Reinforcement Learning

Prof. Shivali Dhaka

Outline

- Prerequisites
- Course Information
- About Reinforcement Learning
- The Reinforcement Learning Problem
- Inside an RL Agent
- Problems with RL

Prerequisites

- Knowledge of programming in Python Probability,
- Calculus, linear algebra.
- General comfort with math Knowledge of machine learning.

Outline

- Prerequisites
- Course Information
- About Reinforcement Learning
- The Reinforcement Learning Problem
- Inside an RL Agent
- Problems with RL

Class Information

- Class Timings:
  - Mondays 3:00-5:50 PM Section-1
  - Wednesdays 8:00-10:50 AM Section-2
- Contact me: Shivali.Dhaka@georgiancollege.ca
- Expect 48 hours for email reply.

Assessment

Assessment	Weightage
Project1	15%
Project 2	20%
Project 3	20%
Discussion Board	5%
Midterm Exam	20%
Final Exam	20%
Total	100%

Text Books

1. An Introduction to Reinforcement Learning, Sutton and Barto, 1998

- MIT Press, 1998
- ~ 40 pounds Available free online!

2. Algorithms for Reinforcement Learning, Szepesvari

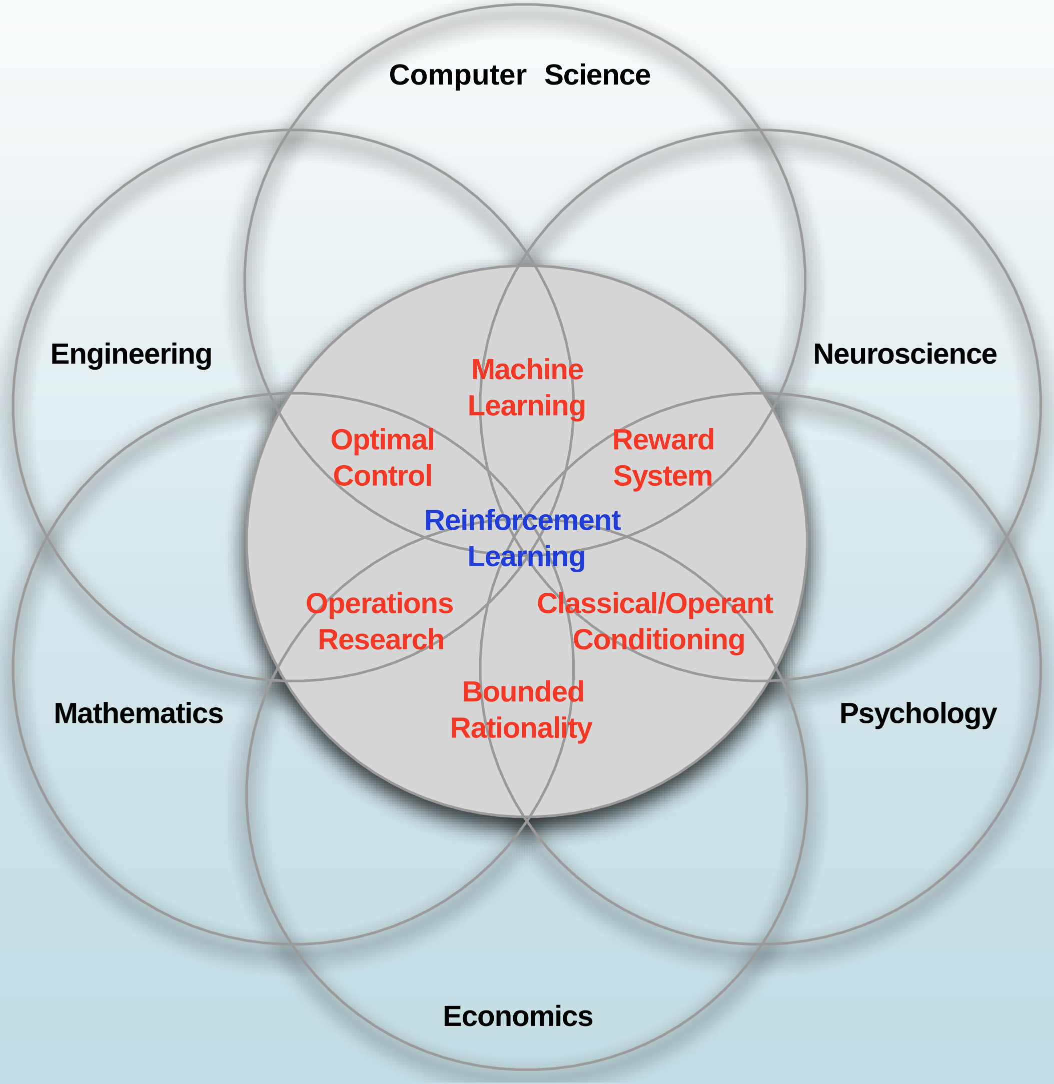
- Morgan and Claypool, 2010
- ~ 20 pounds Available free online!

Note: Pdf books are available in Blackboard under About the course.

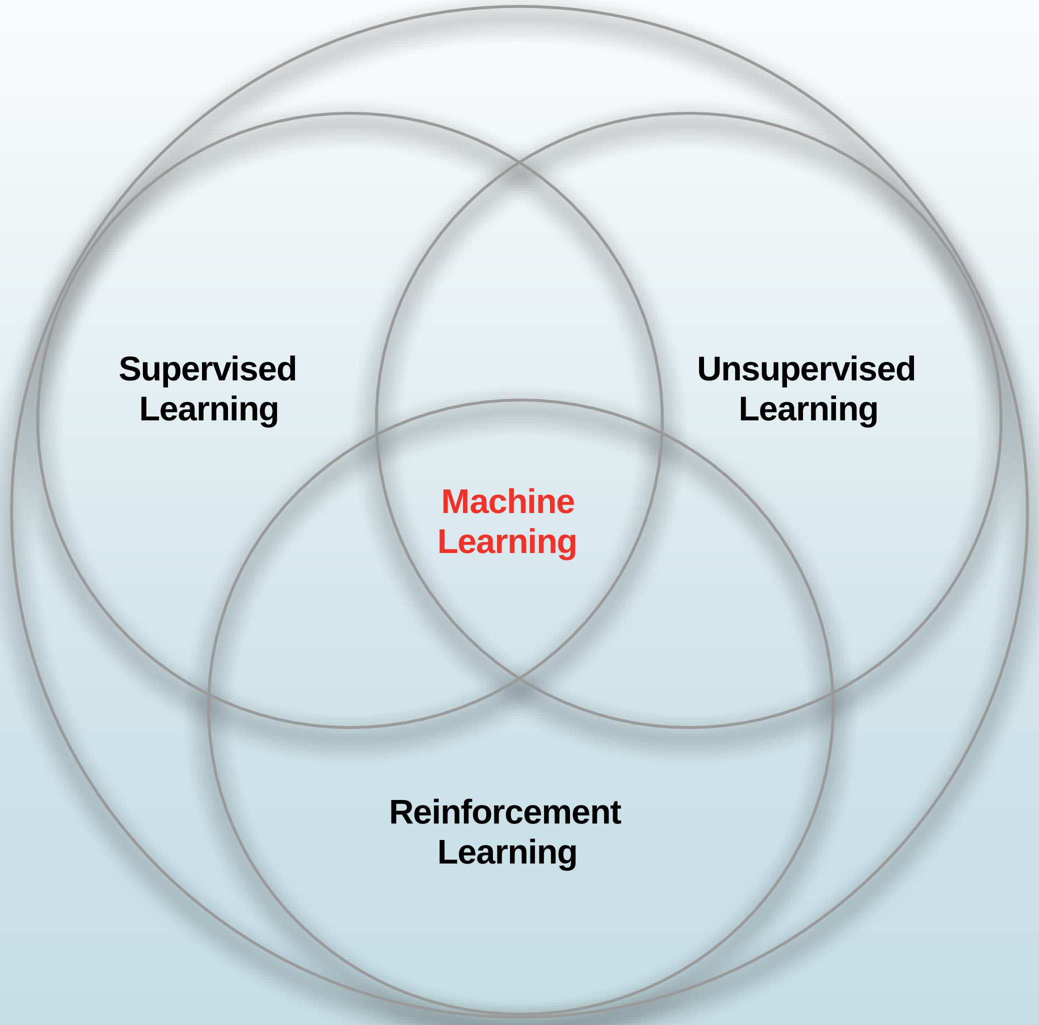
Outline

- Prerequisites
- Course Information
- About Reinforcement Learning
- The Reinforcement Learning Problem
- Inside an RL Agent
- Problems with RL

