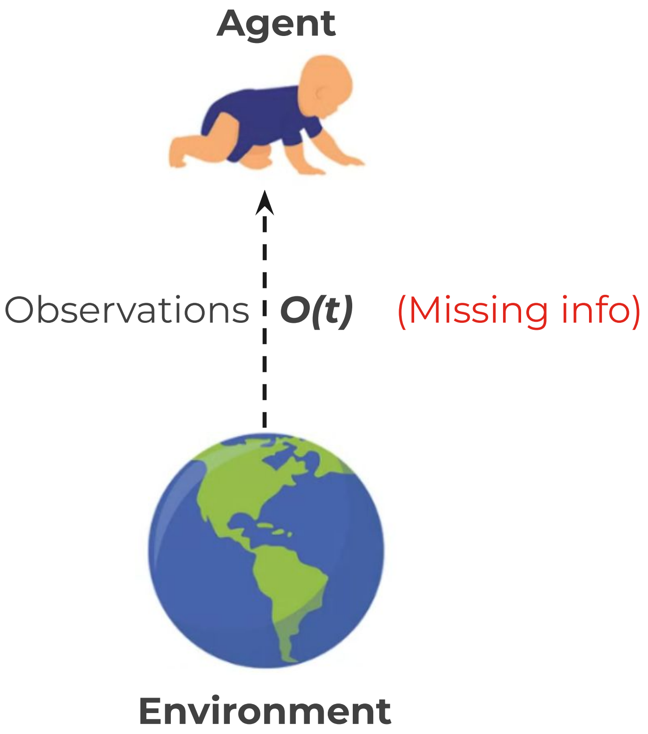
- This is assumed in **Markov Decision Process (MDP)**



Partially Observable Environment

- **Environment observations ≠ Agent state**
  - A **drone** navigating a forest only sees nearby obstacles.
  - A **healthcare** agent observes patient symptoms but not the underlying disease.
  - A **self-driving car** detects nearby vehicles but not hidden pedestrians.
  - A **weather forecasting** model observes recent conditions but not future patterns

- This is called **Partially Observable Markov Decision Process (POMDP)**



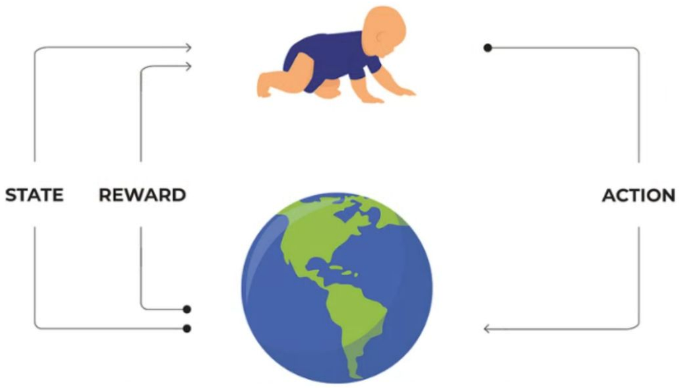
Reinforcement Learning

General Architecture

Agent: The system that takes actions to be trained.

State: The information required by the agent to take an action. This info is observed from the environment.

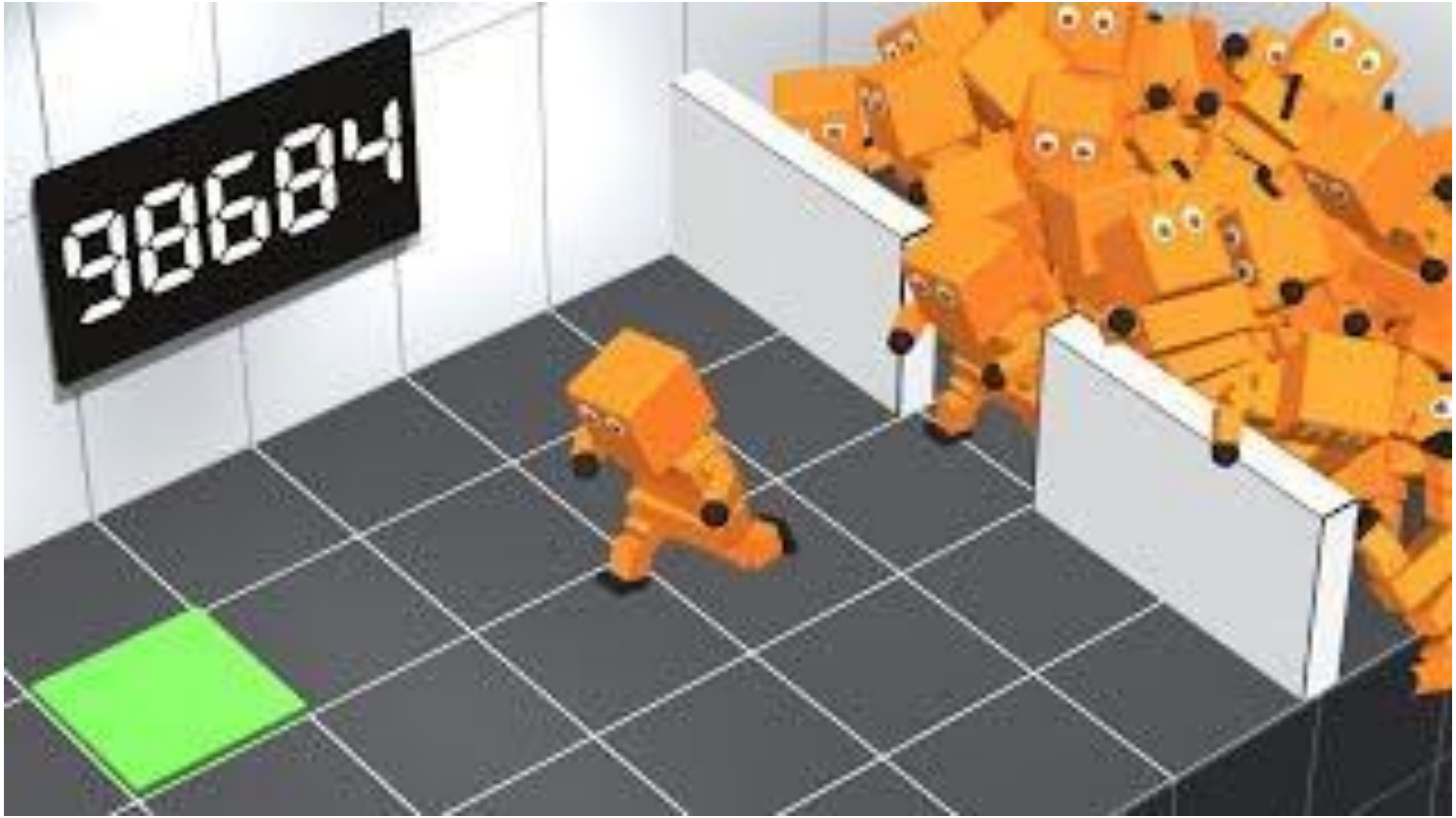
Reward: Feedback received by the agent to evaluate the taken action under a certain state.



Action: The decision or move that the agent makes at a particular state

Environment: The external system with which the agent interacts.

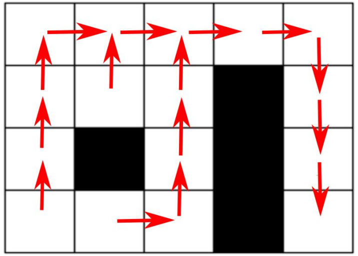
Reinforcement Learning



RL Agent

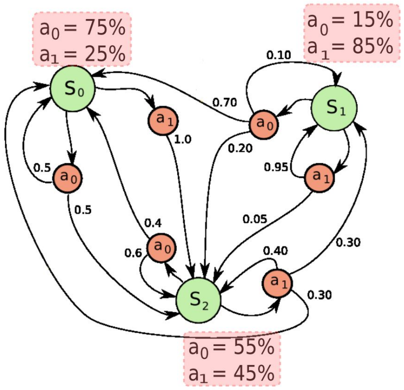
Policy

A **policy** defines the agent's behavior in the environment. It represents a mapping from states to actions, for example:



- **Deterministic policy:** $a = \pi(s)$,

  where the action ***a*** is chosen directly based on state ***s***.



- **Stochastic policy:** $\pi(a|s) = P[A_t = a | S_t = s]$,

  where the policy gives the probability of taking action ***a*** given state ***s***.

