Introduction to Reinforcement Learning

Mattia Pellegrino, Ph.D Fellow
mattia.pellegrino@unipr.it

Summary

- Information about the course
- About Reinforcement Learning
- The Reinforcement Learning challenges
- RL Agent
- Reinforcement Learning Problems

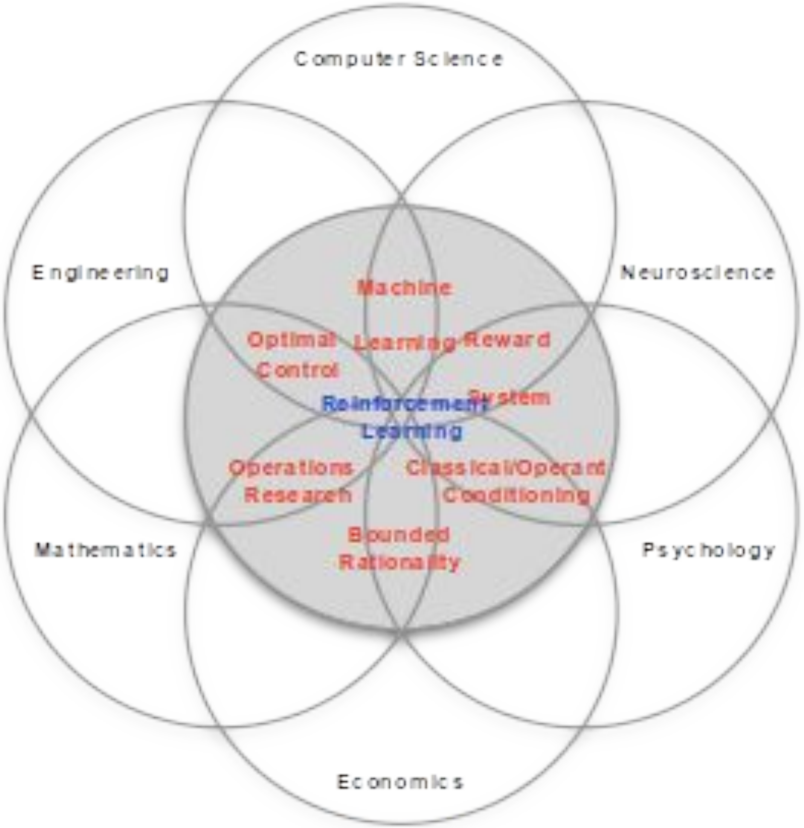
COURSE INFORMATION

- Contact me: **mattia.pellegrino@unipr.it**
- **TextBooks**:
    - Montague, P. Read. "Reinforcement learning: an introduction, by Sutton, RS and Barto, AG." Trends in cognitive sciences 3.9 (1999): 360. http://webdocs.cs.ualberta.ca/~sutton/book/the-book.html
    - Appress, “**Deep Reinforcement Learning with python**, by Nimish Sanghi”, (2021)