Many Faces of Reinforcement Learning

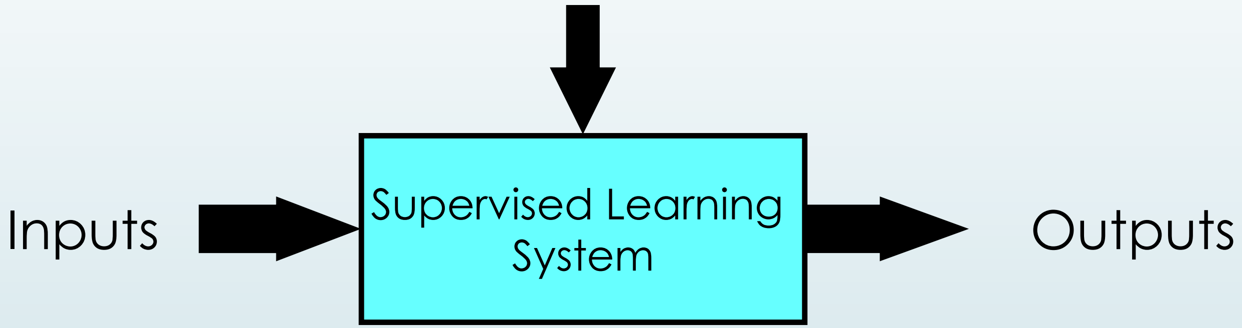


Branches of Machine Learning



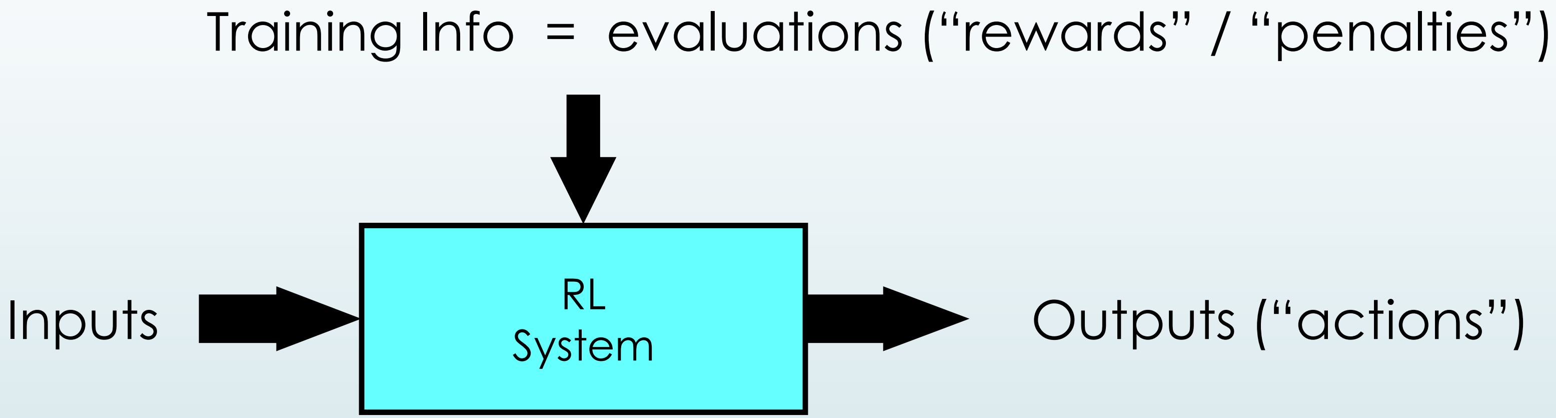
Supervised Learning

Training Info = desired (target) outputs



Error = (target output – actual output)

Reinforcement Learning



Objective: get as much reward as possible

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

Contd…

- Learner is not told which actions to take
- Trial-and-Error search
- Possibility of delayed reward (sacrifice short-term gains for greater long-term gains)
- The need to ***explore*** and ***exploit***
- Considers the whole problem of a goal-directed agent interacting with an uncertain environment

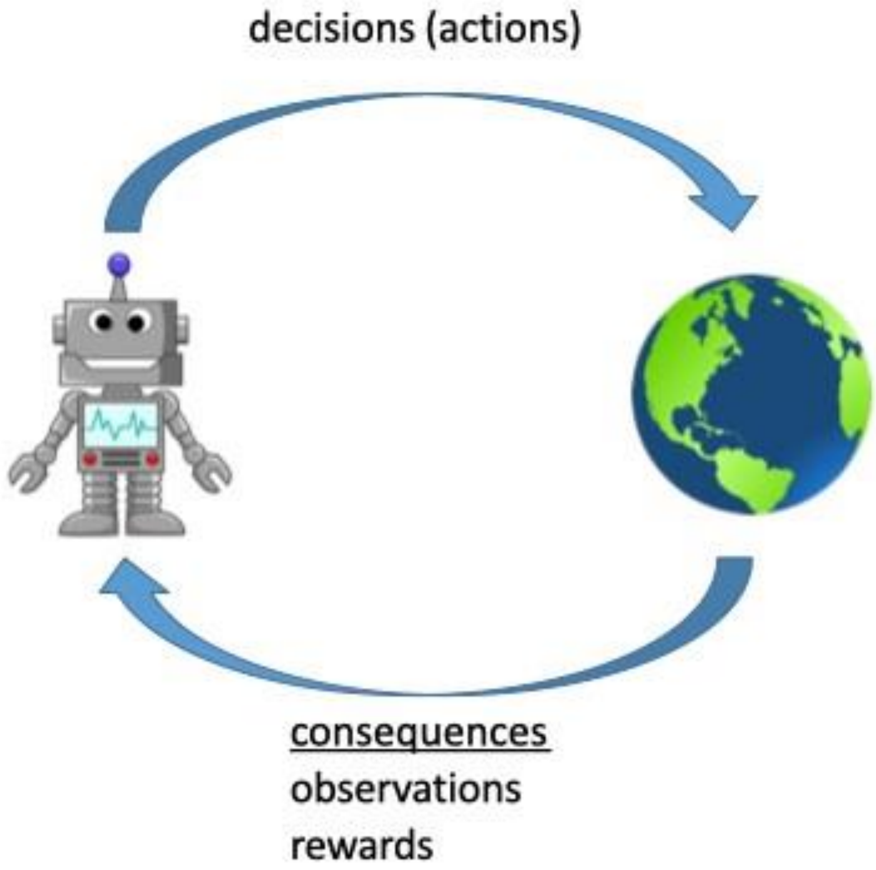
Examples of Reinforcement Learning

- 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 Atari games better than humans

Videos Examples.

- :[(3) Stanford Autonomous Helicopter - Airshow #1 - YouTube]
- [(3) Deep Reinforcement Learning: Agent Playing Atari - YouTube]

Reinforcement learning provides a formalism for *behavior*



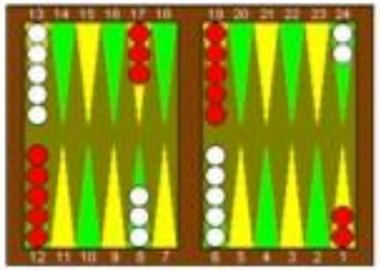
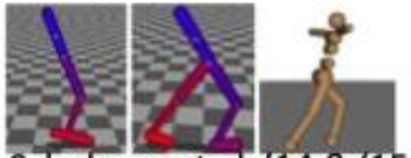


Figure 2. An illustration of the normal opening position in backgammon. TD-Gammon has sparked a near-universal conversion in the way experts play certain opening rolls. For example, with an opening roll of 4-1, most players have now switched from the traditional move of 13-9, 6-5, to TD-Gammon's preference, 13-9, 24-23. TD-Gammon's analysis is given in Table 2.

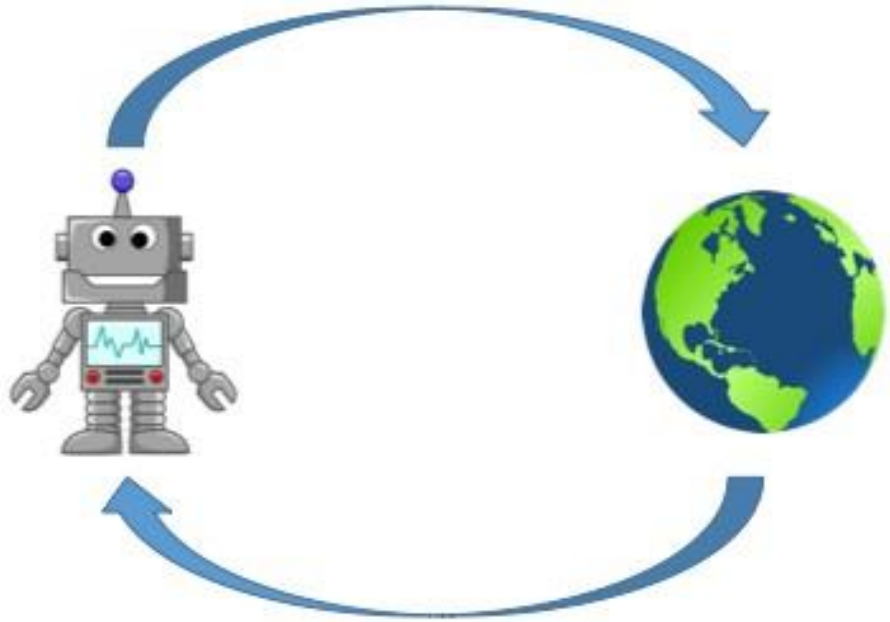


Schulman et al. '14 & '15

Mnih et al. '13

Levine*, Finn*, et al. '16

decisions (actions)



