CNRS - Toulouse INP - UT3 - UT Capitole - UT2

Institut de Recherche en Informatique de Toulouse

Lecture 1: Introduction

N8EN18B - Contrôle et Apprentissage

Guilherme IECKER RICARDO

IRIT, Université de Toulouse, CNRS, Toulouse INP, UT3

Contents

Course Organization

Class Info

- Contact: guilherme.ricardo@irit.fr
- Moodle: Announcements, lecture slides, and additional material
- No official textbook – Recommended reading:
  - An Introduction to Reinforcement Learning, Sutton and Barto (2020).
  - Algorithms for Reinforcement Learning, Szepesvari (2010).
- Assessment:
  - 25% TP
  - 75% Final Exam
- Slides are based on Prof. Silver's course [Silver, 2015]

Schedule

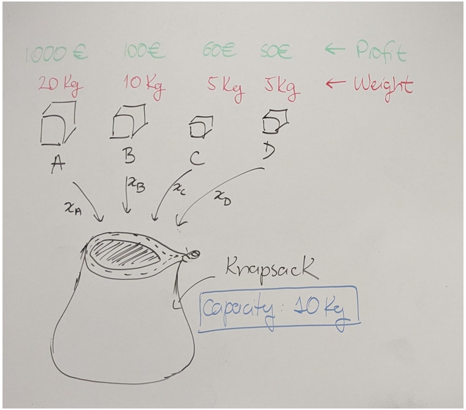
Class	Date	Time	Topic
CM1	29/03/2024	10h15 - 12h00	Introduction
CM2	02/04/2024	16h15 - 18h00	MDP
CM3	04/04/2024	10h15 - 12h00	Dynamic Programming
CM4	23/04/2024	10h15 - 12h00	Q-Learning
TP	26/04/2024	A: 08h00 - 09h45	T.B.A.
		R: 10h15 - 12h00	T.B.A.
CM5	14/05/2024	08h00 - 09h45	Deep Q-Learning
Final Exam			

Table: Tentative Schedule

What is this course about?

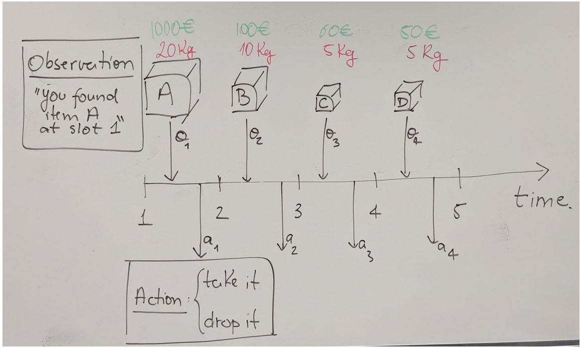
Control and Learning

Control = Optimization



Control and Learning

Learning = Sequential decision making under uncertainty



Reinforcement Learning

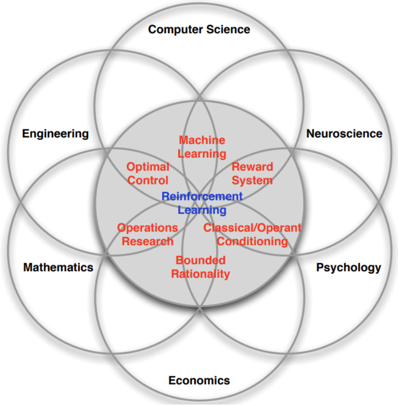


Figure: RL according to different areas.

Reinforcement Learning

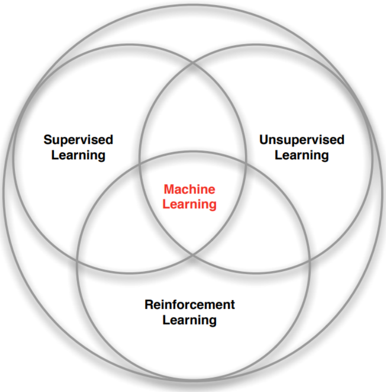


Figure: RL as a Machine Learning paradigm.

Characteristics of RL

What makes RL different from other ML paradigms?