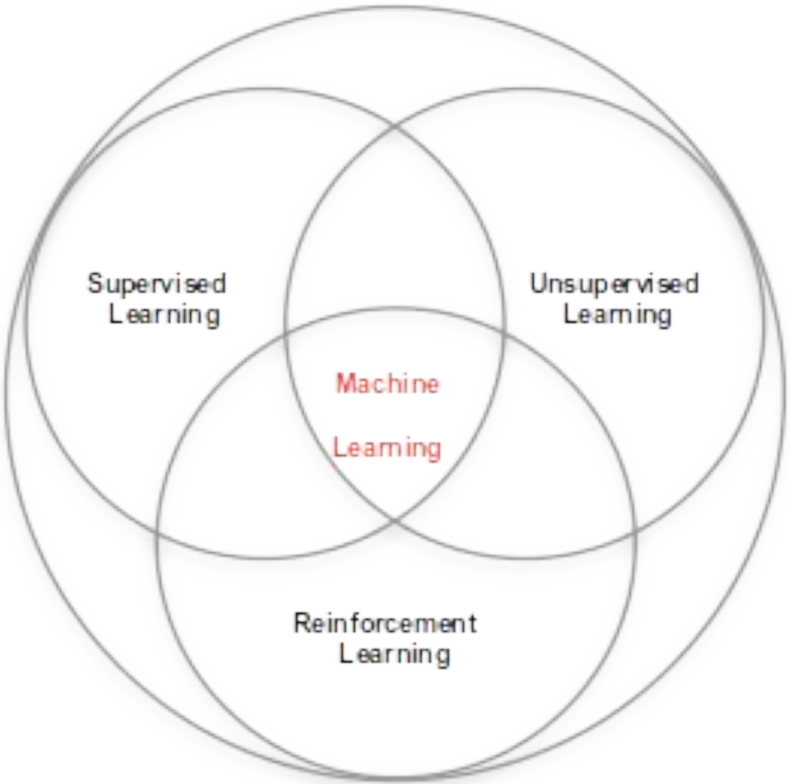
RL PLACEMENT

- All these branches try to solve the same problem: “**Decision Making**”