consequences
observations (states)
rewards

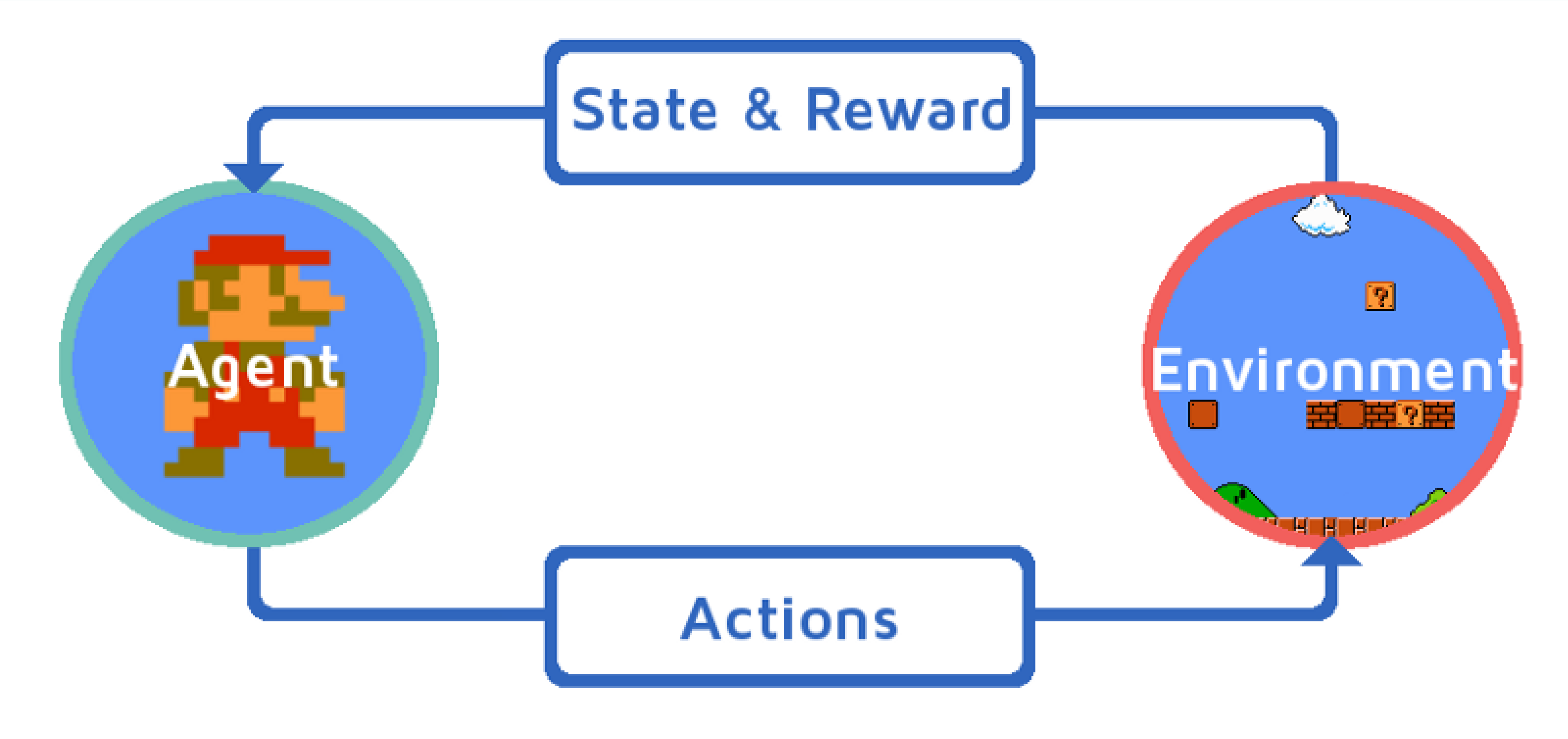
Actions: muscle contractions
Observations: sight, smell
Rewards: food

Actions: motor current or torque
Observations: camera images
Rewards: task success measure (e.g., running speed)

Actions: what to purchase
Observations: inventory levels
Rewards: profit

TERMS IN RL

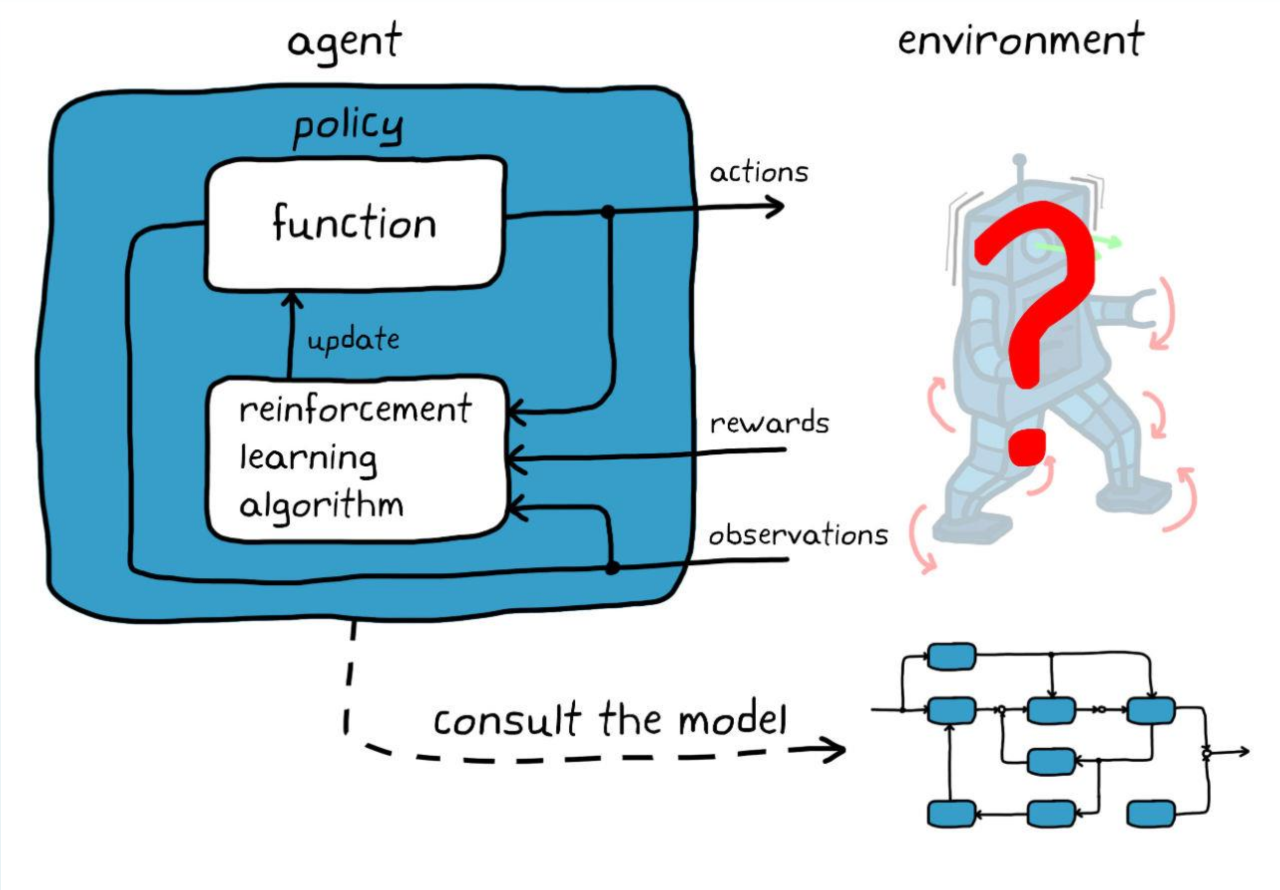
- 1. Agent: Main property in RL
  - Example; It could be a robot learning to walk or an agent learning to drive.
  - RL agents observe and explore the environment to learn.
- 2.State: This is for the position at which the agent is at a given period. It changes when agent moves.
- 3. Environments: The environment that agent learns to improve.
  - Different position in the environment represent the state.
- 4. Action: Agents choice of activity in a state.
  - Whatever acts or steps agent decide to take after observing the environment.
  - If the action taken by the agent is correct, it gets a positive reward.



[Action reward reinforcement learning - Bing images]

- 5. Reward: Prize/Score for taking a right or wrong action.
  - Correct Action--→ Positive Reward
  - Wrong Action -→ Negative Reward
    - When Agent fails and gets a negative reward, it learns from it and changes its actions with the aim of choosing right actions.
  - 6. Policy: Strategy for deciding the best action. These strategies in RL is known as policies.
  - 7. Goal: The Agent's mission.
    - Finally, when the agents explore an environment, it has mission, it has something it wants to learn and that's its goal.