RL Agent

Value Function

A value function

- estimates the expected future reward
- assesses the quality of states, helping to determine the best actions to take.

For example, the state value under policy π is given by:

$$v_\pi(s) = \mathbb{E}_\pi \left[R_{t+1} + \quad R_{t+2} + \quad R_{t+3} + \cdots \mid S_t = s \right]$$

This equation expresses the expected sum of discounted rewards starting from state s.

RL Agent

Value Function

A value function

- estimates the expected future reward
- assesses the quality of states, helping to determine the best actions to take.

For example, the state value under policy π is given by:

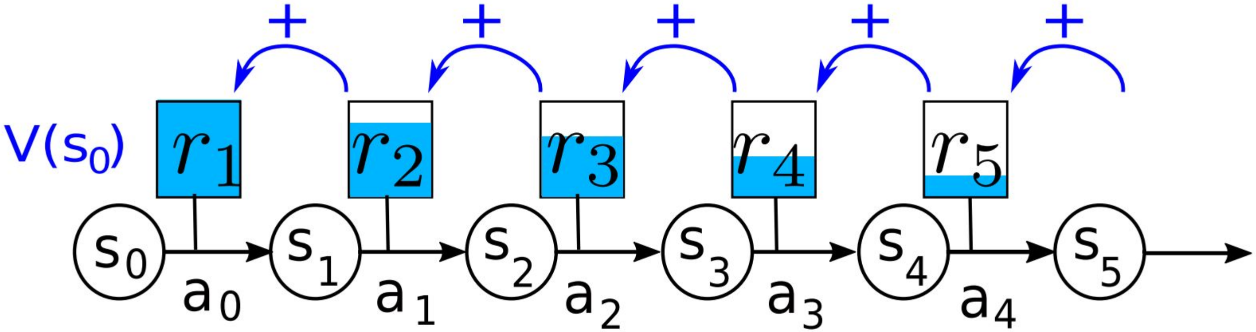
$$v_\pi(s) = \mathbb{E}_\pi \left[R_{t+1} + \boxed{\gamma} R_{t+2} + \boxed{\gamma^2} R_{t+3} + \cdots \mid S_t = s \right]$$

This equation expresses the expected sum of discounted rewards starting from state s.

RL Agent

Value Function

$$v_\pi(s) = \mathbb{E}_\pi\left[R_{t+1} + \boxed{\gamma} R_{t+2} + \boxed{\gamma^2} R_{t+3} + \cdots \mid S_t = s\right]$$



γ∈[0,1]:

- If **γ=0**, the agent focuses **solely on immediate** rewards.
- If **γ=1**, **future rewards** are valued equally to immediate rewards.

RL Agent

Model

A model forecasts the environment's next state and expected reward:

- ***P*** represents the probability of the next state given the current state and action:

$$P^a_{s,s'} = P[S_{t+1} = s' \mid S_t = s, A_t = a]$$

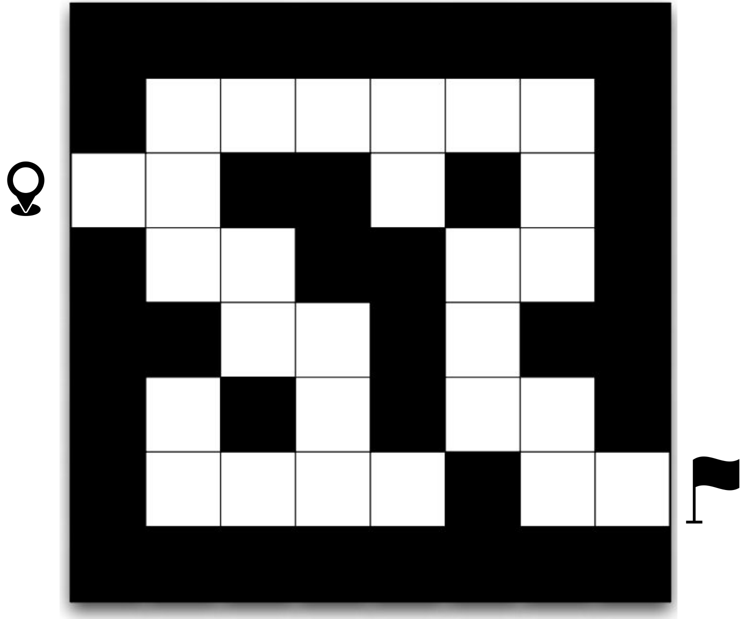
- ***R*** represents the expected immediate reward given the current state and action:

$$R^a_s = \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right]$$

Example: Maze [1]

States, Actions, Rewards

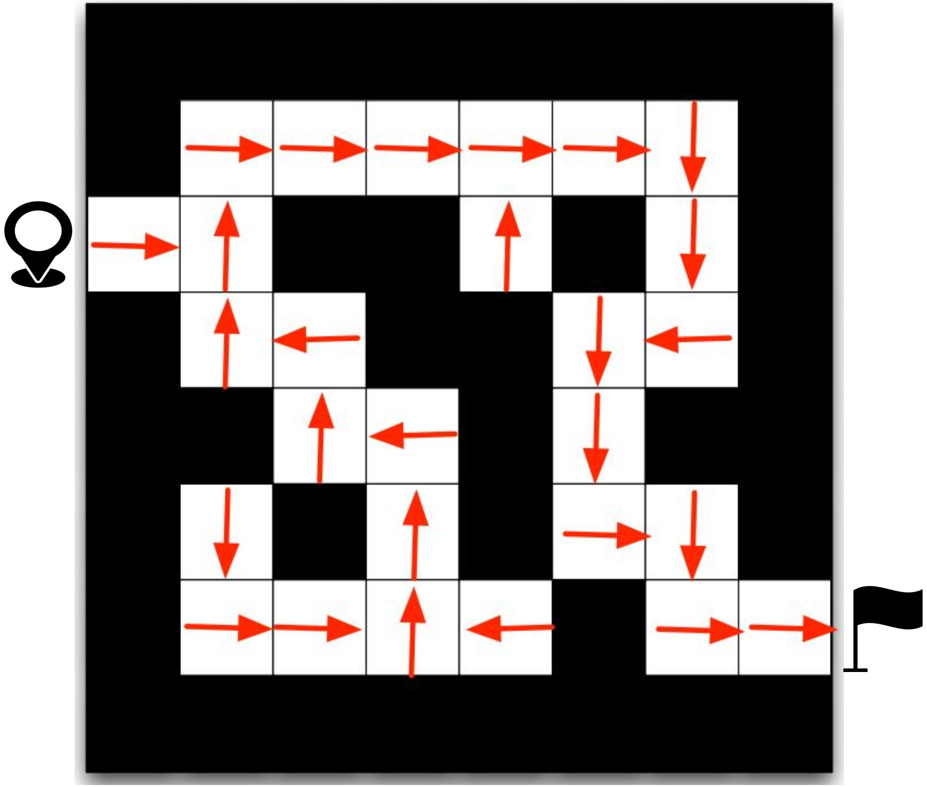
- **States:** Agent's location
- **Actions:** Right, Left, Up, Down
- **Rewards:** -1 per time-step



Example: Maze [1]

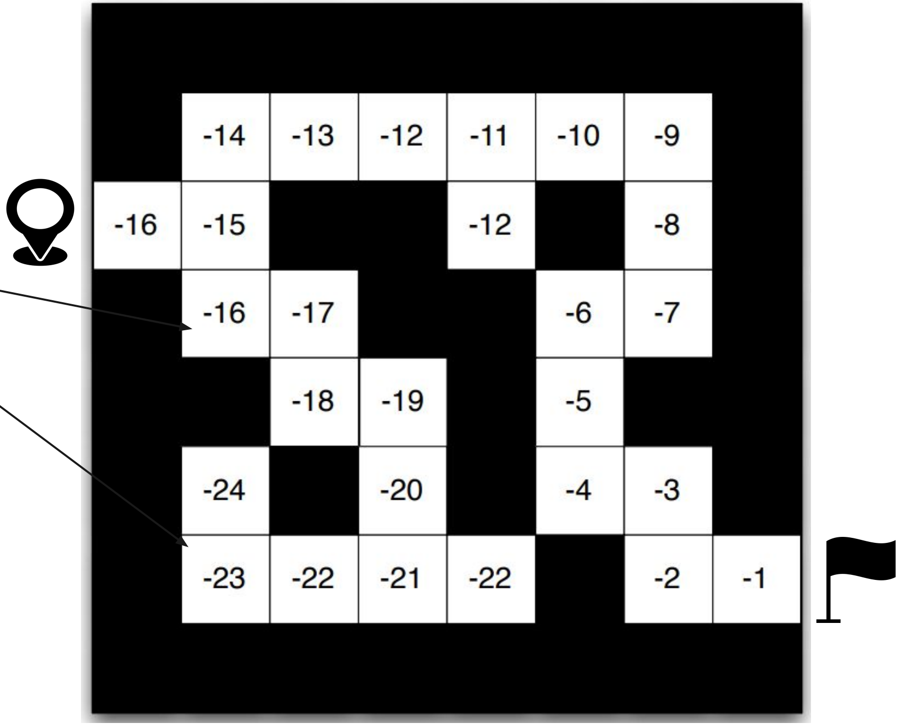
Policy

Arrows represent policy ***π(s)*** for each state ***s***



Example: Maze [1]