- Find the best actions’ combination to get the best result



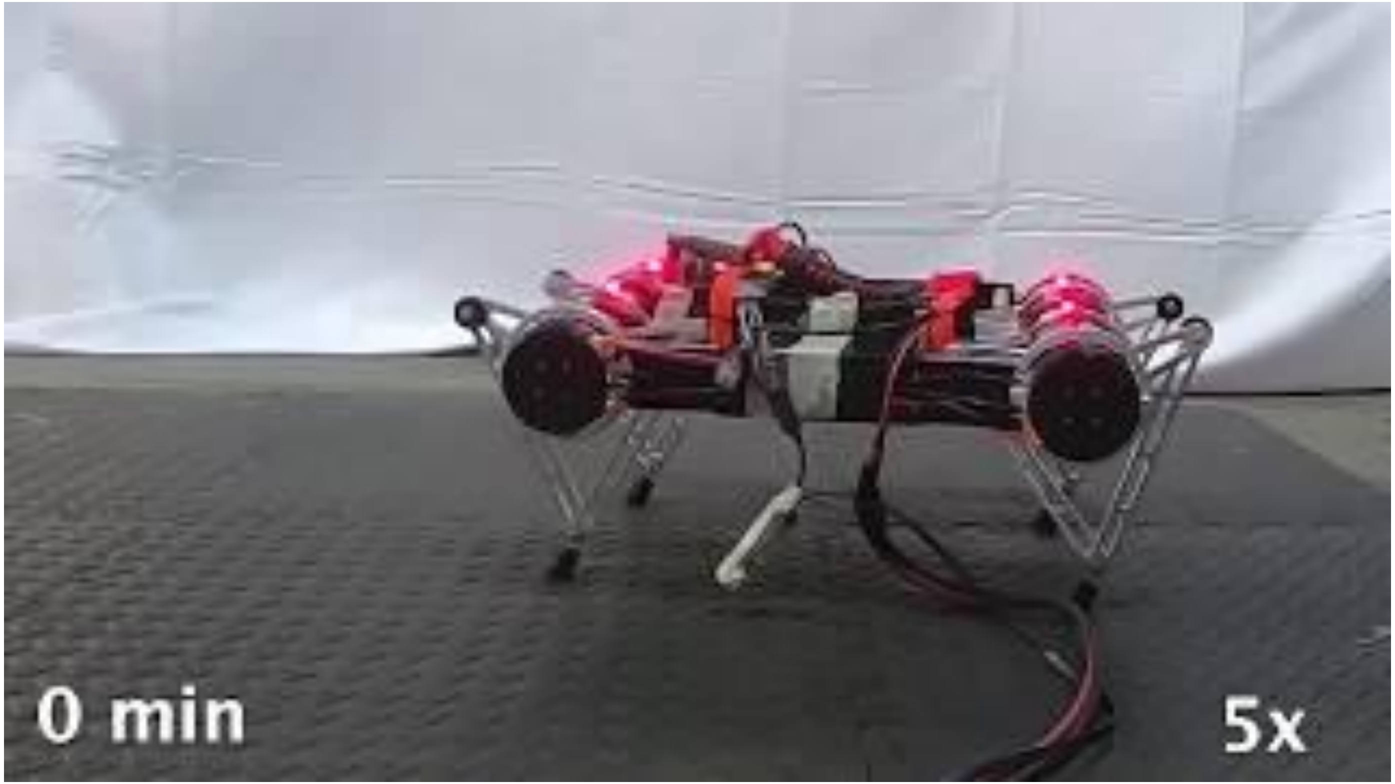
MACHINE LEARNING BRANCHES



RL: PROPERTIES

- What makes RL different from other learning methodologies?
  - There is no supervisor. Instead, there is a *reward signal*
    - *How do we know if what we are doing is right or not?*
  - Reward is not instantaneous
    - *We realize that we have done a bad choice afterwards*
  - Time matters
  - Agent’s actions affect the entire environment

RL: A ROBOT LEARNS TO WALK



RL: TRACKMANIA

RL: BOXING

RL: ATARI GAME

RL: REWARD

- A reward $\mathbf{R_t}$ is a scalar feedback signal
    - It is just a number
- Indicates how good is the action choose by the agent at **timestep *t***
- The agent’s goal is to maximize the total reward

RL is based on the ***reward hypothesis***

“All goals can be described by the maximization of expected cumulative reward”

RL: EXAMPLE OF REWARDS

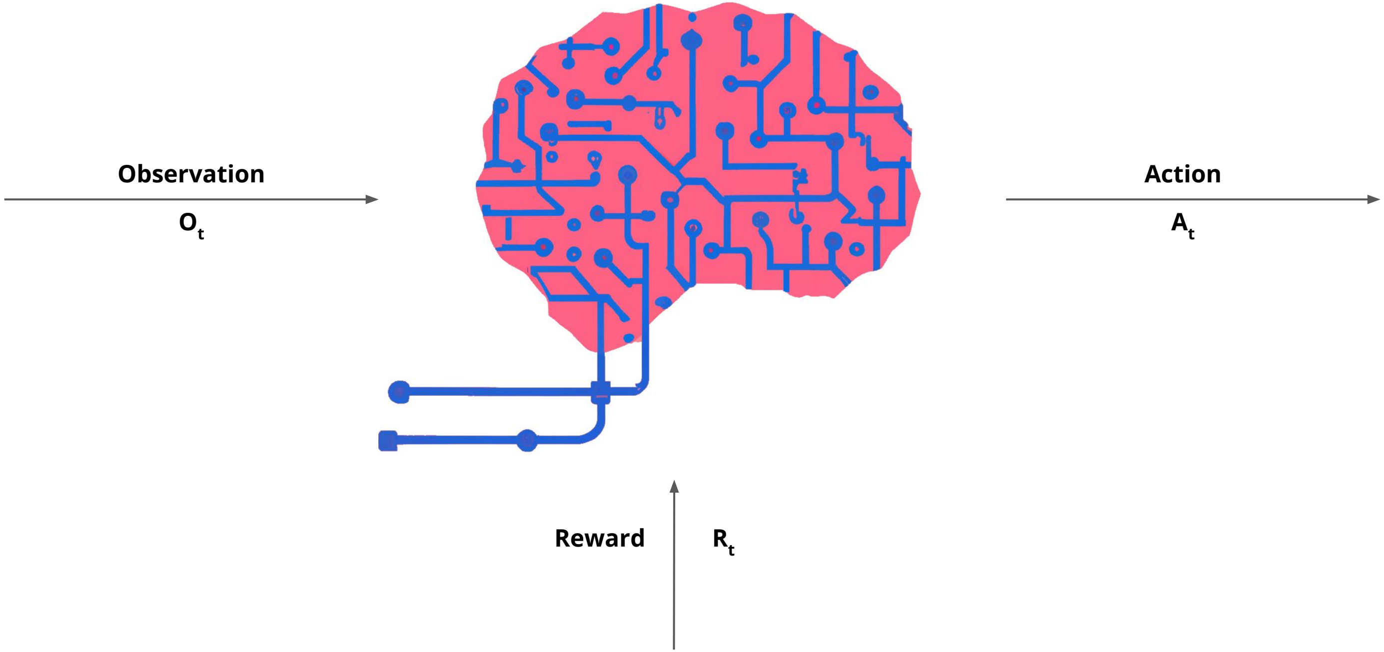
- **A robot learns to walk**
  - + reward for forward motion
  - - reward for falling over
- **Trackmania**
  - + reward for forward motion
  - - reward to falling over
- **Boxing**
  - + reward to stand stand in place
  - - reward to falling over
  - + reward to hit the enemy
  - - reward to get hit
- **Play Atari Games**
  - +/- reward for increasing/decreasing score

