

COMS 4030A/7047A

Adaptive Computation and Machine Learning

Hima Vadapalli

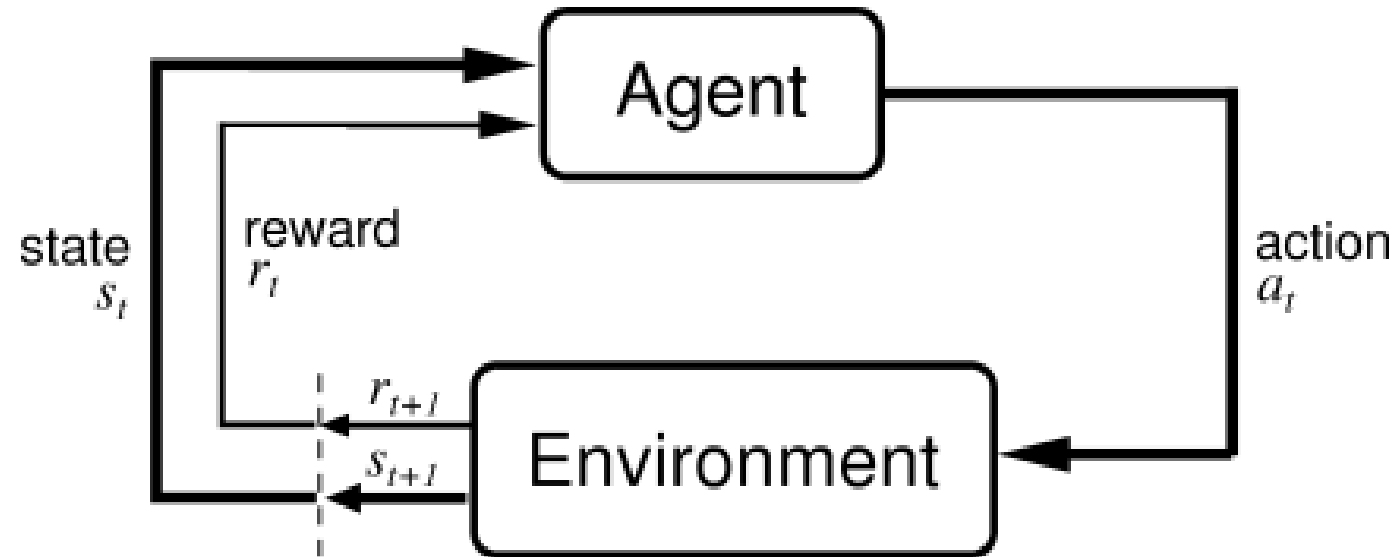
Semester I, 2022

*So far:
Supervised and Unsupervised ML
paradigms*

*Today:
the other paradigm – Reinforcement
Learning*


Reinforcement Learning

- Basic idea:
 - Receive feedback in the form of **rewards**
 - Agent's utility is defined by the reward function
 - Must (learn to) act so as to maximize expected rewards



Characteristics of Reinforcement Learning

What makes reinforcement learning different from other machine learning 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

Examples of Reinforcement Learning

- Fly stunt manoeuvres in a helicopter
- Manage an investment portfolio
- Make a humanoid robot walk
- Play many different Atari games better than humans