22



[Model Free Reinforcement Learning - Bing images]

RL Problem

- It should have all or some of the stated properties in previous slides.
- Model-Free:
  - By some, I mean in cases where the model of the environment is unknown, the agent is set to be performing Model-Free prediction. This means trying to predict action in a state without knowing what the environment looks like.
  - Example: Agent visiting new places and learning new environment.
- Model-Based:
  - The second way of learning by agent is Model-Based prediction method where the agent learn with full knowledge of its environment.
  - Example: Agent living in a house for 2 years and know all about its environment so his action will be learning in model-based environment.

Outline

- Prerequisites
- Course Information
- About Reinforcement Learning
- The Reinforcement Learning Problem
- Inside an RL Agent
- Problems with RL

Reward

- In both cases discussed in previous slides, actions leads to rewards-
  - Positive
  - Negative
- This reward helps you decide on your next action and that’s how exactly a reinforcement learning agent learns.

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

Definition (Reward Hypothesis)

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

Do you agree with this statement?

Examples of Rewards

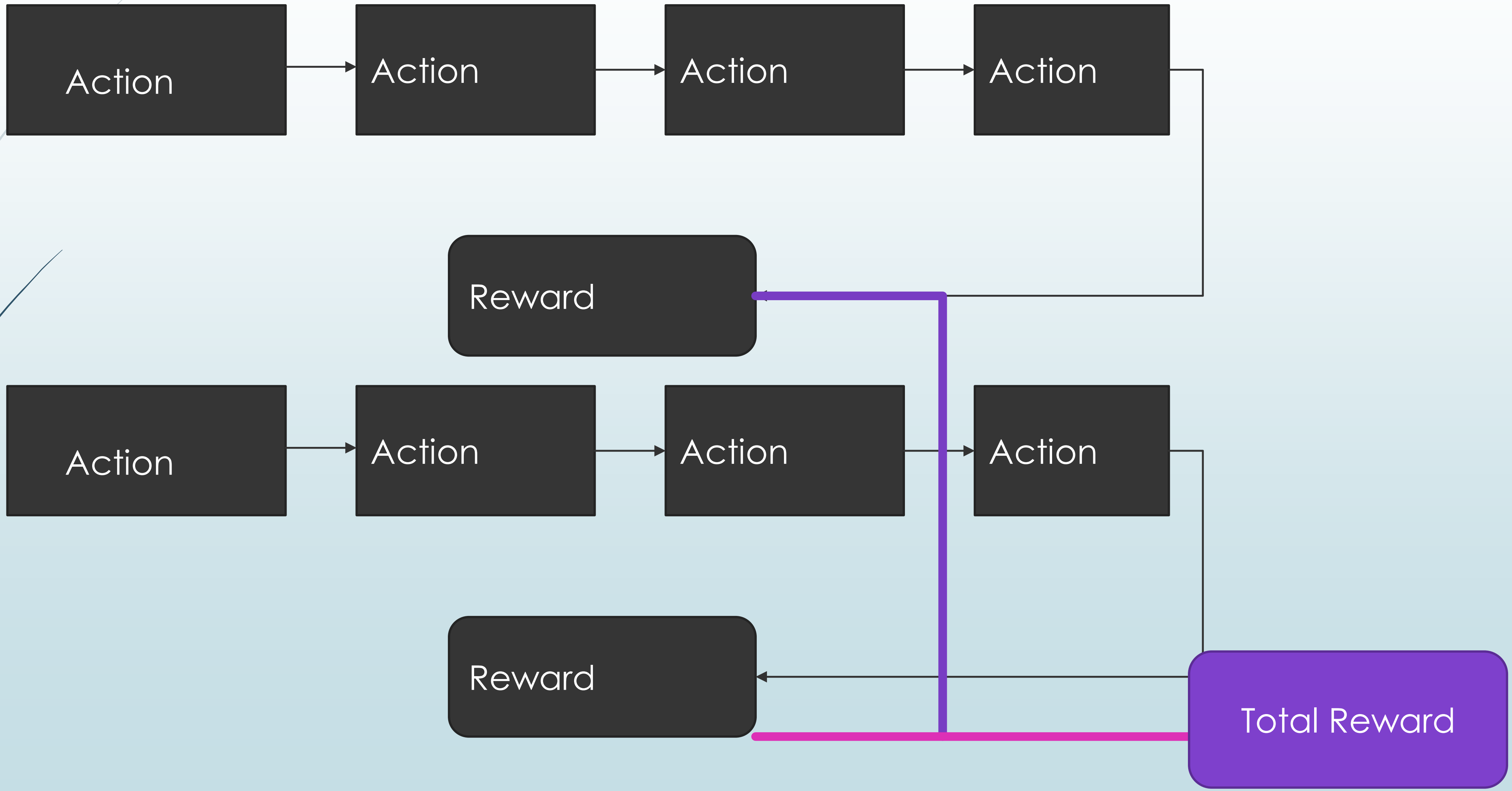
- Fly stunt manoeuvres in a helicopter
  - +ve reward for following desired trajectory
  - −ve reward for crashing
- Defeat the world champion at Backgammon
  - +/−ve reward for winning/losing a game
- Manage an investment portfolio
  - +ve reward for each $ in bank
- Control a power station
  - +ve 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

Goals

- Episodic Tasks:
  - These are tasks that have a defined goal or end point. After this agent stops learning as they happen in a episode and have some final state.
  - Mostly solved by model-based methods.
  - Simply short which means a simpler environment to understand.

- Continuing Tasks:
  - These are the tasks that don’t have an end point, they continue forever.
  - Mostly solved by model-free methods because it has larger environment space which cannot be totally understood by the agent.

Episode



Action
Action
Action
Action
Reward
Action
Action
Action
Action
Reward
Total Reward

- In some cases, reward doesn’t come immediately after the action, but after a set of actions. This set of different actions in different states before a reward is known as episode.
- So, many episodes can occur before an agent reaches its goal and its final reward which is the sum of all rewards gotten at the end of all episodes.