RL: SEQUENTIAL DECISION MAKING

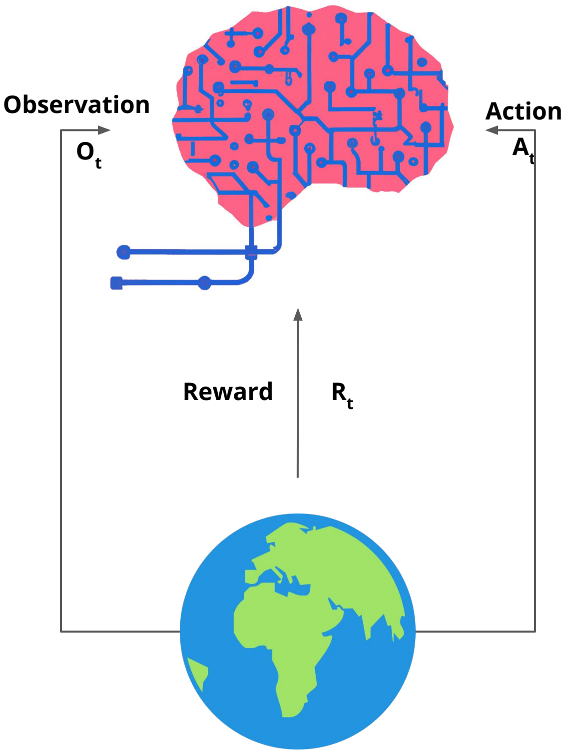
- **Goal**: select actions to maximize total future reward
- Actions may have long term consequences
- Reward may be **delayed**
- It may be better to **sacrifice immediate reward** to gain more **long-term reward**
- Examples:
  - A financial investment (may take months to mature)
  - Blocking opponent moves (might help winning chances many moves from now)

AGENT AND ENVIRONMENT

- We will use the current formalism



AGENT AND ENVIRONMENT



- At each step t the agent:
  - Execute an action A_t
  - Obtain an observation O_t
  - Obtain a reward R_t
- The environment
  - Receives an action A_t
  - Releases an observation O_{t+1}
  - Releases a Reward R_{t+1}

HISTORY - STATE

History is the sequence of observation, action, and rewards (usually huge)

$$H_t = O_1, R_1, A_1, \ldots, A_t, O_t, R_t$$

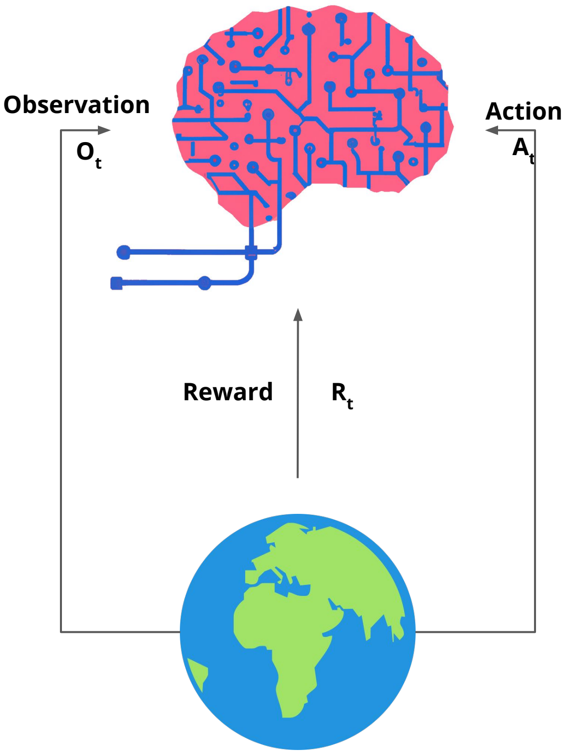
- An algorithm is a **mapping** between the **history** and what **happens next**:
  - The agent selects actions
  - The environment selects observations/rewards