Numbers represent value $v_\pi(s)$ of each state ***s***



RL Agents

Different Types

- ➔ **Value-based:**
  - No Policy
  - Value Function
- ➔ **Policy-based:**
  - Policy
  - No Value Function
- ➔ **Actor-Critic:**
  - Policy
  - Value Function
- ➔ **Model-free:**
  - Policy and/or Value Function
  - No Model
- ➔ **Model-based:**
  - Policy and/or Value Function
  - Model

Markov Decision Processes (MDPs)

Markov Process

- A Markov Process is a memoryless process where the future state depends **only** on the **current state** and **not on any past states**.
- Formally, a Markov Process is a tuple: ***M=(S,P)***

Where:

- ***S*:** A finite set of states.
- ***P*:** Transition probabilities between states, defined as:

$$P(s'|s) = \Pr(S_{t+1} = s' \mid S_t = s)$$

The Markov Property

- **Markov property:** Future depends only on the present, not past states
- Simplifies state transition modeling

$$P(S_{t+1}|S_t) = P(S_{t+1}|S_1, S_2, \ldots, S_t)$$

Markov Reward Process

- A Markov Reward Process is a Markov Process with added rewards.
- It is represented as a tuple: ***MR=(S,P,R,γ)***

Where:

- ***R(s):*** Reward function providing the expected reward at each state ***s***,

$$R(s) = \mathbb{E}[R_{t+1} \mid S_t = s]$$

- ***γ:*** Discount factor, controlling the importance of future rewards.

$$0 \leq \gamma < 1$$

Markov Reward Process

Cumulative Reward - Gain

- **Cumulative Reward *G(t)* :** Expected cumulative reward from state ***s***

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- **(State-)Value Function *v(s)* :** Expected state-value of state ***s***

$$v(s) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right]$$

Bellman Equation

State-Value Function

- The state-value function can be presented as an immediate reward and future reward as follows:

$$v(s) = \mathbb{E}\left[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s\right]$$

PROOF?

Bellman Equation

Proof

$$v(s) \overset{?}{=} \mathbb{E}\left[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s\right]$$

$$v(s) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right]$$

$$\Longrightarrow v(s) = \mathbb{E}\left[R_{t+1} + \gamma G_{t+1} \mid S_t = s\right]$$

$$\boxed{G_{t+1} = \sum_{k=0}^{\infty} \gamma^k R_{t+k+2}}$$

$$v(S_{t+1}) = \mathbb{E}\left[G_{t+1} \mid S_{t+1}\right]$$

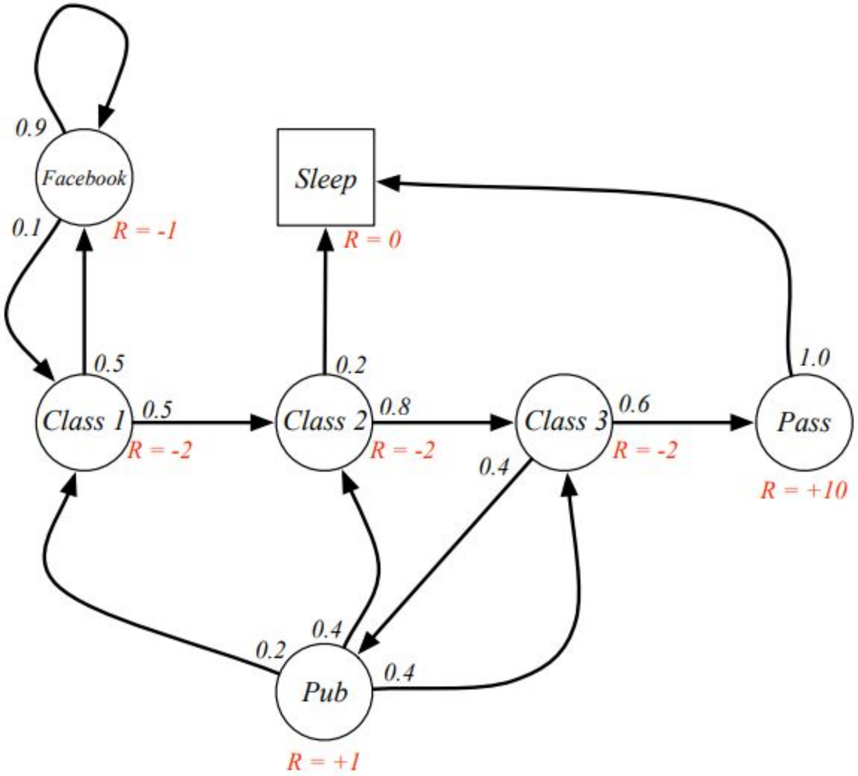
$$\Longrightarrow v(s) = \mathbb{E}\left[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s\right]$$

Stochastic Eq.

$$v(s) = \sum_{s' \in S} P(s'|s)\left[R(s,s') + \gamma v(s')\right]$$

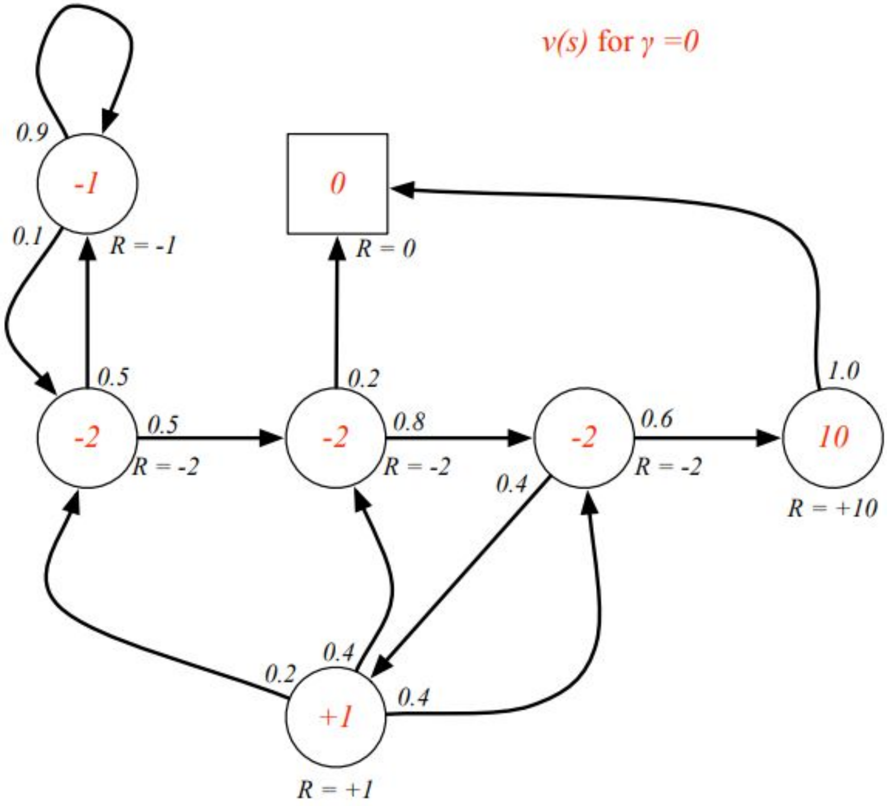
Example

Example: Student MRP (P, S, R) [1]