- There is no supervisor, only a reward signal

Characteristics of RL

What makes RL different from other ML paradigms?

- There is no supervisor, only a reward signal
- Feedback is delayed, not instantaneous

Characteristics of RL

What makes RL different from other ML paradigms?

- There is no supervisor, only a reward signal
- Feedback is delayed, not instantaneous
- Time really matters (sequential, non-i.i.d. data)

Characteristics of RL

What makes RL different from other ML paradigms?

- There is no supervisor, only a reward signal
- Feedback is delayed, not instantaneous
- Time really matters (sequential, non-i.i.d. data)
- Agent’s actions affect the subsequent data it receives

Example of RL Applications

- Fly stunt manoeuvres in a helicopter

Example of RL Applications

- Fly stunt manoeuvres in a helicopter
- Defeat the world champion at Backgammon

Example of RL Applications

- Fly stunt manoeuvres in a helicopter
- Defeat the world champion at Backgammon
- Manage an investment portfolio

Example of RL Applications

- Fly stunt manoeuvres in a helicopter
- Defeat the world champion at Backgammon
- Manage an investment portfolio
- Control a power station

Example of RL Applications

- Fly stunt manoeuvres in a helicopter
- Defeat the world champion at Backgammon
- Manage an investment portfolio
- Control a power station
- Make a humanoid robot walk

Example of RL Applications

- Fly stunt manoeuvres in a helicopter
- Defeat the world champion at Backgammon
- Manage an investment portfolio
- Control a power station
- Make a humanoid robot walk
- Play many different video games better than humans

Example of RL Applications in Networking

5G Drone Relay [Gangula et al., 2018]

The RL Problem

The RL Problem

Reward

Rewards

- A **reward** R_t is a scalar feedback signal

Rewards

- A **reward** R_t is a scalar feedback signal
- Indicates how well agent is doing at step t

Rewards

- A **reward** R_t is a scalar feedback signal
- Indicates how well agent is doing at step t
- The agent's job is to maximize cumulative reward

Rewards

- A **reward** R_t is a scalar feedback signal
- Indicates how well agent is doing at step t
- The agent's job is to maximize cumulative reward
- RL is based on the **reward hypothesis**:

Definition (Reward Hypothesis)

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

Rewards

- A **reward** R_t is a scalar feedback signal
- Indicates how well agent is doing at step t
- The agent's job is to maximize cumulative reward
- RL is based on the **reward hypothesis**:

Definition (Reward Hypothesis)

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

Do you agree with this statement?

Examples of Rewards

- Fly stunt manoeuvres in a helicopter
  - +ve reward for following desired trajectory
  - -ve reward for crashing

Examples of Rewards

- Fly stunt manoeuvres in a helicopter
  - +ve reward for following desired trajectory
  - -ve reward for crashing
- Manage an investment portfolio
  - +ve/-ve reward for each $ in bank

Examples of Rewards

- Fly stunt manoeuvres in a helicopter
  - +ve reward for following desired trajectory
  - -ve reward for crashing
- Manage an investment portfolio
  - +ve/-ve reward for each $ in bank
- Control a power station
  - +ve reward for producing power
  - -ve reward for exceeding safety thresholds

Examples of Rewards

- Fly stunt manoeuvres in a helicopter
  - +ve reward for following desired trajectory
  - -ve reward for crashing
- Manage an investment portfolio
  - +ve/-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

Examples of Rewards

- Fly stunt manoeuvres in a helicopter
  - +ve reward for following desired trajectory
  - -ve reward for crashing
- Manage an investment portfolio
  - +ve/-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/-ve reward for increasing/decreasing score

Examples of Rewards

What are the rewards for the 5G relay example?