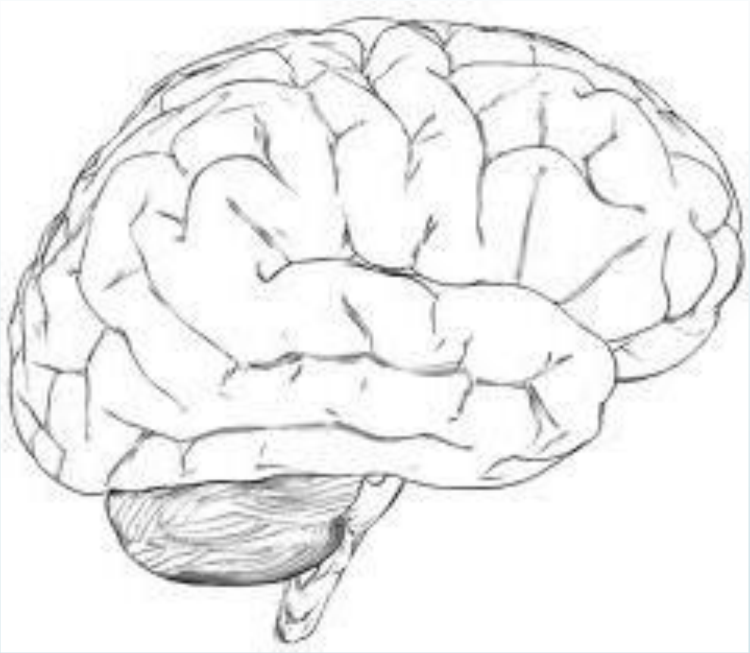
Sequential Decision Making

- 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

observation

O_t



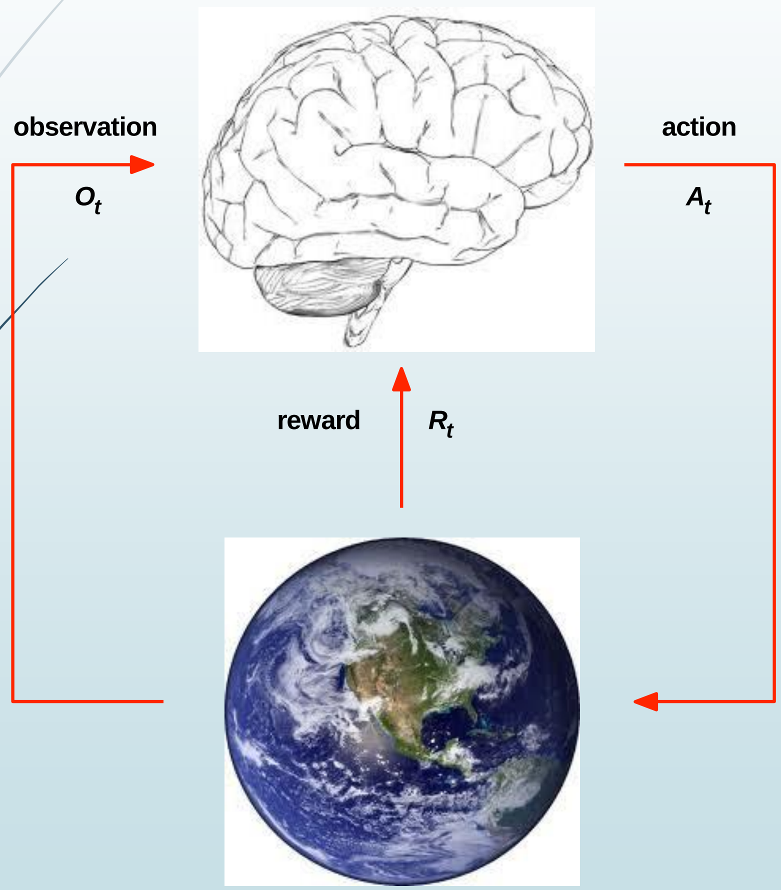
action

A_t

reward

R_t

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

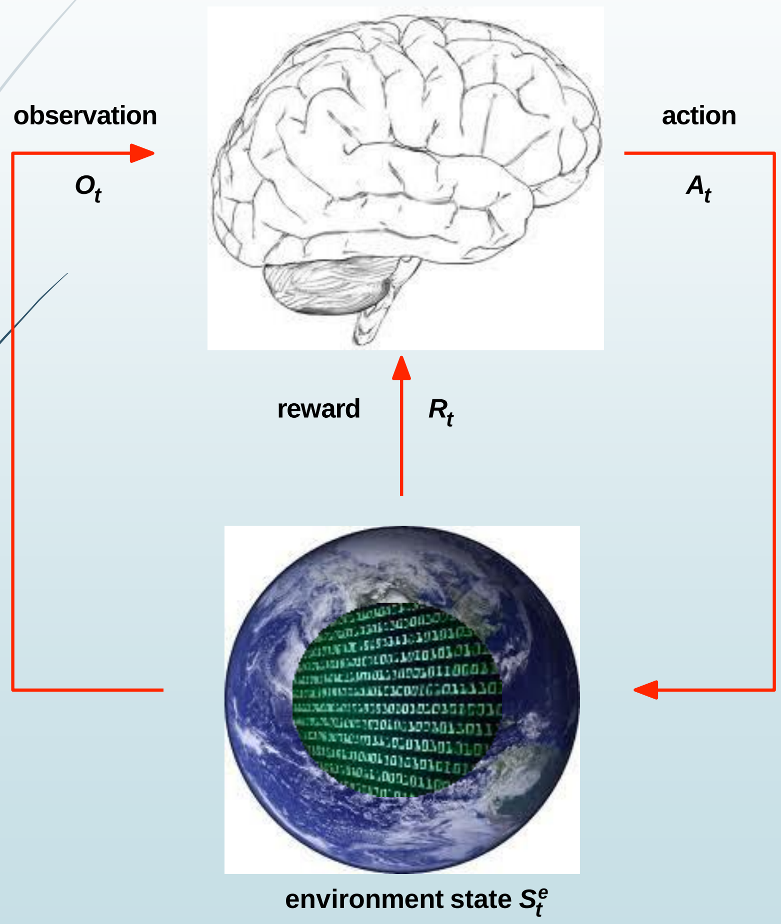
- The history is the sequence of observations, actions, rewards

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

- 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:

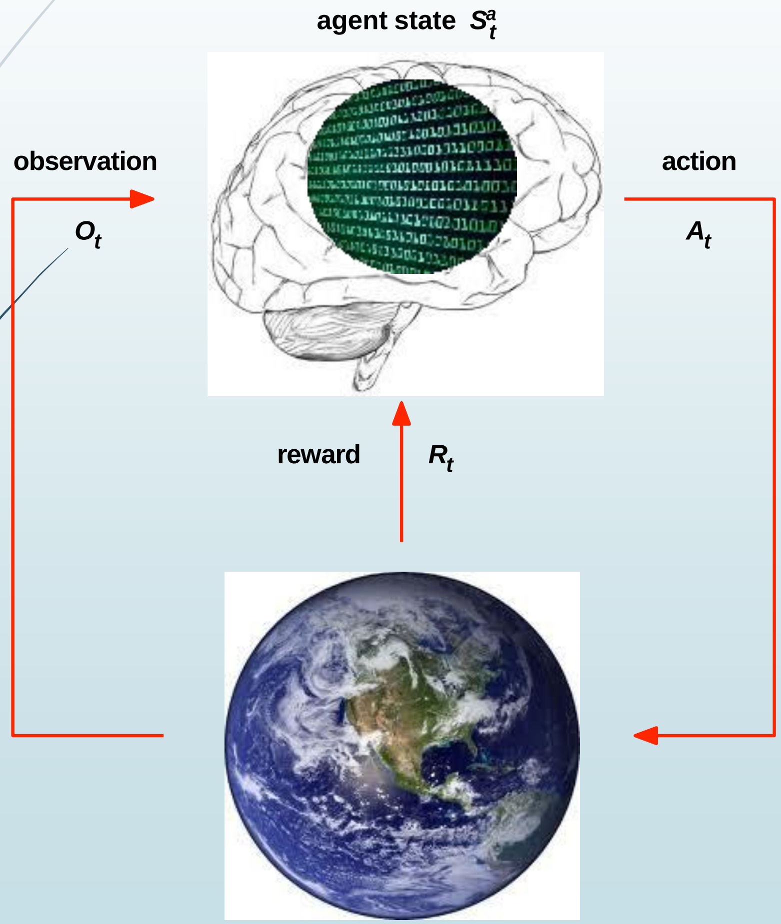
$$S_t = f(H_t)$$

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_t^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.

Definition

A state S_t is Markov if and only if

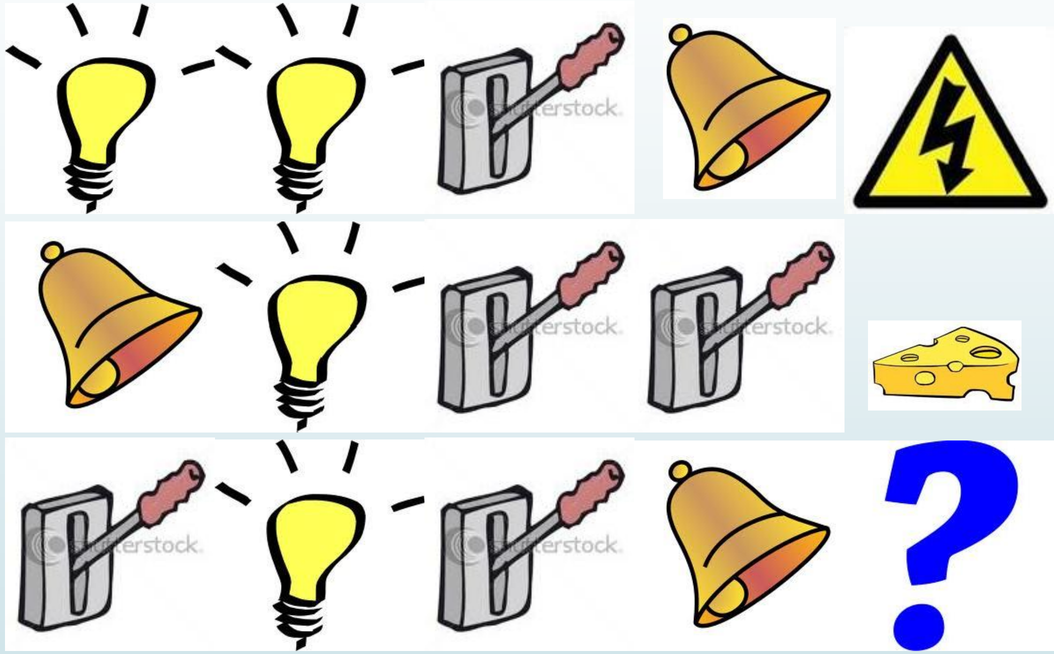
$$\mathrm{P}[S_{t+1} \mid S_t] = \mathrm{P}[S_{t+1} \mid S_1, \ldots, S_t]$$

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

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

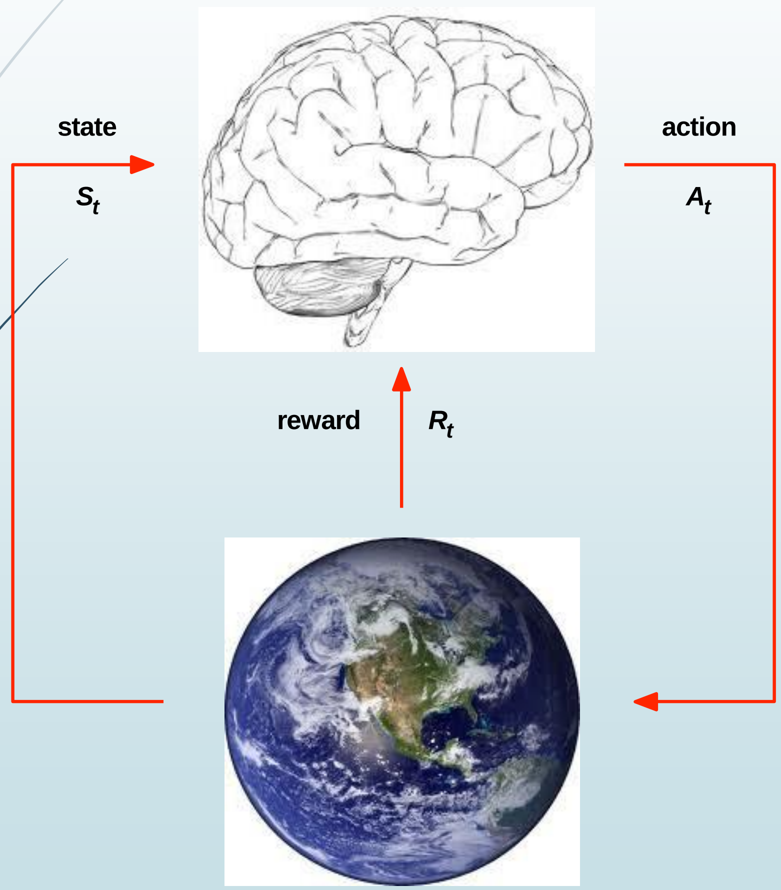
- Once the state is known, the history may be thrown away
- i.e. The state is a sufficient statistic of the future
- The environment state S_t^e is Markov
- The history H_t is Markov

Rat Example



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

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)
- (Next lecture and the majority of this course)

Example of Fully Observable Env.