The state is the information used to determine what happens next

- It's a **synthesis** of what happened and we base on this because the history usually is too big to compute
- Formally, state is a function of the history

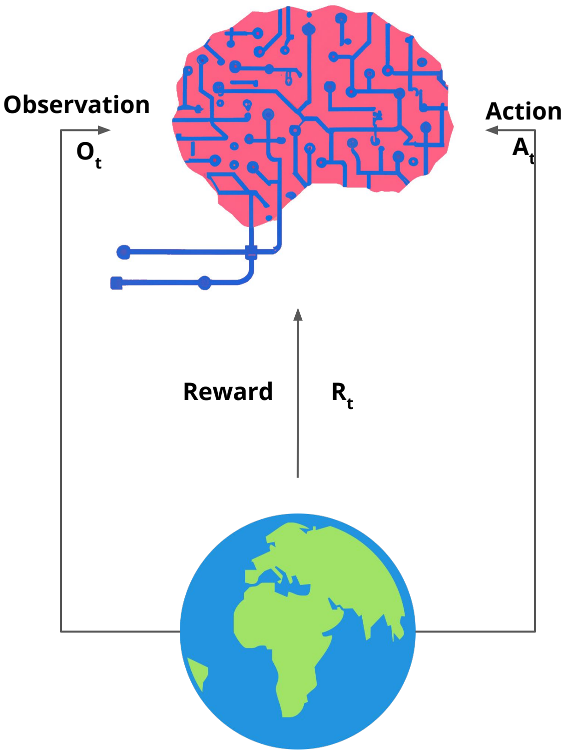
$$S_t = f(H_t)$$

ENVIRONMENT STATE



- The ***environment state*** $\boldsymbol{S_t^e}$ is the environment's private representation
- This state is **not** usually **visible** to the agent
- Sometimes contains **irrelevant** information

AGENT STATE



- The ***agent state*** $\mathbf{S_t^a}$ is the agent's internal representation
- Whatever information the agent can use to choose the next action
- It can be any function of history:

$$S_t = f(H_t)$$

INFORMATION STATE

An information state (a.k.a. Markov state) contains all useful information from the history

- ***A state S_{t} is a Markov state if and only if***

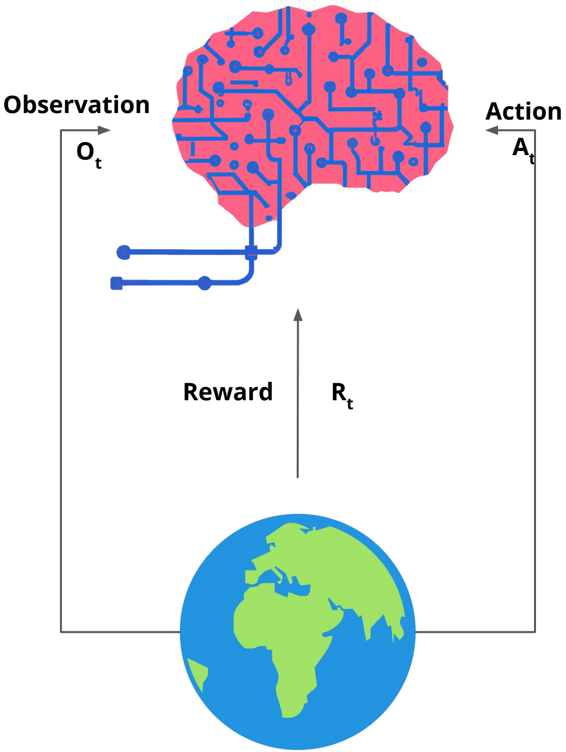
$$\mathbb{P}\left[S_{t+1} \mid S_t\right] = \mathbb{P}\left[S_{t+1} \mid S_1, \ldots, S_t\right]$$