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 maximise cumulative reward

Reinforcement learning is based on the **reward hypothesis**

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

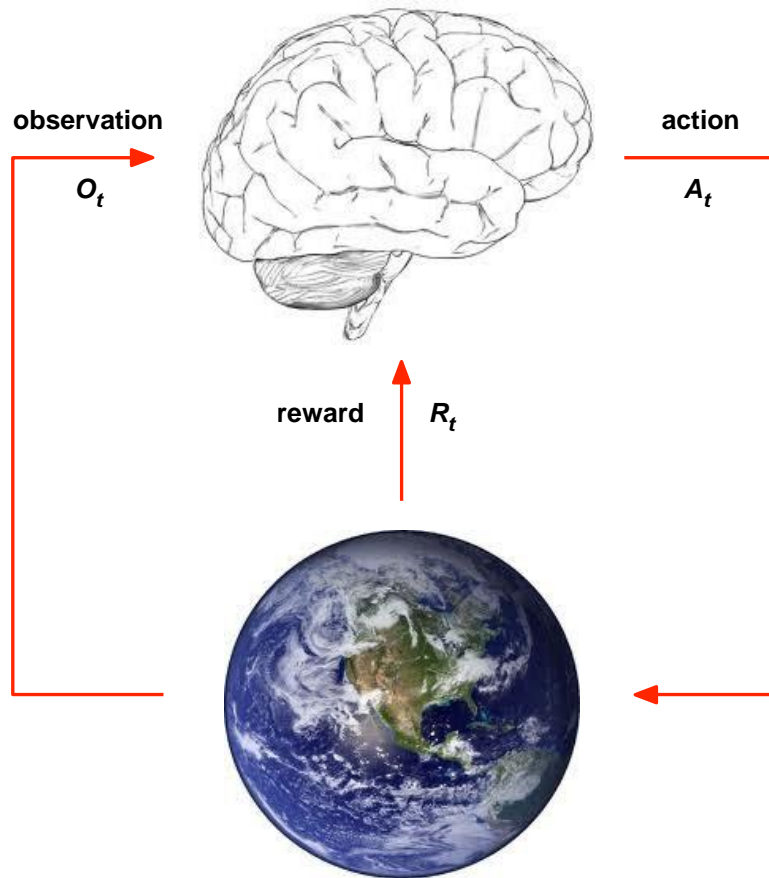
Examples of Rewards

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

Sequential Decision Making

- Goal: *select actions to maximise 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)

Agent and Environment



- 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 env. step

History and State

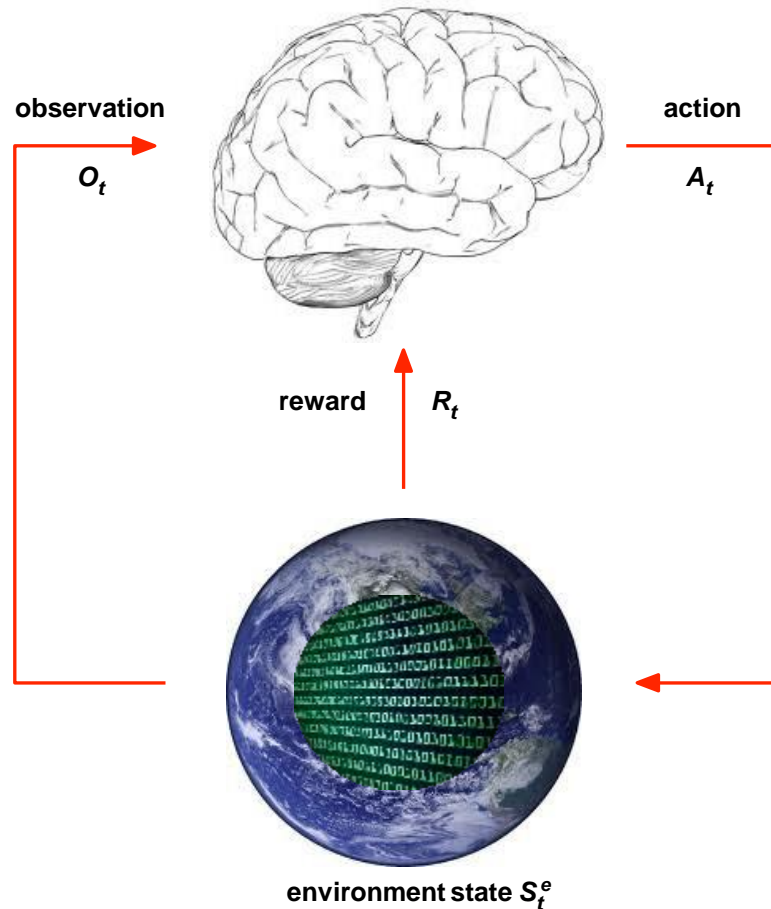
- The **history** is the sequence of observations, actions, rewards