- In the game of chess, the agent can always see the complete position of itself and its opponent on the board. Hence it is a fully observable environment.

Partially Observable Environments

- Partial observability: agent indirectly observes environment:
  - A robot with 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 Markov decision process (POMDP)
- Agent must construct its own state representation S_t^a, e.g.
  - Complete history: $S_t^a = H_t$
  - Beliefs of environment state: $S_t^a = (\mathbb{P}[S_t^e = s^1], ..., \mathbb{P}[S_t^e = s^n])$
  - Recurrent neural network: $S_t^a = \sigma(S_{t-1}^a W_s + O_t W_o)$

Example of Partially Observable Env.

- In the game of poker, the agent cannot see the hands of the opponent for the most part of the game, and hence it is a partially observable environment.

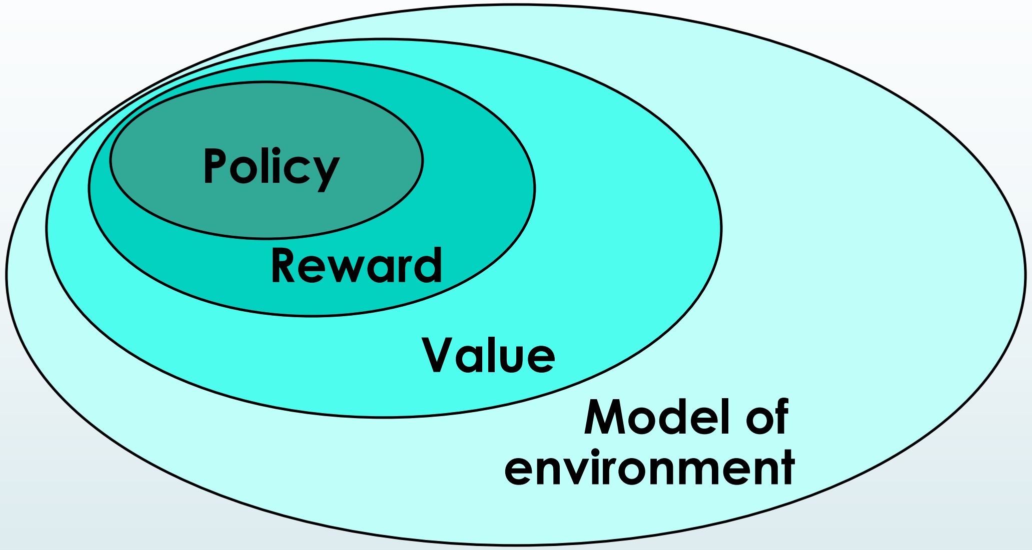
Outline

- Prerequisites
- Course Information
- About Reinforcement Learning
- The Reinforcement Learning Problem
- Inside an RL Agent
- Problems with RL

Major Components of an RL Agent

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

Elements of RL



- **Policy**: what to do
- **Reward**: what is good
- **Value**: what is good because it *predicts* reward
- **Model**: what follows what

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: $\pi(a|s) = \mathbb{P}[A_t = a | S_t = s]$

Deterministic vs Stochastic Environment

- **Deterministic Environment**
  - In a deterministic environment, the next state of the environment can always be determined based on the current state and the agent's action.
  - For example, while driving a car if the agent performs an action of steering left, the car will move left only. In a perfect world, it will not happen that you steer the car left but it moves right, it will always move left and this is deterministic.
- **Stochastic Environment**
  - In a stochastic reinforcement learning environment, we cannot always determine the next state of the environment from the current state by performing a certain action.
  - For example, suppose our agent's world of driving a car is not perfect. When an agent applies a break there is a small probability that the break may fail and the car does not stop. Similarly, when the agent tries to accelerate the car, it may just stop with a small probability. So in this environment, the next state of the car cannot always be determined based on its current state and the agent's action.

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.

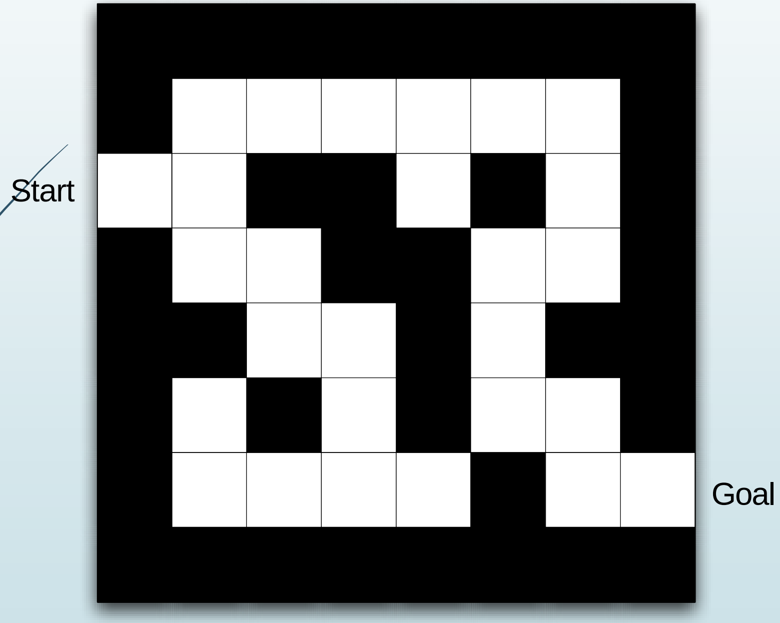
$$v_\pi(s) = \mathbb{E}_\pi \left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \mid S_t = s \right]$$

Model

- A model predicts what the environment will do next
- P predicts the next state
- R predicts the next (immediate) reward, e.g.

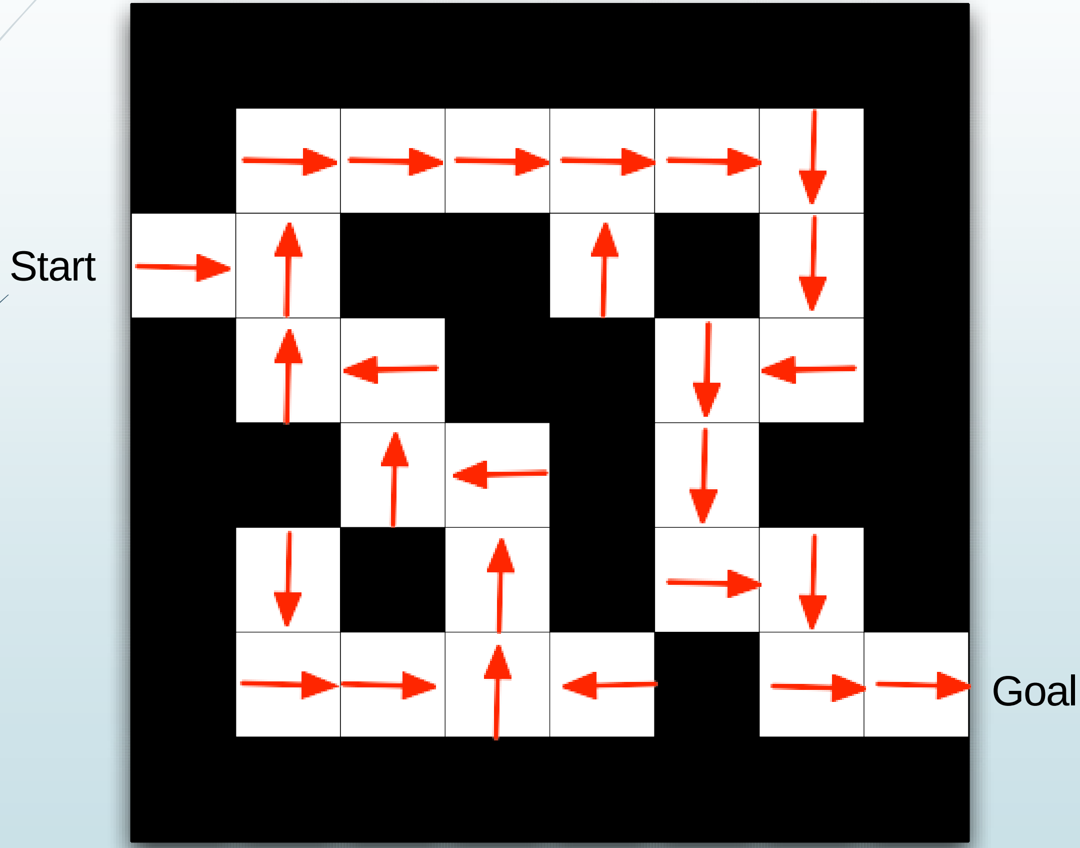
$$P^a_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$$
$$R^a_s = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$$