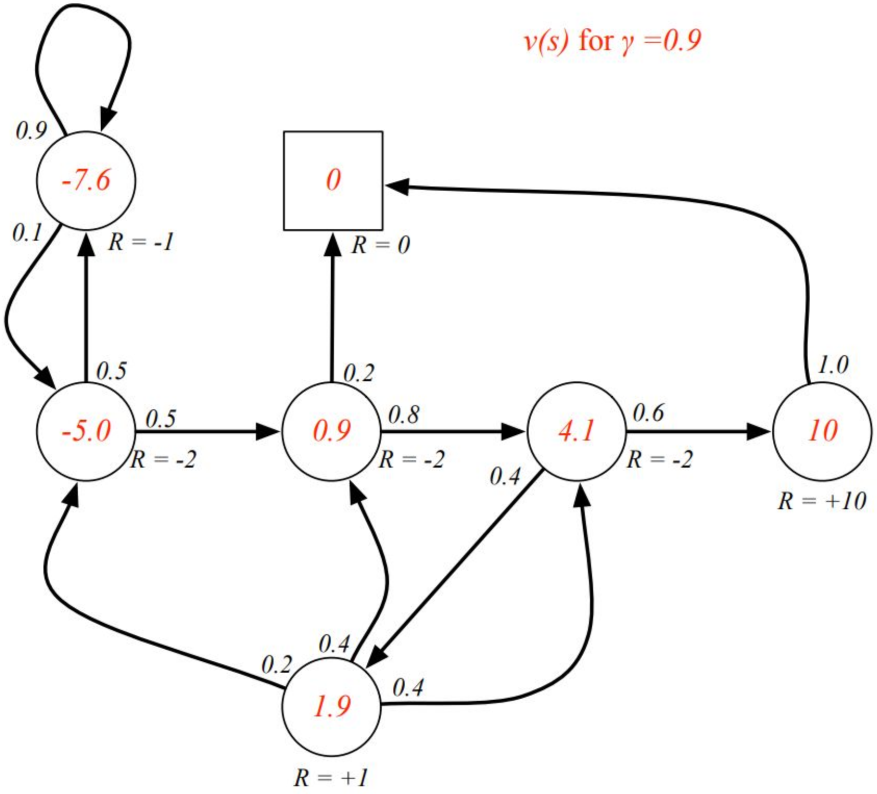
Example: Student MRP (P, S, R) [1]

Discount factor effect



Example: Student MRP (P, S, R) [1]

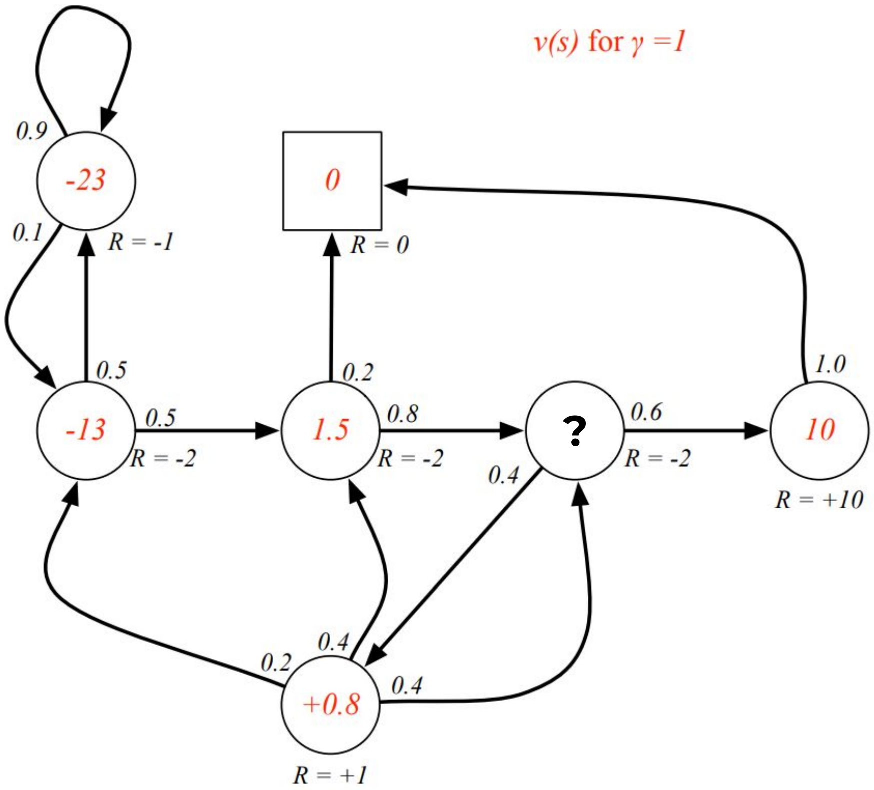
Discount factor effect



Exercise

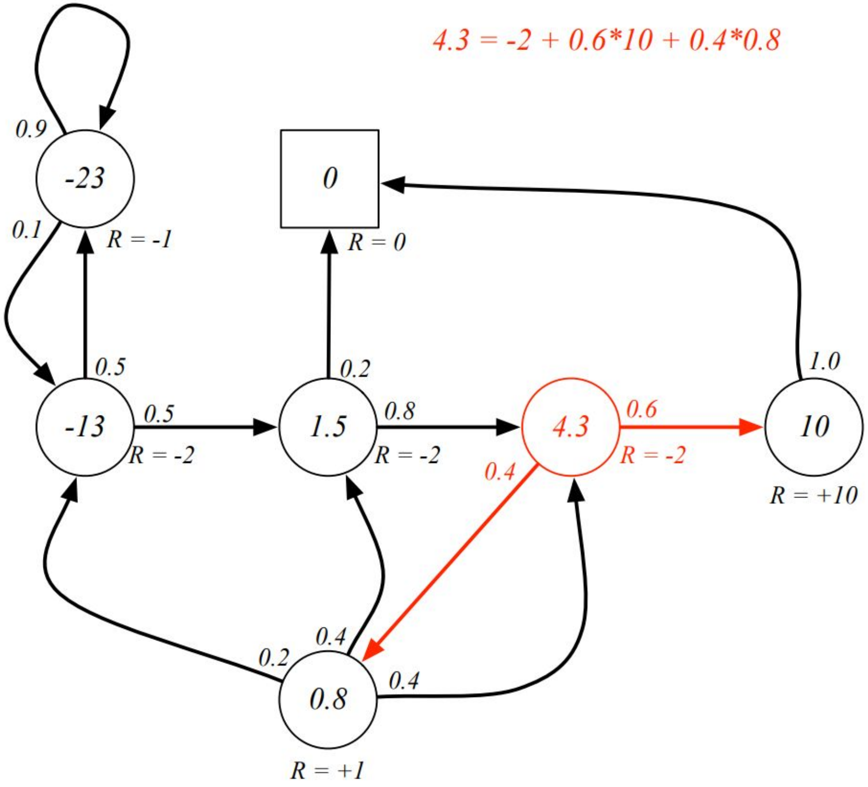
Example: Student MRP (P, S, R) [1]

Discount factor effect



Example: Student MRP (P, S, R) [1]

Example of Bellman's equation



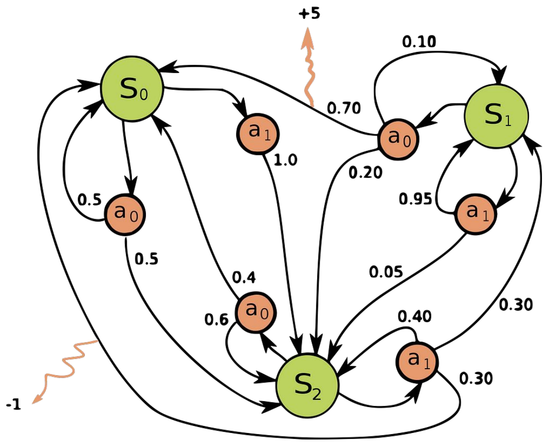
MRP → MDP

$(P, S, R) \rightarrow (P, S, A, R)$

Markov Decision Process (MDP)

A Markov decision process is a 4-tuple (S, A, P, R):

- **States (*S*):** Describe environment situations
- **Actions (*A*):** Choices available to the agent
- **Rewards (*R*):** Immediate feedback for actions
- **Transition Probabilities (*P*):** Likelihood of reaching a new state



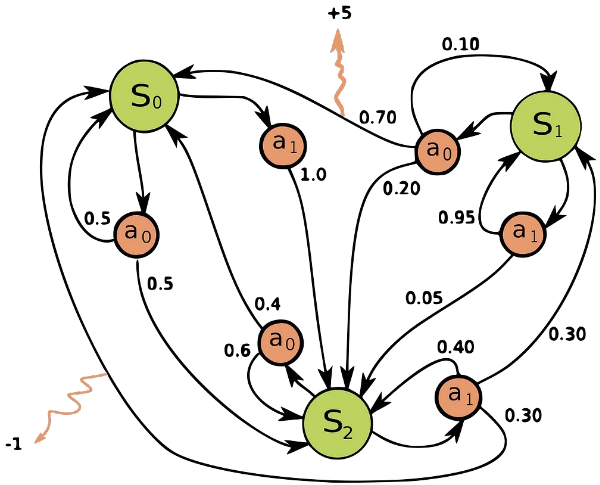
Note: A finite MDP is an MDP with finite state, action, and reward sets. Much of the current theory of reinforcement learning is restricted to finite MDPs.