- The **future** is **independent** of the **past** given present

$$H_{1:t} \longrightarrow S_t \longrightarrow H_{t+1:\infty}$$

- Once the **state** is **known**, the **history** is **irrelevant**
- The environment state S^e_t is a Markov State
- The history H_t is a Markov state

FULLY OBSERVABLE ENVIRONMENTS



- Full observability: agent directly observes environment state

$$O_t = S_t^a = S_t^e$$

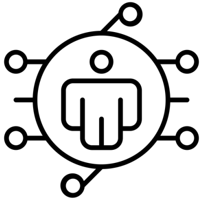
- Agent state = environment state = information state

- Formally, this is a Markov Decision Process (MDP)

PARTIALLY OBSERVABLE ENVIRONMENTS

- **Partial observability:** agent indirectly observes environment:
  - A robot with a camera vision
  - A card player agent
  - A trading agents
- In this case, the agent and the environment state are not the same
- This is a **partially observable Markov decision process (POMDP)**
- An agent must build its own state representation (S_a^t)

MAJOR RL AGENT COMPONENTS



Policy

Agent behavior function

Value Function

How good is each state and/or action

Model

Agent representation of the environment

POLICY

- A **policy** is the agent's behaviour
- It is a map from state of action
- There are 2 types of policies:
  - Deterministic policy: $a = \pi(s)$
  - Stochastic policy: $\pi(a \mid s) = \mathbb{P}\left[A_t = a \mid S_t = s\right]$

VALUE FUNCTION

- Value function is a prediction of future reward
- We use it to evaluate the goodness of the various states
- We can use it to pick the best action

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

MODEL

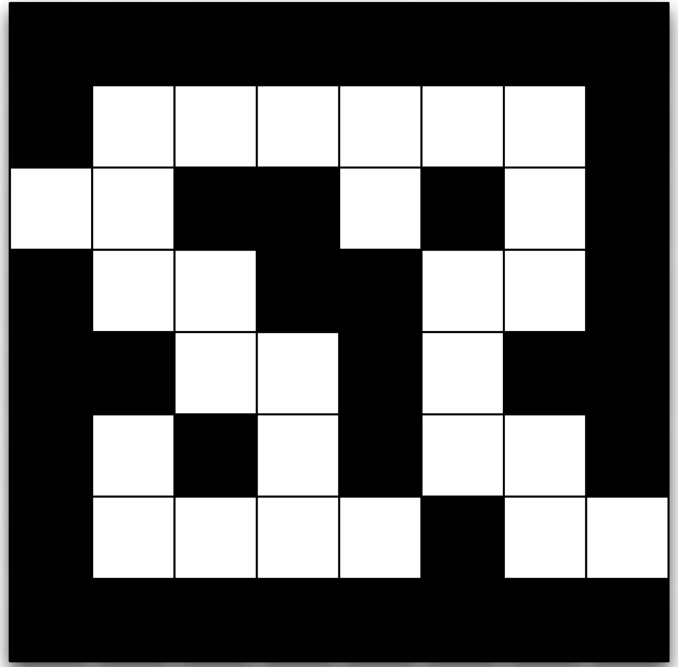
- A model predicts what the environment will do next
- P predicts the next state
- R predicts the next (immediate) rewards