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

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

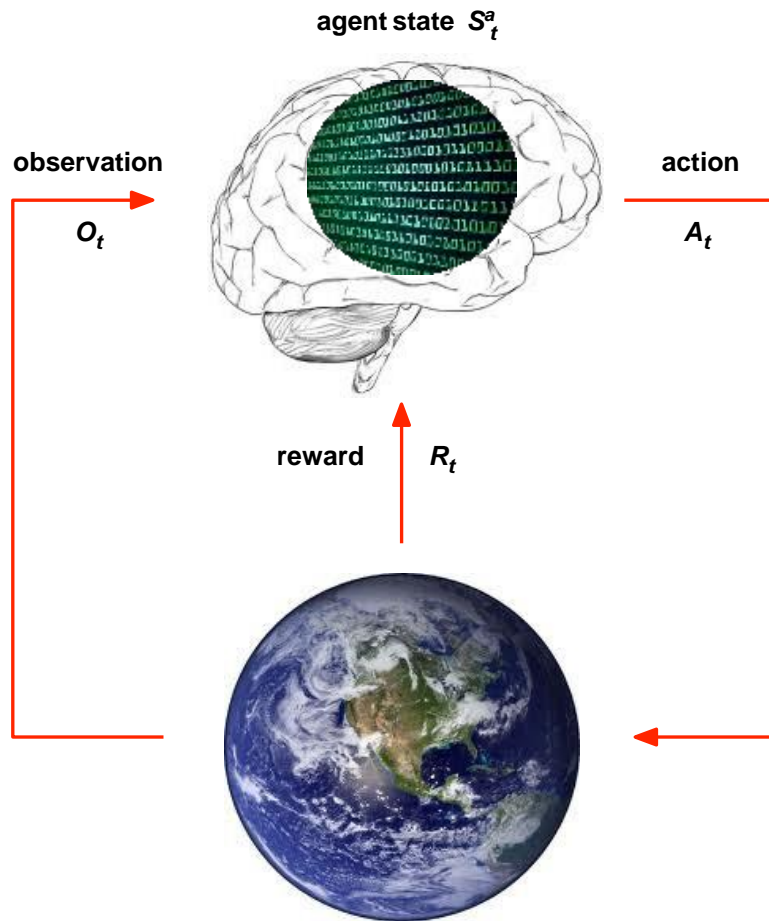
$$S_t = f(H_t)$$

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^e is visible, it may contain irrelevant information

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 reinforcement learning algorithms
- It can be any function of history:

$$S_t^a = f(H_t)$$

Information State

- An **information state** (a.k.a. **Markov state**) contains all useful information from the history.
- A state S_t is **Markov** if and only if