Markov Decision Process

State Transitions - **Policy**

- **Transition probability:** P(s'|s,a)
- Models probability of moving to ***s'*** from ***s*** after action ***a***

$$P^a_{s,s'} = P(S_{t+1} = s' \mid S_t = s, A_t = a)$$

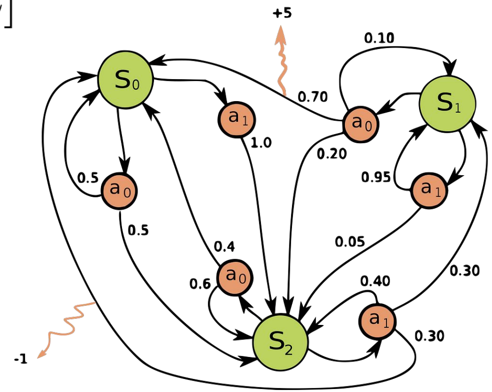


Markov Decision Process

Reward Function and Policy

- **Reward function *R(s,a)*:** Immediate feedback
- Positive rewards encourage actions; negative prevent actions

$$R(s,a) = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$$



- **Deterministic policy:** $a = \pi(s)$,

  where the action ***a*** is chosen directly based on state ***s***.

- **Stochastic policy:** $\pi(a|s) = P[A_t = a|S_t = s]$,

  where the policy gives the probability of taking action ***a*** given state ***s***.

Markov Decision Process

Policies

- **Deterministic policy:**

$$a = \pi(s)$$

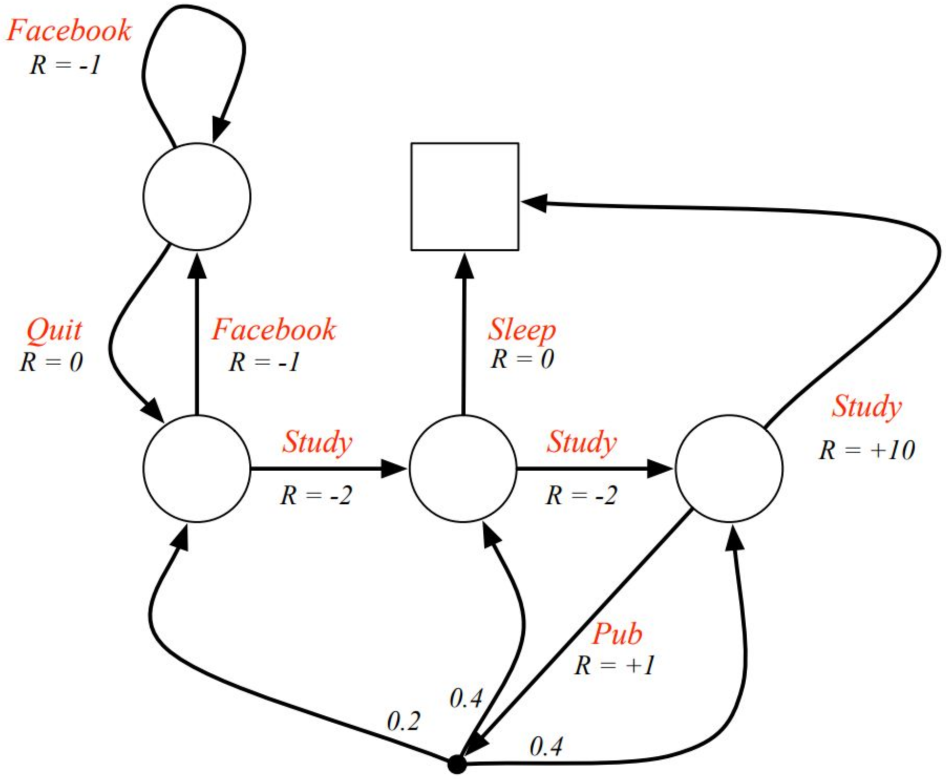
where the action ***a*** is chosen directly based on state ***s***.

- **Stochastic policy:**

$$\pi(a|s) = P[A_t = a|S_t = s]$$

where the policy gives the probability of taking action ***a*** given state ***s***.

Example: Student MDP (P, S, A, R) [1]



Markov Decision Process

Value Functions

- **State-value function :** Expected cumulative reward from state ***s*** under policy **π**

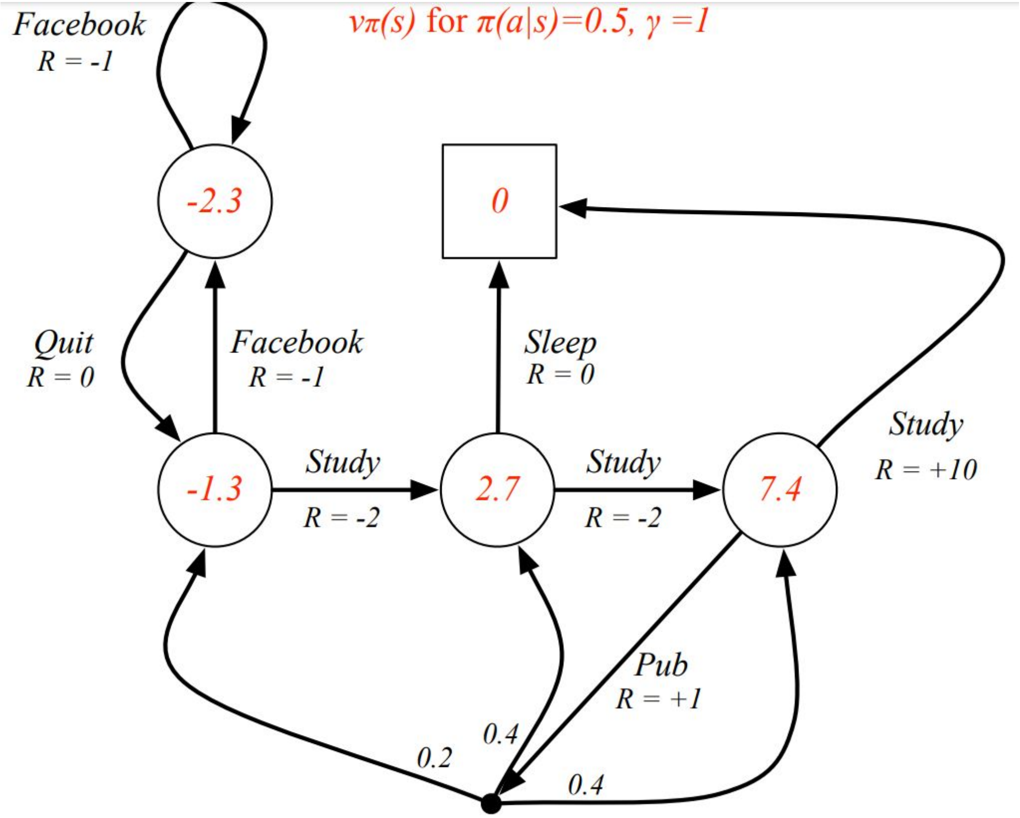
$$v_\pi(s) = \mathbb{E}_\pi\left[\overbrace{\boxed{\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}}}^{G_t} \mid S_t = s\right]$$

- **Action-value function:** Expected reward of taking action **a** in state s under policy **π**

$$q_\pi(s,a) = \mathbb{E}_\pi\left[\overbrace{\boxed{\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}}}^{G_t} \mid S_t = s, A_t = a\right]$$