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

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

MAZE EXAMPLE

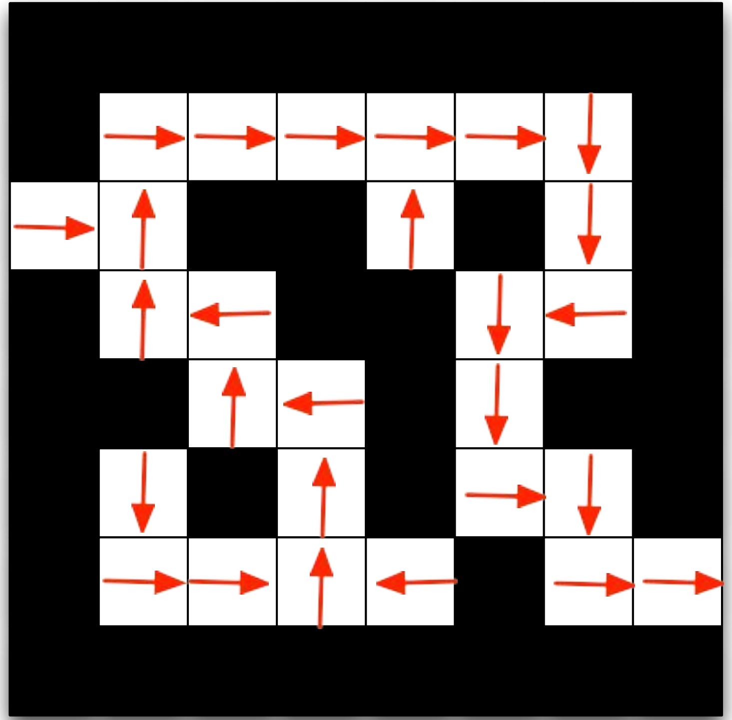
Start



Goal

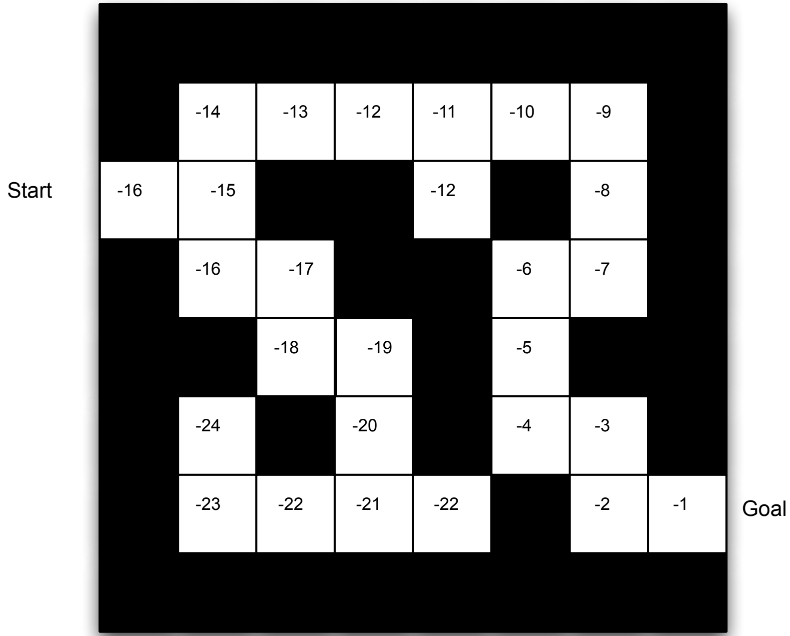
- Rewards: -1 per time-step
- Action: L,R,U,D
- States: Agent’s Location

MAZE EXAMPLE



- Arrows represent policy $\pi(s)$ for each state s

MAZE EXAMPLE

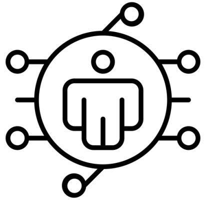


Numbers represent value $v_\pi(s)$ of each state s

EXPLORATION AND EXPLOITATION

- Reinforcement learning is like trial-and-error learning
- The agents should discover a good policy from its experiences of the environment
- Without losing too much reward along the way
- Exploration find more information about the environment
- Exploitation exploits known information to maximize reward

RL AGENT TAXONOMY



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