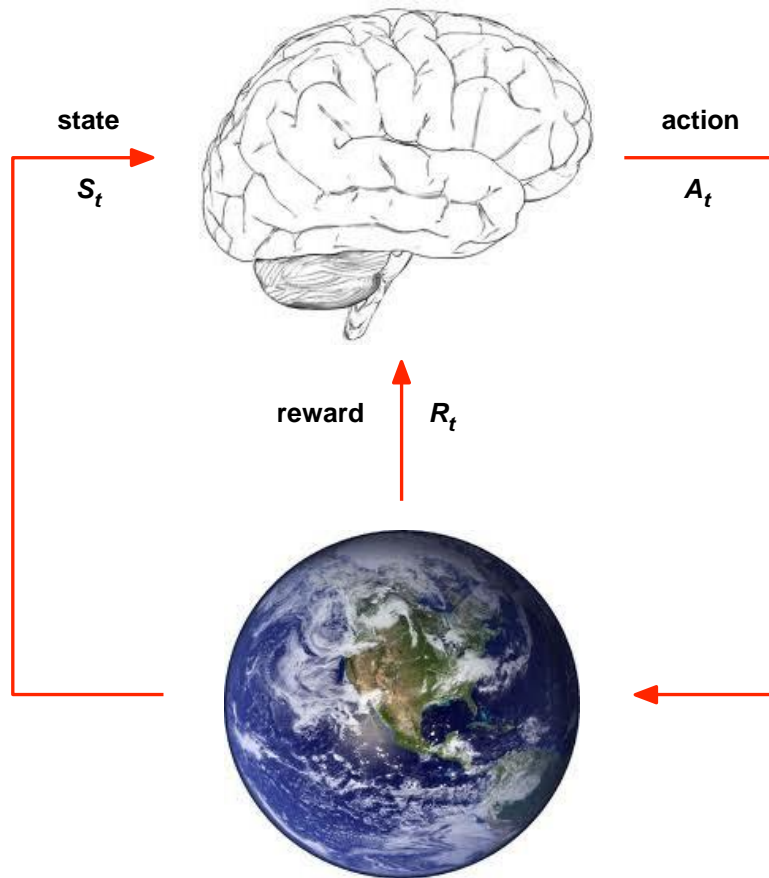
$$P[S_{t+1} | S_t] = P[S_{t+1} | S_1, \dots, S_t]$$

- “The future is independent of the past given the present”

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

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

Fully Observable Environment



- **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 Environment

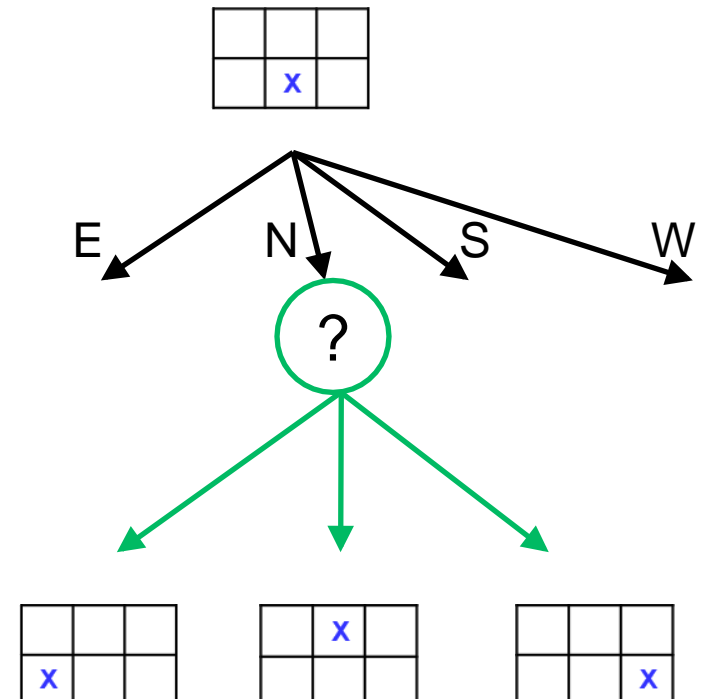
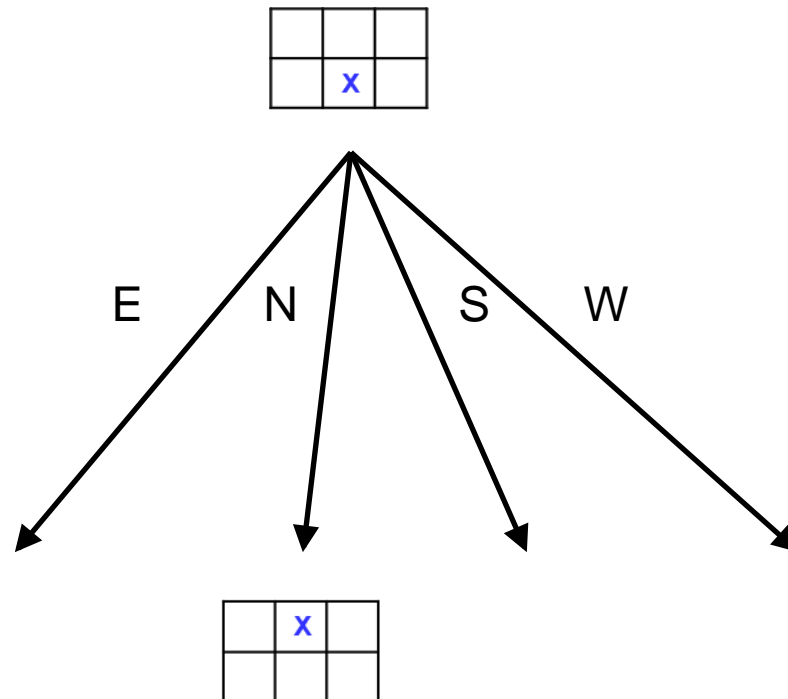
- **Partial observability:** agent **indirectly** observes environment:
 - A robot with camera vision isn't told its absolute location
 - A trading agent only observes current prices
- Now agent state \neq environment state
- Formally this is a **partially observable Markov decision process** (POMDP)
- Agent must construct its own state representation S_t^a