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)
  - Refuelling a helicopter (might prevent a crash in several hours)
  - Blocking opponent moves (might help winning chances many moves from now)

The RL Problem

Agent and Environment

Agent

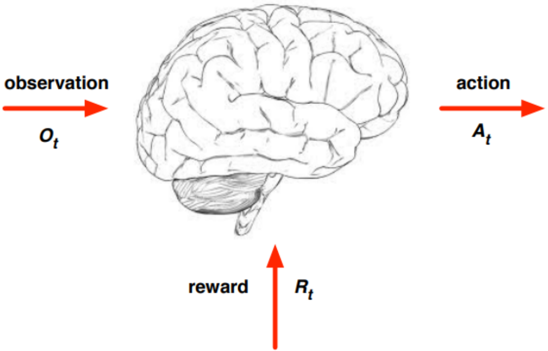


Figure: Agent Abstraction

Agent and Environment

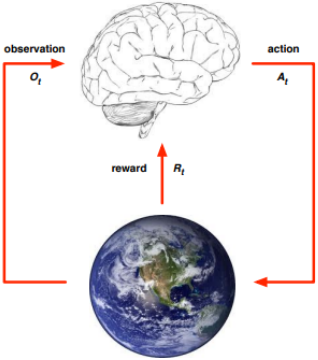


Figure: Agent-Environment cycle

At each step t,

- the agent:
  - Executes action A_t
  - Receives observation O_t
  - Receives scalar reward R_t
- the environment:
  - Receives action A_t
  - Emits observation O_{t+1}
  - Emits scalar reward R_{t+1}
- t increments at environment step

The RL Problem

State

History and State

- The *history* is the sequence of observations, actions, rewards:

$$H_t = O_1, R_1, A_1, \ldots, A_{t-1}, O_t, R_t \tag{1}$$

History and State

- The *history* is the sequence of observations, actions, rewards:

$$H_t = O_1, R_1, A_1, \ldots, A_{t-1}, O_t, R_t \tag{1}$$

- i.e., all observable variables up to time t

History and State

- The *history* is the sequence of observations, actions, rewards:

$$H_t = O_1, R_1, A_1, \ldots, A_{t-1}, O_t, R_t \tag{1}$$

- i.e., all observable variables up to time t
- What happens next depends on the history:
  - The agent selects actions
  - The environment selects observation/rewards

History and State

- The *history* is the sequence of observations, actions, rewards:

$$H_t = O_1, R_1, A_1, \ldots, A_{t-1}, O_t, R_t \tag{1}$$

- i.e., all observable variables up to time t
- What happens next depends on the history:
  - The agent selects actions
  - The environment selects observation/rewards
- *State* is the information used to determine what happens next

History and State

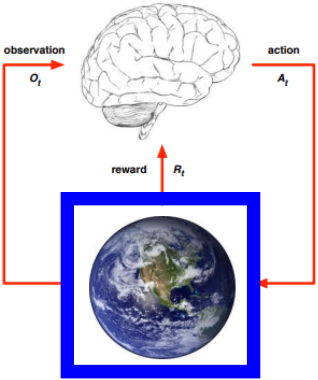
- The *history* is the sequence of observations, actions, rewards:

$$H_t = O_1, R_1, A_1, \ldots, A_{t-1}, O_t, R_t \tag{1}$$

- i.e., all observable variables up to time t
- What happens next depends on the history:
  - The agent selects actions
  - The environment selects observation/rewards
- *State* is the information used to determine what happens next
- Formally, the state S_t at time t is a function of the history:

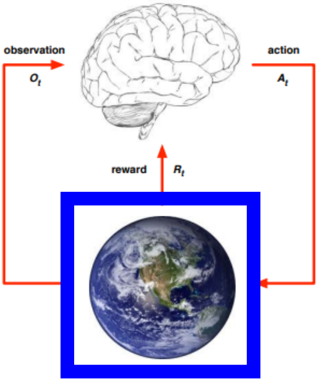
$$S_t = f(H_t) \tag{2}$$

Environment State



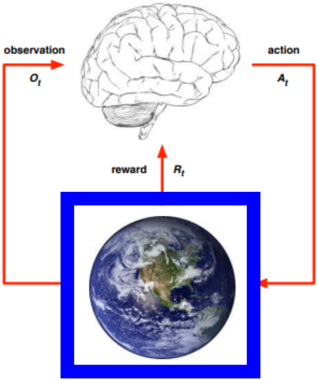
- The **environment state** S_t^e is the environment's private representation