Maze Example



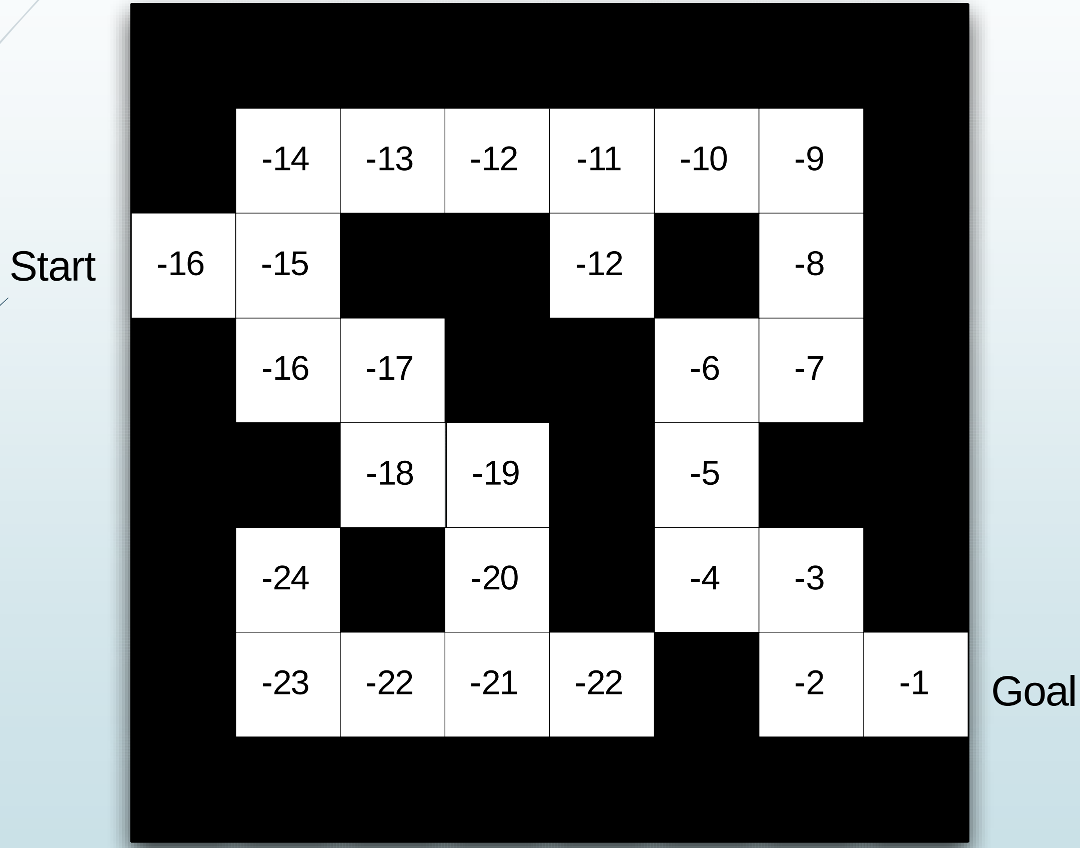
- Rewards: -1 per time-step
- Actions: N, E, S, W
- States: Agent's location

Maze Example: Policy



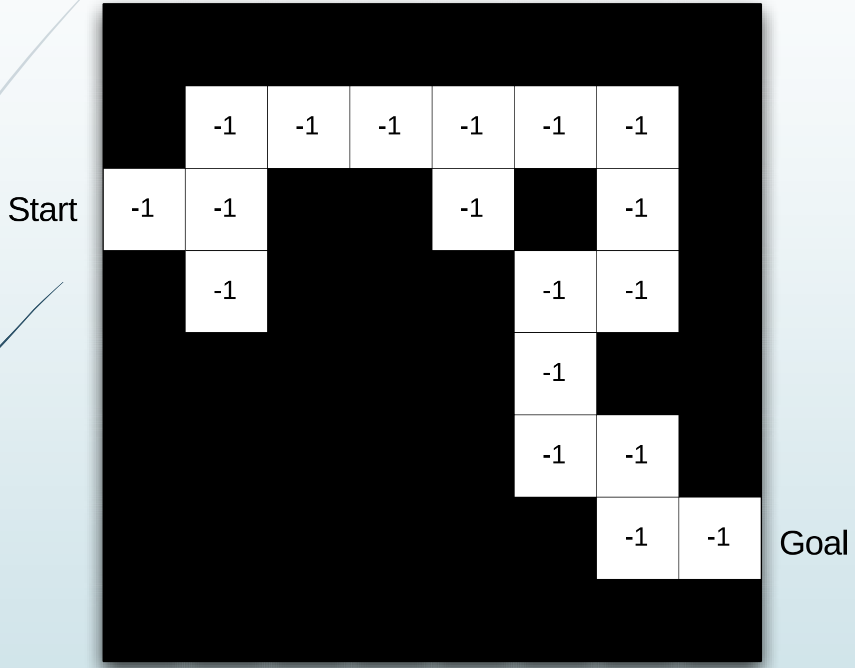
- Arrows represent policy $\pi(s)$ for each state s

Maze Example: Value Function



- Numbers represent value $v_\pi(s)$ of 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 $P^{a}_{ss'}$
- Numbers represent immediate reward R^{a}_{s} from each state s (same for all a)

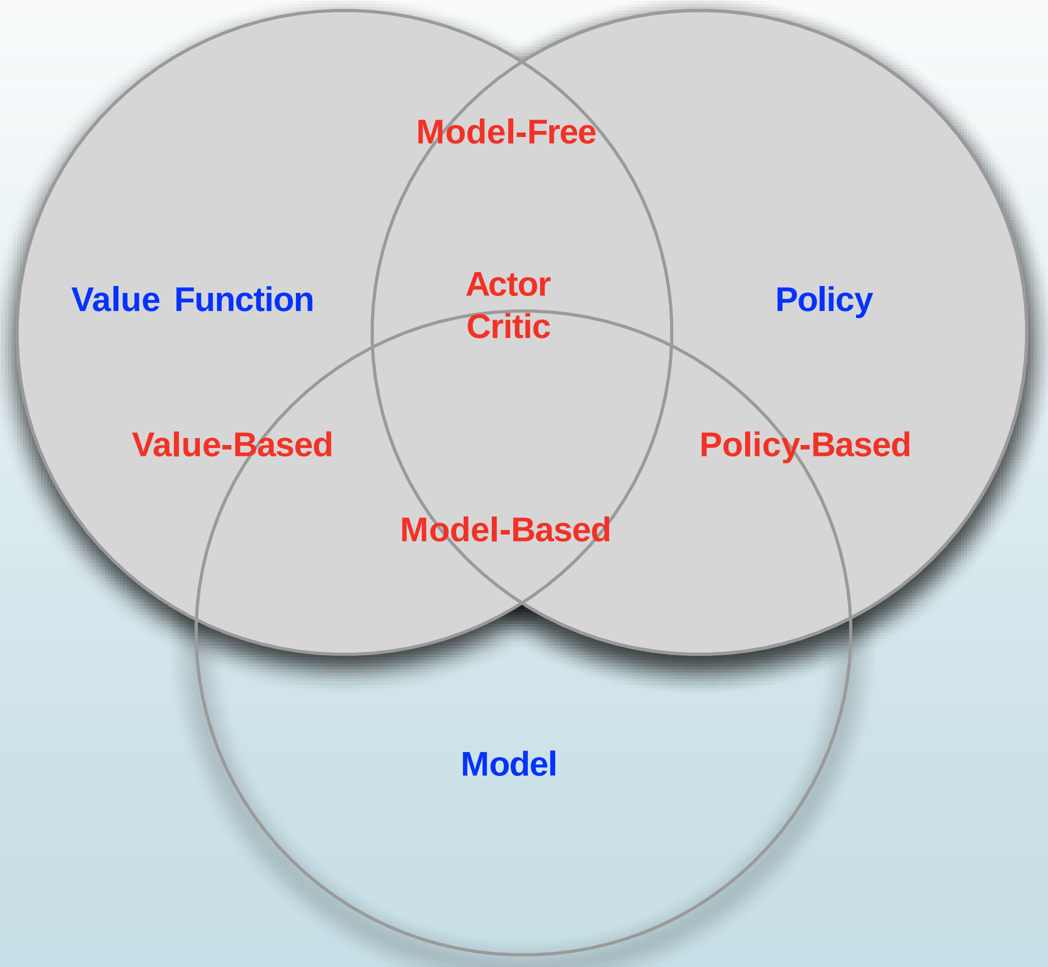
Categorizing RL agents (1)