Markov Decision Process

State-Value Function



Bellman Expectation Equation

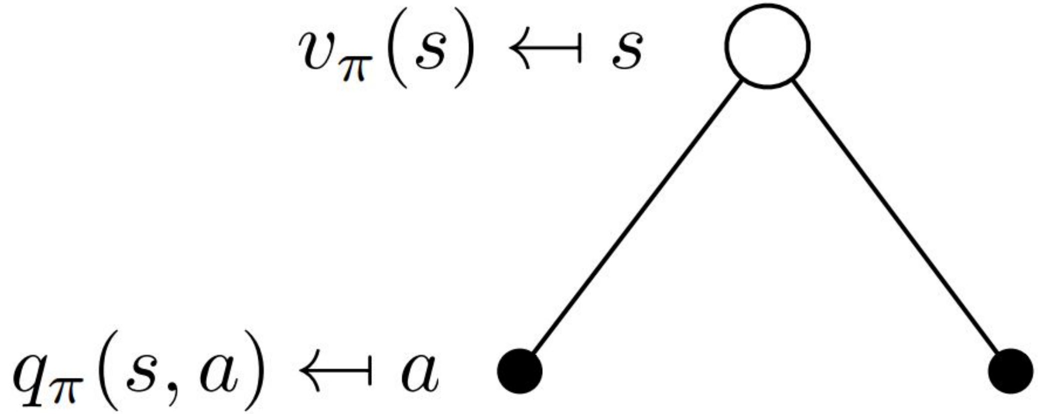
- **State-value function :** Expected cumulative reward from state ***s*** under policy **π**

$$v_\pi(s) = \sum_{a \in A} \pi(a|s) \sum_{s' \in S} P(s'|s,a)\left[R(s,a,s') + \gamma v_\pi(s')\right]$$

- **Action-value function:** Expected reward of taking action **a** in state s under policy **π**

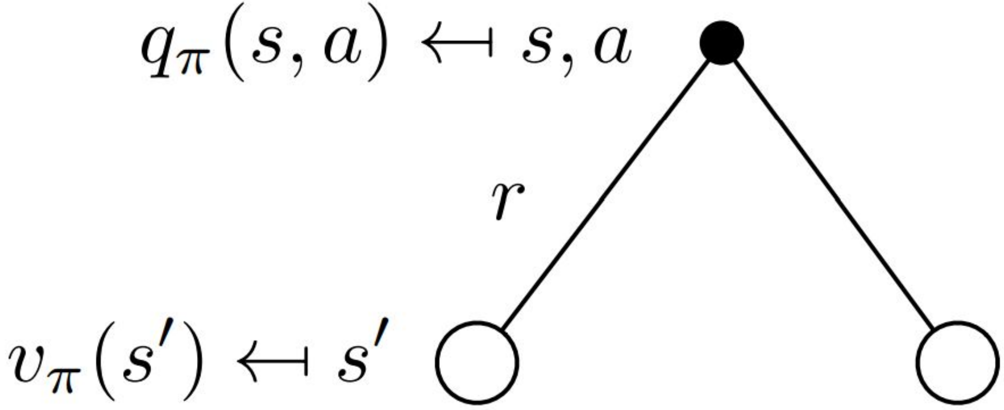
$$q_\pi(s,a) = \sum_{s'} P(s'|s,a)\left[R(s,a) + \gamma \sum_{a'} \pi(a'|s') q_\pi(s',a')\right] v_\pi(s)$$

Bellman Expectation Equation [1]



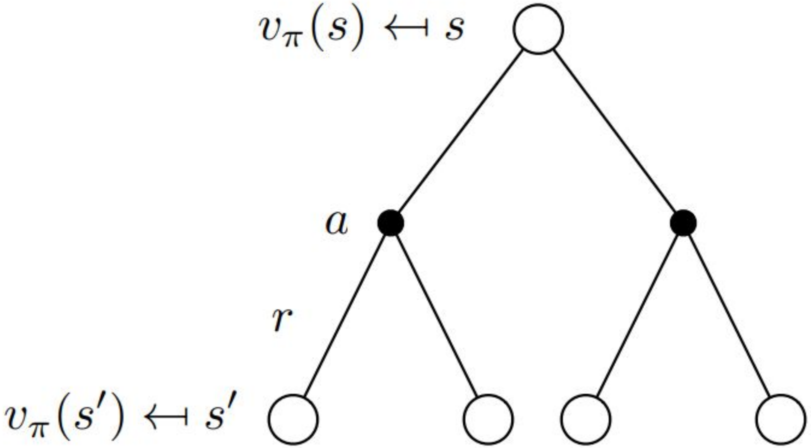
$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s,a)$$

Bellman Expectation Equation [1]



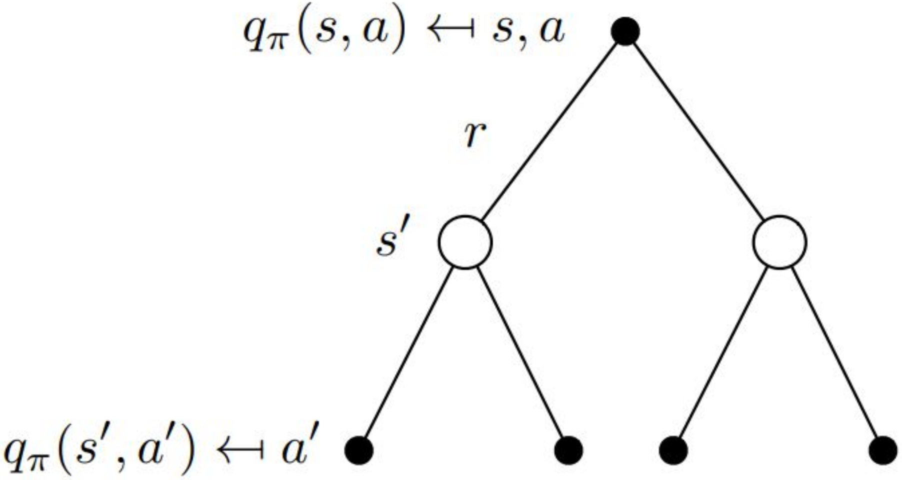
$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s')$$

Bellman Expectation Equation [1]



$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s') \right)$$

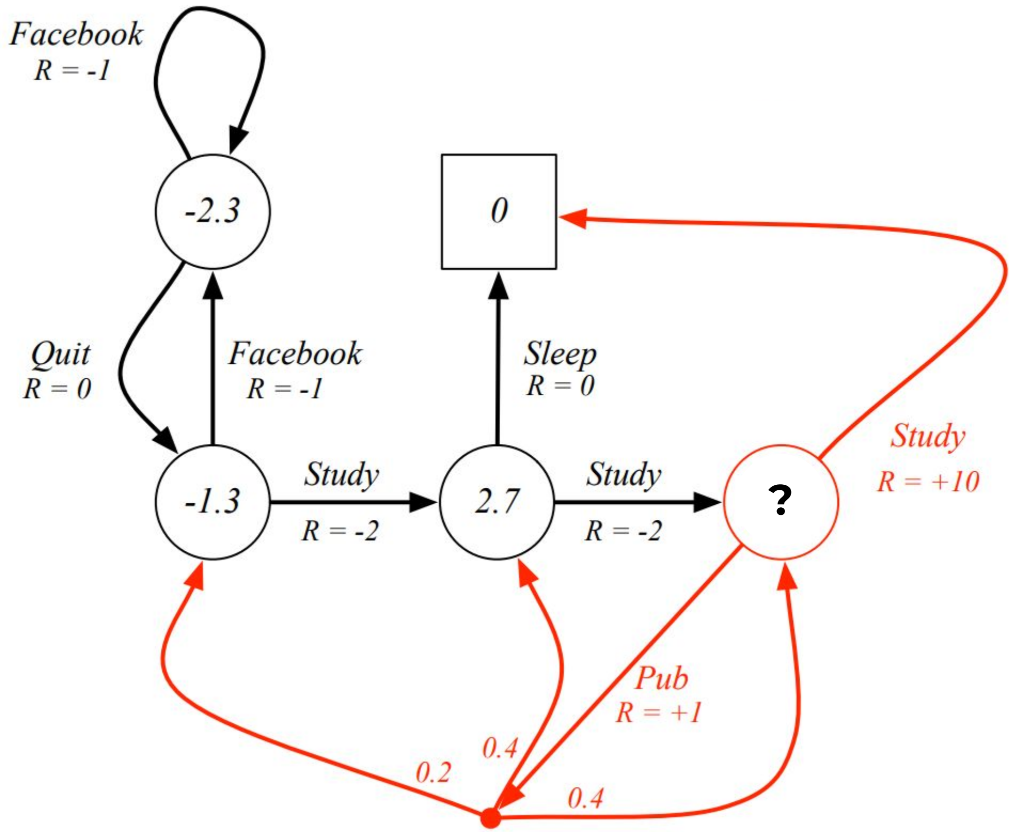
Bellman Expectation Equation [1]