Environment State



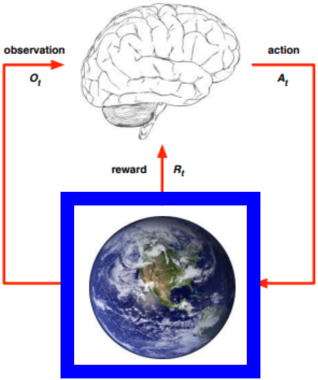
- The **environment state** S_t^e is the environment's private representation
- i.e., whatever data the environment uses to pick the next observation/reward

Environment State



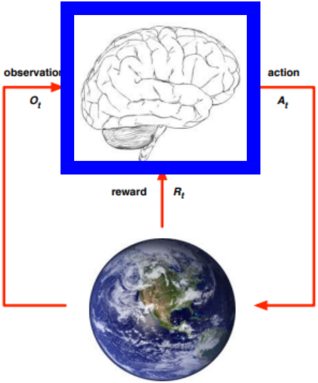
- The **environment state** S_t^e is the environment's private representation
- i.e., whatever data the environment uses to pick the next observation/reward
- The environment state is not usually visible to the agent

Environment State



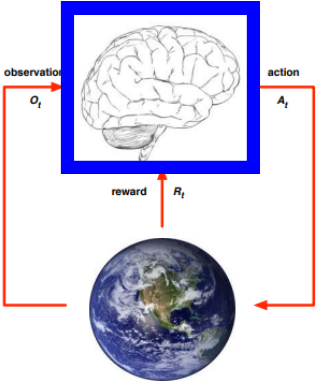
- The **environment state** S_t^e is the environment's private representation
- i.e., whatever data the environment uses to pick the next observation/reward
- The environment state is not usually visible to the agent
- Even if S_t^e is visible, it may contain irrelevant information

Agent State



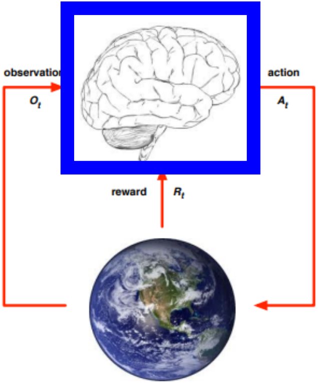
- The **agent state** S_t^a is the agent's internal representation

Agent State



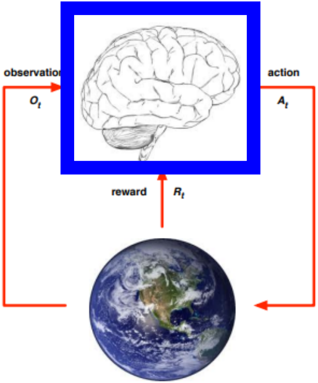
- The **agent state** S_t^a is the agent's internal representation
- i.e., whatever information the agent uses to pick the next action

Agent State



- The **agent state** S_t^a is the agent's internal representation
- i.e., whatever information the agent uses to pick the next action
- i.e., it is the information used by RL algorithms

Agent State



- The **agent state** S_t^a is the agent's internal representation
- i.e., whatever information the agent uses to pick the next action
- i.e., it is the information used by RL algorithms
- It can be any function of history:

$$S_t^a = f(H_t) \tag{3}$$

Information State

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

Information State

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

Definition

A state S_t is **Markovian** if, and only if,

$$\mathbb{P}(S_{t+1}|S_t) = \mathbb{P}(S_{t+1}|S_1, \ldots, S_t) \tag{4}$$

Information State

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

Definition

A state S_t is **Markovian** if, and only if,

$$\mathbb{P}(S_{t+1}|S_t) = \mathbb{P}(S_{t+1}|S_1, \ldots, S_t) \tag{4}$$

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

$$H_{1:t} \rightarrow S_t \rightarrow H_{t+1:\infty} \tag{5}$$

Information State

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

Definition

A state S_t is **Markovian** if, and only if,

$$\mathbb{P}(S_{t+1}|S_t) = \mathbb{P}(S_{t+1}|S_1, \ldots, S_t) \tag{4}$$

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

$$H_{1:t} \rightarrow S_t \rightarrow H_{t+1:\infty} \tag{5}$$

- Once the state is known, the history may be thrown away

Information State

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

Definition

A state S_t is **Markovian** if, and only if,

$$\mathbb{P}(S_{t+1}|S_t) = \mathbb{P}(S_{t+1}|S_1, \ldots, S_t) \tag{4}$$

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

$$H_{1:t} \rightarrow S_t \rightarrow H_{t+1:\infty} \tag{5}$$

- Once the state is known, the history may be thrown away
- i.e., The state is a sufficient statistic of the future

Rat Example

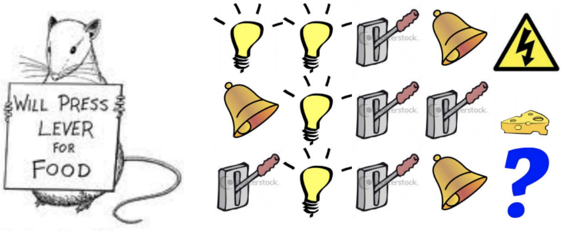


Figure: Cognitive Experiment with Rats

Rat Example

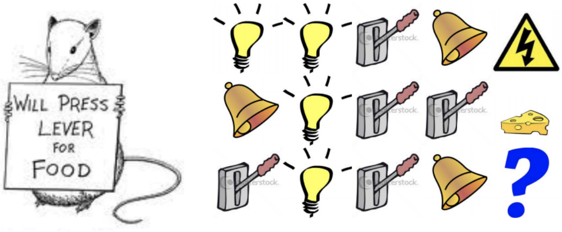


Figure: Cognitive Experiment with Rats

- What if agent state = "last 3 items in sequence"?

Rat Example

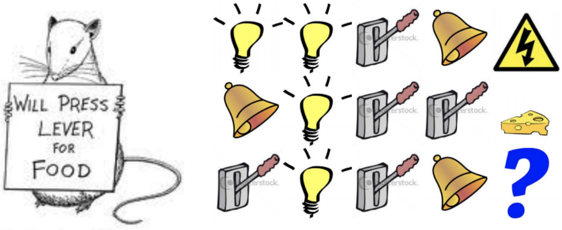


Figure: Cognitive Experiment with Rats

- What if agent state = "last 3 items in sequence"?
- What if agent state = "counts for lights, bells, or levers"?

Rat Example

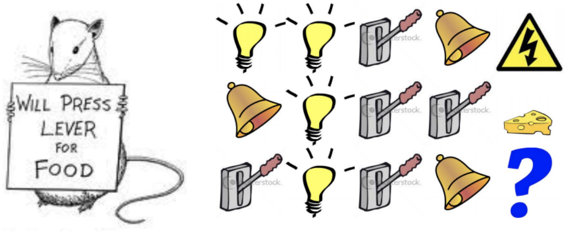


Figure: Cognitive Experiment with Rats

- What if agent state = "last 3 items in sequence"?
- What if agent state = "counts for lights, bells, or levers"?
- What if agent state = "complete sequence"?

Fully Observable Environments

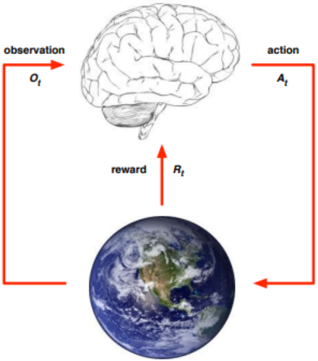


Figure: Agent-Environment cycle

Full observability: agent directly observes environment state

$$O_t = S_t^a = S_t^e \tag{6}$$

- Agent state = environment state = information state
- Formally, this is a **Markov Decision Process (MDP)**

Partially Observable Environments

- **Partial observability**: agent **indirectly** observes environment:

Partially Observable Environments

- **Partial observability**: agent **indirectly** observes environment:
  - A robot with a camera vision isn't told its absolute location