- Value Based
  - No Policy (Implicit)
  - Value Function
- Policy Based
  - Policy
  - No Value Function
- Actor Critic
  - Policy
  - Value Function

Categorizing RL agents (2)

- Model Free
  - Policy and/or Value Function
  - No Model
- Model Based
  - Policy and/or Value Function
  - Model

RL Agent Taxonomy

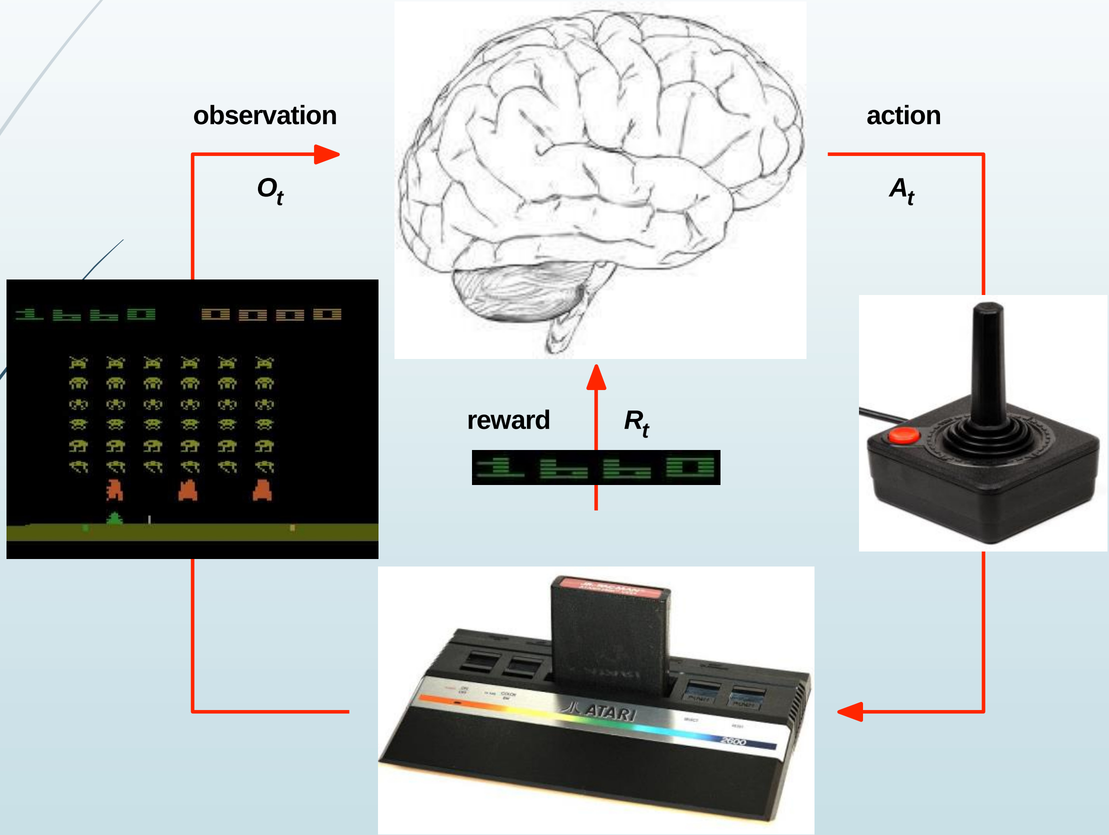


Learning and Planning

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. deliberation, reasoning, introspection, pondering, thought, search

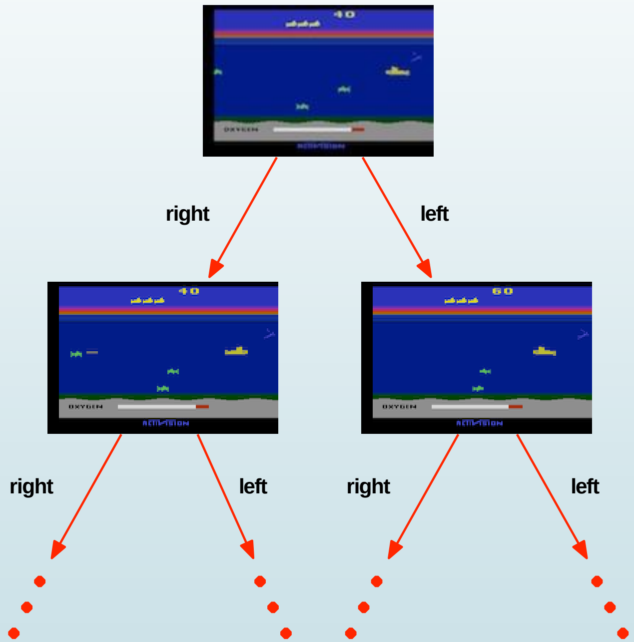
Atari Example: Reinforcement Learning



- Rules of the game are unknown
- Learn directly from interactive game-play
- Pick actions on joystick, see pixels and scores

Atari Example: Planning

- Rules of the game are known
- Can query emulator
  - perfect model inside agent's brain
- If I take action *a* from state *s*:
  - what would the next state be?
  - what would the score be?
- Plan ahead to find optimal policy
  - e.g. tree search



Exploration and Exploitation (1)

- 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 and Exploitation (2)

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

Examples

- Restaurant Selection
  - Exploitation Go to your favourite restaurant
  - Exploration Try a new restaurant
- Online Banner Advertisements
  - Exploitation Show the most successful advert
  - Exploration Show a different advert
- 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

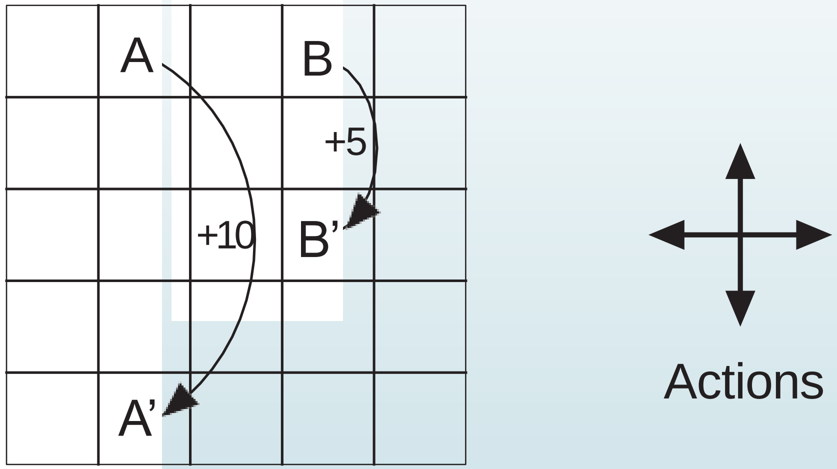
Outline

- Prerequisites
- Course Information
- About Reinforcement Learning
- The Reinforcement Learning Problem
- Inside an RL Agent
- Problems with RL

Prediction and Control

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

Gridworld Example: Prediction



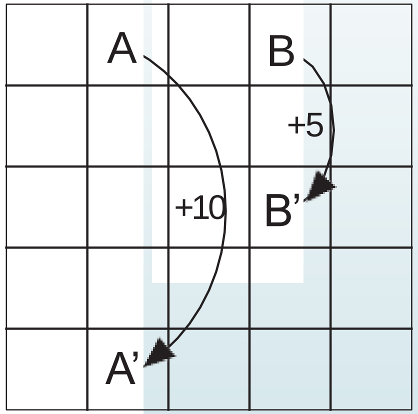
(a)

3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

(b)

What is the value function for the uniform random policy?

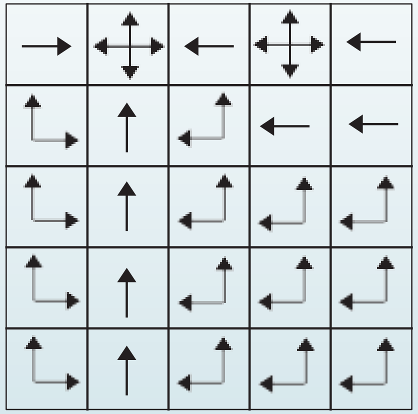
Gridworld Example: Control



a) gridworld

22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

b) *V*



c) ⇡

What is the optimal value function over all possible policies?

What is the optimal policy?

Course Outline

- Part I: Elementary Reinforcement Learning
  1. Introduction to RL
  2. Markov Decision Processes
  3. Planning by Dynamic Programming
  4. Model-Free Prediction
  5. Model-Free Control
- Part II: Reinforcement Learning in Practice
  1. Value Function Approximation
  2. Policy Gradient Methods
  3. Integrating Learning and Planning
  4. Exploration and Exploitation
  5. Case study - RL in games