Major components of an RL Agent

- An RL agent may include one or more of these components:
 - Policy: agent's behaviour function
 - Reward function: next (immediate) reward
 - Value function: how good is each state and/or action
 - Model: agent's representation of the environment

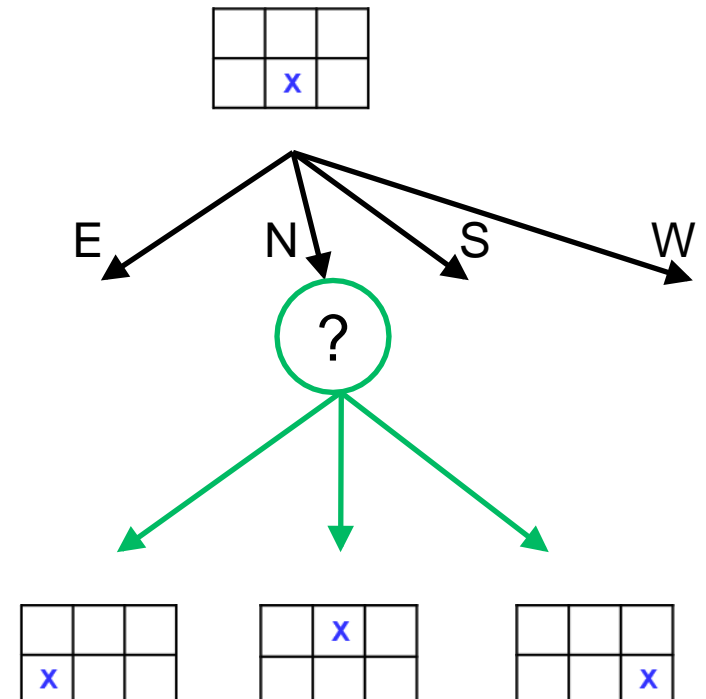
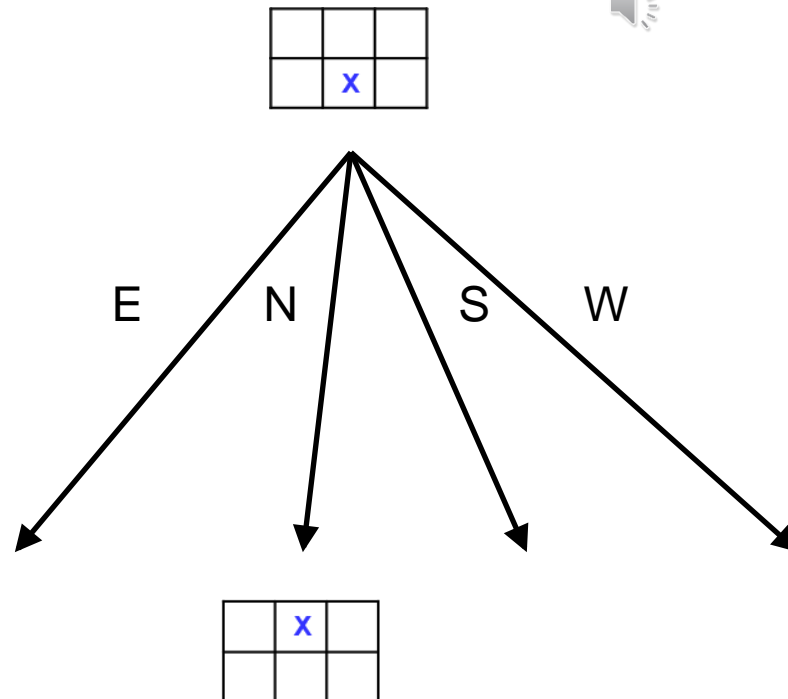
Policy