Partially Observable Environments

- **Partial observability**: agent **indirectly** observes environment:
  - A robot with a camera vision isn't told its absolute location
  - A trading agent only observes current prices

Partially Observable Environments

- **Partial observability**: agent **indirectly** observes environment:
  - A robot with a camera vision isn't told its absolute location
  - A trading agent only observes current prices
  - A poker playing agent only observes public cards

Partially Observable Environments

- **Partial observability**: agent **indirectly** observes environment:
  - A robot with a camera vision isn't told its absolute location
  - A trading agent only observes current prices
  - A poker playing agent only observes public cards
- Now, agent state $\neq$ environment state

Partially Observable Environments

- **Partial observability**: agent **indirectly** observes environment:
  - A robot with a camera vision isn't told its absolute location
  - A trading agent only observes current prices
  - A poker playing agent only observes public cards
- Now, agent state $\neq$ environment state
- Formally this is a **Partially Observable MDP (POMDP)**

Partially Observable Environments

- **Partial observability**: agent **indirectly** observes environment:
  - A robot with a camera vision isn't told its absolute location
  - A trading agent only observes current prices
  - A poker playing agent only observes public cards
- Now, agent state $\neq$ environment state
- Formally this is a **Partially Observable MDP (POMDP)**
- Agent must construct its own state representation S_t^a, e.g.,

Partially Observable Environments

- **Partial observability**: agent **indirectly** observes environment:
  - A robot with a camera vision isn't told its absolute location
  - A trading agent only observes current prices
  - A poker playing agent only observes public cards
- Now, agent state $\neq$ environment state
- Formally this is a **Partially Observable MDP (POMDP)**
- Agent must construct its own state representation S_t^a, e.g.,
  - Complete history: $S_t^a = H_t$

Partially Observable Environments

- **Partial observability**: agent **indirectly** observes environment:
  - A robot with a camera vision isn't told its absolute location
  - A trading agent only observes current prices
  - A poker playing agent only observes public cards
- Now, agent state $\neq$ environment state
- Formally this is a **Partially Observable MDP (POMDP)**
- Agent must construct its own state representation S_t^a, e.g.,
  - Complete history: $S_t^a = H_t$
  - **Beliefs** of environment state: $S_t^a = (\mathbb{P}(S_t^e = s^1), \ldots, \mathbb{P}(S_t^e = s^n))$

Partially Observable Environments

- **Partial observability**: agent **indirectly** observes environment:
  - A robot with a camera vision isn't told its absolute location
  - A trading agent only observes current prices
  - A poker playing agent only observes public cards
- Now, agent state $\neq$ environment state
- Formally this is a **Partially Observable MDP (POMDP)**
- Agent must construct its own state representation S_t^a, e.g.,
  - Complete history: $S_t^a = H_t$
  - **Beliefs** of environment state: $S_t^a = (\mathbb{P}(S_t^e = s^1), \ldots, \mathbb{P}(S_t^e = s^n))$
  - Recurrent neural network: $S_t^a = \sigma(S_{t-1}^a W_s + O_t W_o)$

The RL Problem
Inside An RL Agent

Major Components of an RL Agent

An RL agent may include one or more of these components:

- *Policy*: agent's behavior function
- *Value function*: how good is each state and/or action
- *Model*: agent's representation of the environment

Policy

- A **policy** is the agent's behavior
- It is a map from state to action, e.g.,
- Deterministic policy: $a = \pi(s)$
- Stochastic policy: $\pi(a|s) = \mathbb{P}(A_t = a | S_t = s)$

Value Function

- **Value function** is a prediction of future reward
- Used to evaluate the goodness/badness of states
- And therefore to select between actions, e.g.,

$$v_\pi(s) = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots | S_t = s] \tag{7}$$