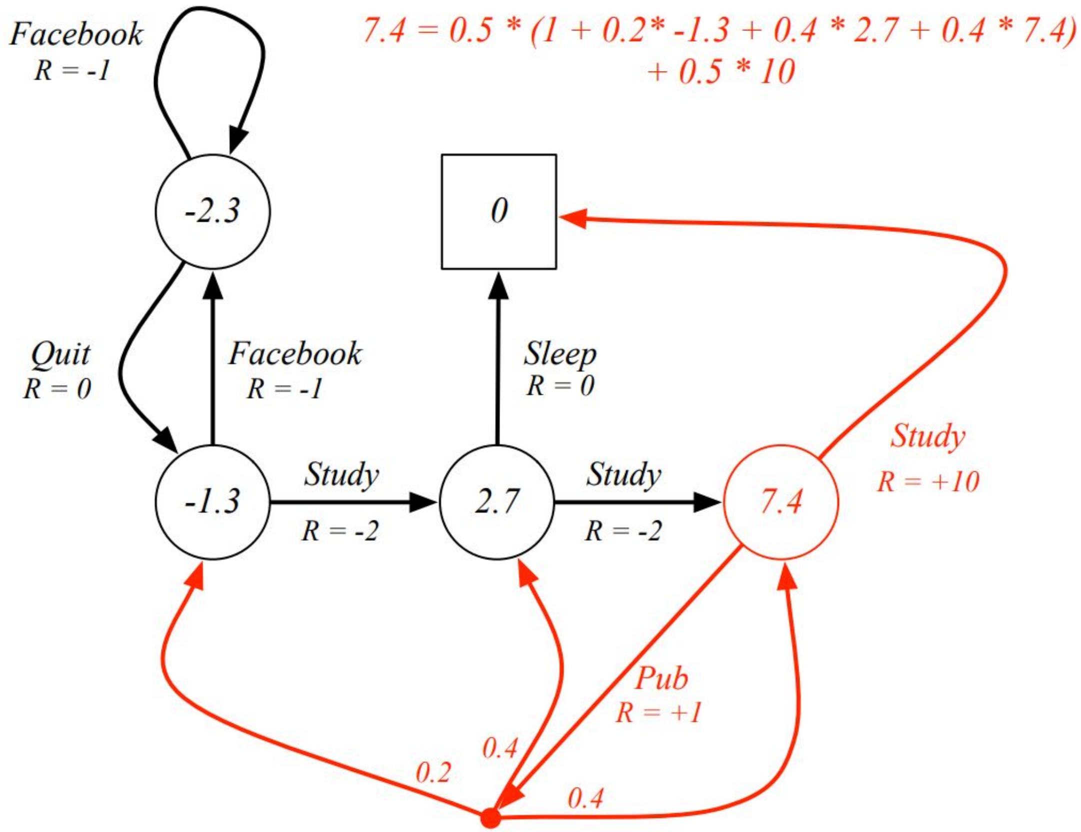
$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \sum_{a' \in \mathcal{A}} \pi(a'|s') q_\pi(s',a')$$

Exercise

Example: Student MDP



Example: Student MDP



Bellman Optimality

State-Value and Action-Value Functions

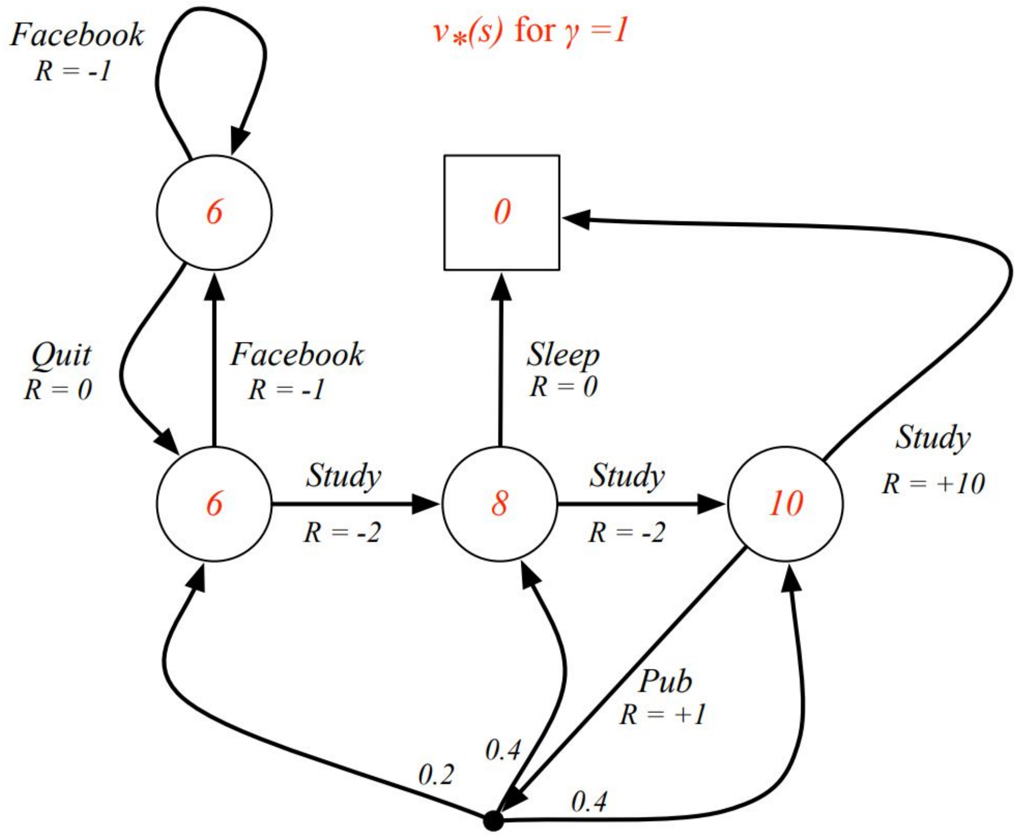
- The optimal state-value function

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

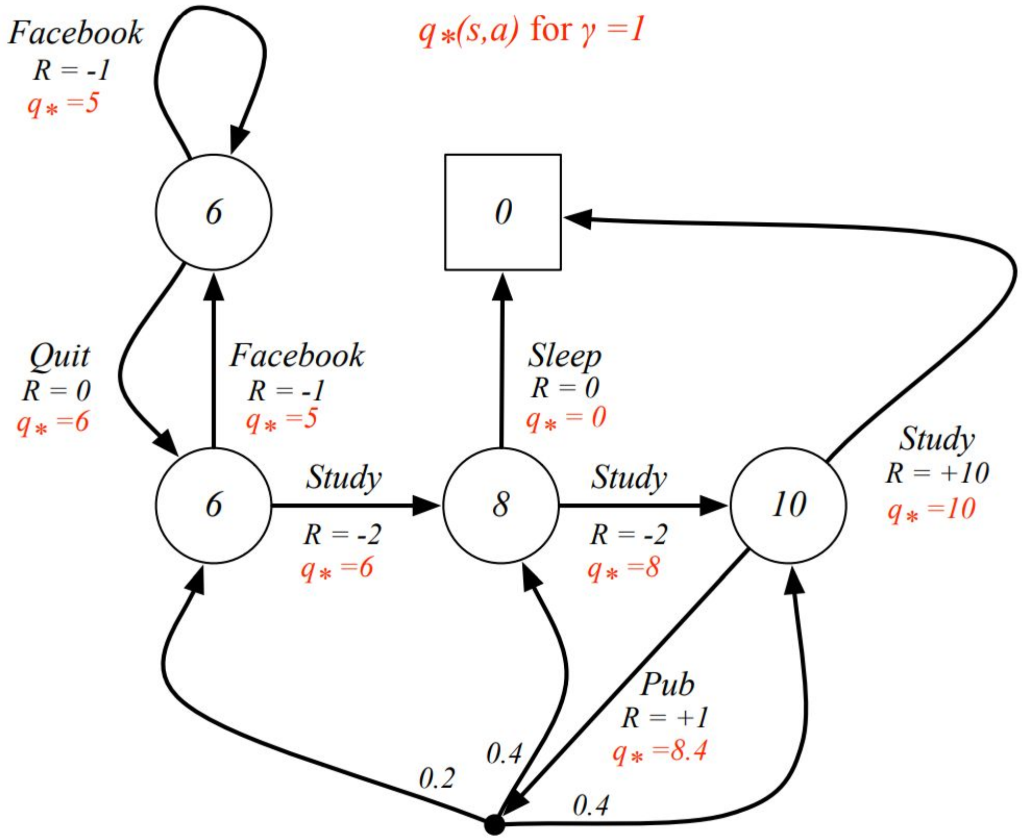
- Optimal action-value function

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

Exercise: Optimal State-Value Function [1]