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



Policy

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



Reward signal (function)

- Defines what are good and bad events for the agent
- At each time step:
 - Agent receives a single number i.e reward

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

Objective : maximise the total reward it received over the long run

Value Function

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

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

- Typically discount rewards by $\gamma < 1$ each time step
 - Sooner rewards have higher utility than later rewards
 - Also helps the algorithms converge

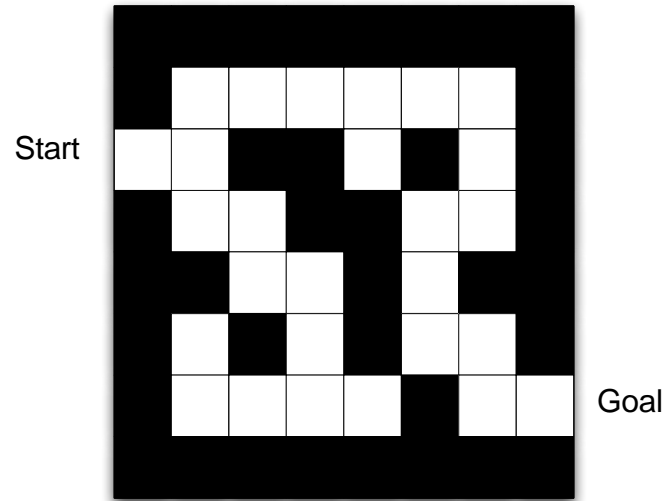
Model

- A **model** predicts what the environment will do next
 - Mimics the behaviour of environment
- Given a state and action, the model might
 - predict the next state (\mathcal{P})
 - predicts the next (immediate) reward (\mathcal{R})