Model

- A **model** predicts what the environment will do next
- Transitions: $\mathcal{P}$ predicts the next state
- Rewards: $\mathcal{R}$ predicts the next (immediate) reward, e.g.,

$$\mathcal{P}^a_{ss'} = \mathbb{P}(S_{t+1} = s' | S_t = s, A_t = a) \tag{8}$$

$$\mathcal{R}^a_{ss'} = \mathbb{E}(R_{t+1} | S_t = s, A_t = a) \tag{9}$$

Maze Example

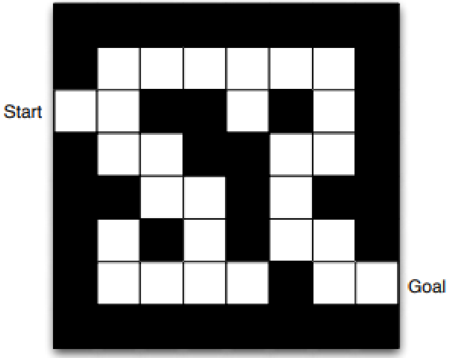


Figure: Maze Example

- Rewards: -1 per time step
- Actions: N, S, E, W
- States: Agent's location

Maze Example: Policy and Value Function

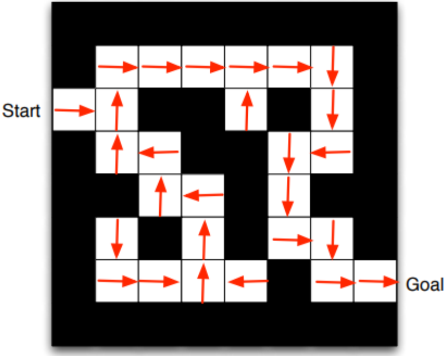


Figure: Arrows represent **policy** $\pi(s)$ for each state s

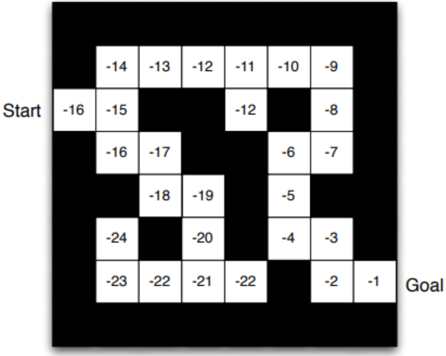
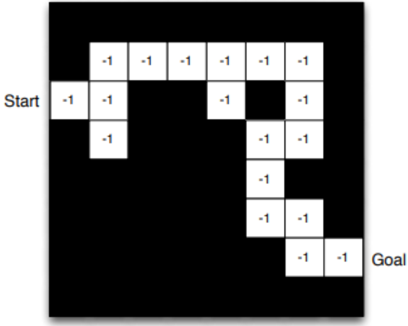


Figure: Numbers represent **value** $v_\pi(s)$ for each state s

Maze Example: Model



- Agent may have an internal model of the environment
- Dynamics: how actions change the state
- Rewards: how much reward from each state
- The model may be imperfect

- Grid layout represents transition model $\mathcal{P}^a_{ss'}$
- Numbers represent immediate reward $\mathcal{R}^a_s$ from each state s (same for all a)

The RL Problem
Problems within RL

Exploration and Exploitation

- RL is like trial-and-error learning
- The agent should discover a good policy
- From its experiences of the environment
- Without losing too much reward along the way

- Exploration finds more information about the environment
- Exploitation exploits known information to maximize reward
- **It is usually important to explore as well as exploit**

Exploration and Exploitation: Examples

- Restaurant Selection
  Exploitation: Go to your favorite restaurant every time
  Exploration: Try a new restaurant
- Online Banner Advertisements
  Exploitation: Show the most successful advertisement
  Exploration: Show a different advertisement
- Oil Drilling
  Exploitation: Drill at the best known location
  Exploration: Drill at a new location
- Game Playing
  Exploitation: Play the move you believe is best
  Exploration: Play an experimental move

Bibliography

Gangula, R., Esrafilian, O., Gesbert, D., Roux, C., Kaltenberger, F., and Knopp, R. (2018).
Flying rebots: First results on an autonomous uav-based lte relay using open airinterface.
In *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pages 1–5.

Silver, D. (2015).
Lectures on reinforcement learning.
url: https://www.davidsilver.uk/teaching/.