Exercise: Optimal Action-Value Function [1]



Find an Optimal Policy

- An optimal policy ***π**** can be determined by selecting actions that maximize the optimal action-value function ***q*(s,a)***. The optimal policy ***π*** ***(a/s)*** is defined as:

$$\pi^*(a|s) = \begin{cases} 1 & \text{if } a = \arg\max_{a \in A} q^*(s,a) \\ 0 & \text{otherwise} \end{cases}$$

- For any MDP, there is always a deterministic optimal policy. If ***q*(s,a)*** is known, we can directly derive the optimal policy from it.

Exercises

Exercise 1: Understanding Policies

Question:

Let ***S={s1,s2}*** be a set of two states and ***A={a1,a2}*** be a set of two actions. Suppose a stochastic policy ***π*** is defined as follows:

$$\pi(a_1|s_1) = 0.7, \pi(a_2|s_1) = 0.3$$
$$\pi(a_1|s_2) = 0.4, \pi(a_2|s_2) = 0.6$$

1. What is the probability of taking action ***a2*** in state ***s1*** under this policy?
2. If the agent is in state ***s2***, what is the probability of taking action ***a1*** under this policy?

Exercise 1: Understanding Policies

Solution:

1. The probability of taking action ***a2*** in state ***s1*** is given directly by $\boldsymbol{\pi(a2|s1)=0.3}$
2. The probability of taking action ***a1*** in state ***s2*** is given by $\boldsymbol{\pi(a1|s2)=0.4}$

Exercise 2: State-Value Function

Question:

Consider a simple MDP with two states ***s1*** and ***s2*** and a single action ***a*** with the following reward structure:

- Starting from ***s1*** and taking action ***a***, the agent moves to ***s2*** with a **reward of 5**.
- Starting from ***s2*** and taking action ***a***, the agent stays in ***s2*** and receives a **reward of 3**.

Assuming a discount factor ***γ=0.9*** and a **deterministic policy** where action ***a*** is always taken, compute the value of each state ***v(s1)*** and ***v(s2)***.

Exercise 2: State-Value Function

Solution:

The Bellman equation for the value of each state ***s*** is:

$$v(s) = R(s,a) + \sum_{s'} P(s'|s,a)v(s')$$

1. For ***s2***:

$$v(s_2) = 3 + \gamma v(s_2) \Rightarrow v(s_2) = 3 + 0.9v(s_2)$$

   Solving for ***v(s2)*** → ***v(s2)***=30

2. For ***s1***:

$$v(s_1) = 5 + \gamma v(s_2) = 5 + 0.9 \times 30 = 5 + 27 = 32$$

Thus, ***v(s1)***=32 and ***v(s2)***=30.

Exercise 3: Action-Value Function

Question:

Using the same MDP setup as in Exercise 2, calculate the action-value ***q(s1,a)*** and ***q(s2,a)*** for each state-action pair.

Exercise 3: Action-Value Function

Solution:

The Bellman equation for the action-value function is:

$$q(s, a) = R(s, a) + \gamma \sum_{s'} P(s'|s, a)v(s')$$

Using the state values calculated in Exercise 2:

$$q(s_1, a) = 5+\gamma v(s_2) = 5+0.9\times 30 = 5+27 = 32$$
$$q(s_2, a) = 3+\gamma v(s_2) = 3+0.9\times 30 = 3+27 = 30$$

Exercise 4: Bellman Optimality Equation

Question:

Suppose we have an MDP with three states ***S={s1,s2,s3}*** and two actions ***A={a1,a2}***.
The reward function and transitions are given below:

- From ***s1*** taking ***a1*** leads to ***s2*** with **reward 4**.
- From ***s1*** taking ***a2*** leads to ***s3*** with **reward 2**.
- From ***s2*** taking ***a1*** leads to ***s3*** with **reward 5**.
- From ***s3*** taking ***a1*** or ***a2*** leads back to ***s3*** with **reward 3**.

Assuming a discount factor ***γ=0.9***, write the Bellman optimality equation for ***v*(s1)***.

Exercise 4: Bellman Optimality Equation

Solution:

The Bellman optimality equation for the state-value function is:

$$v^*(s) = \max_a \sum_{s'} P(s'|s,a)\left[R(s,a,s') + \gamma v^*(s')\right]$$

$$v^*(s_1) = \max\left(R(s_1,a_1,s_2) + \gamma v^*(s_2),\ R(s_1,a_2,s_3) + \gamma v^*(s_3)\right)$$

Substituting the rewards:

$$v^*(s_1) = \max\left(4 + 0.9v^*(s_2),\ 2 + 0.9v^*(s_3)\right)$$

To solve this, we would need the values of ***v*(s2)*** and ***v*(s3)***, which can be calculated recursively by applying the Bellman optimality equation to each state.

Exercise 4: Bellman Optimality Equation

Solution:

$$v^*(s_2) = 5 + 0.9 \cdot v^*(s_3)$$
$$v^*(s_3) = 3 + 0.9 \cdot v^*(s_3)$$

The optimal values for each state are:

- ***v*(s1) = 32.8***
- ***v*(s2) = 32***
- ***v*(s3) = 30***

Exercise 5: Optimal Policy Derivation

Question:

If the optimal action-value function ***q*(s,a)*** for some state ***s*** is given by:

- ***q*(s,a1)=12***
- ***q*(s,a2)=10***

What is the optimal policy ***π*(a|s)***?

Exercise 5: Optimal Policy Derivation

Solution:

The optimal policy ***π*(a|s)*** chooses the action that maximizes ***q*(s,a)***.

So:

$$\pi^*(a|s) = \begin{cases} 1 & \text{if } a = a_1 \\ 0 & \text{if } a = a_2 \end{cases}$$

Thus, the optimal policy is to always choose action ***a1*** in state ***s***, since ***q*(s,a1)>q*(s,a2)***.

Exploration & Exploitation

Exploration vs. Exploitation

- In RL, the agent faces a **dilemma** between:
  - **Exploration:** Trying **new actions** to discover valuable outcomes. (can be harmful...)
  - **Exploitation:** Choosing **actions** that have yielded **high rewards** in the past.

- **Goal:** Balance exploration and exploitation to maximize rewards over time.
- **Challenge:** Too much exploration can delay achieving rewards, while too much exploitation can lead to suboptimal long-term results.

Exploration: Discovering New Opportunities

- **Example 1 - A robot navigating a maze:**
  - The robot tries unfamiliar paths to locate shorter routes or more valuable rewards.
- **Example 2 - A recommendation system:**
  - Occasionally recommends new, lesser-known products to a user to learn their interests.
- **Benefit:** Exploration can uncover higher rewards that aren't immediately obvious.

Exploitation: Leveraging Known Information

- **Example 1 - A trading agent:**
    - Selects stocks it has previously identified as profitable, prioritizing consistency over discovering new options.
- **Example 2 - A game-playing AI:**
    - Repeats a high-reward move (e.g., a chess opening) that has led to victories in past games.
- **Benefit:** Exploitation capitalizes on known successes, ensuring steady rewards.

Any Questions ?

Don’t hesitate to contact me

morand@isir.pmc.fr

Any Questions ?

Contact us !

hussam.atoui@valeo.com

References

[1] David Silver, Lectures on Reinforcement Learning, 2015

[2] Reinforcement Learning and Advanced Deep Learning (Sorbonne) - Olivier Sigaud

[3] Sutton, R. S. and Barto, A. G. (2018), Reinforcement Learning: An Introduction (Second edition). MIT Press

Olivier Sigaud Youtube Channel