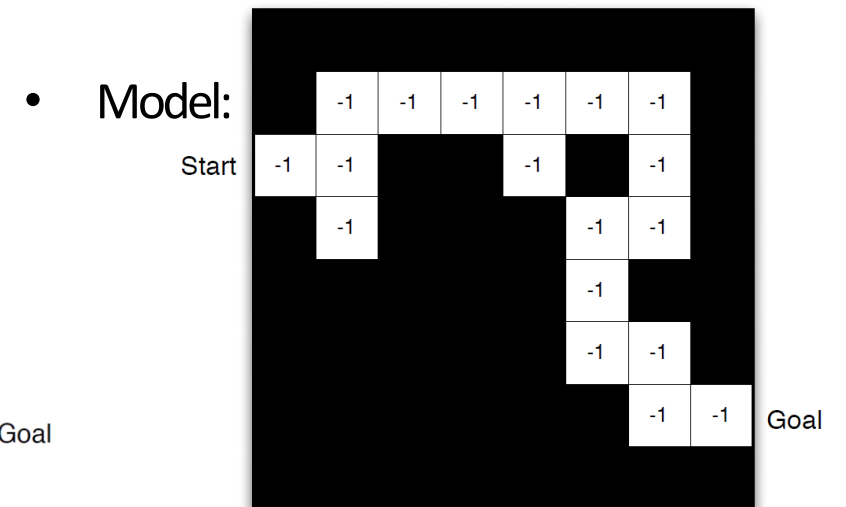
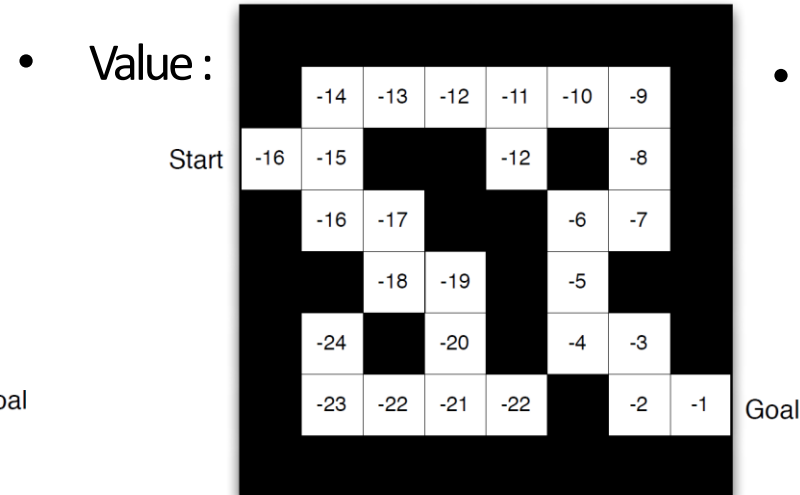
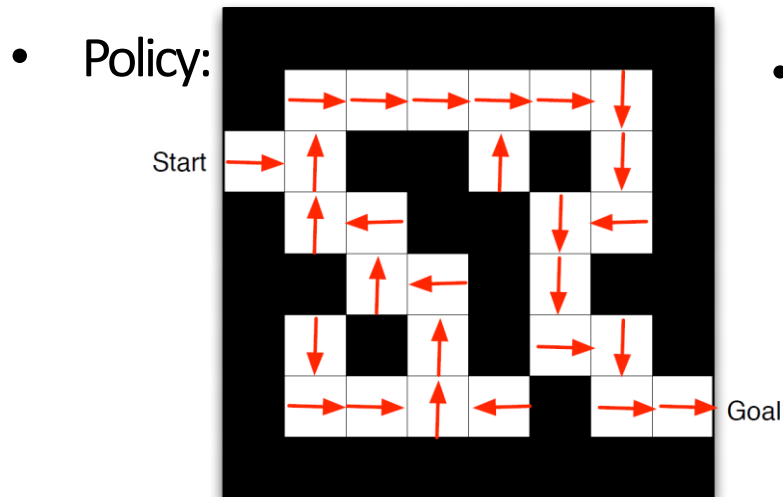
$$\mathcal{P}_{ss'}^a = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$$

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

Maze example to understand the components



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



Categories of RL Agents:

- Value Based
 - No Policy (Implicit)
 - 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

Learning and Planning

Two fundamental problems in sequential decision making

- Reinforcement Learning:
 - The environment is initially unknown
 - The agent interacts with the environment
The agent improves its policy
- Planning:
 - A model of the environment is known
 - The agent performs computations with its model (without any external interaction)
 - The agent improves its policy
 - a.k.a., search

Exploration and Exploitation

- Reinforcement learning 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

Game Playing

Exploitation: Play the move you believe is best

Exploration: Play an experimental move

Prediction and Control

- Prediction: evaluate the future
 - Given a policy
- Control: optimise the future
 - Find the best policy