00 Introduction

Table of contents

Subject Overview: COMP90054 AI Planning for Autonomy

Teaching Team

- Adrian Pearce (Subject Coordinator)
- Sarita Rosenstock (Subject Coordinator)
- Andrew Chester (Education specialist)
- Grady Fitzpatrick (Education specialist)
- Guang Hu (Tutor)
- Geye Guo (Tutor)
- Muhammad Bilal (Tutor)
- Chao Lei (Tutor)
- Jiajia Song (Tutor)

Weekly Lectures

Short videos covering background knowledge (in place of one 1 hour lecture)

- watch before attending live lecture
- include questions

Live in-person lecture: Question & Answer, polls, and worked examples

Tutorials (Workshops)

Tutorials (workshops) start in week 1 (Friday)

- Tutorial classes start on Friday 6th March in week 1
- Please complete pre-class activities in Ed beforehand, week 1 is here: https://edstem.org/au/courses/32360/lessons/102843/slides/707129

Practice Quizzes (ungraded self-assessment)

Subject Structure

- See Canvas (LMS): Home
- Also see Subject Overview

Assignments

- Individual project: Search + modelling a planning problem
- Mid-semester test: Short quiz
- Individual project: Reinforcement Learning

Marking: Ungrading + standards

Assessment flexibility: Tokens

Token Participation Streak = Engagement Flexibility

Streak = consecutive weekly lecture participation.

- You can “spend” tokens for assignment extensions (1 token = ½ day extension)
- If you miss a live lecture you can freeze the streak (1 token)

Your token budget = 2 tokens + max streak (weekly PollEv participation)

- We encourage agency and responsibility,
- are using less rigid policies, and
- focus on learning instead of point-chasing (this is the ungrading approach).

Tokens adds structure and flexibility

- Clear, fair, and manageable, and
- no need to justify every request.

Things You Should Know to Enjoy this Subject

1. Algorithms such as Dynamic Programming
2. Basic Set Theory and Propositional Logic
3. Probabilistic Theory such as Conditional Probabilities
4. Python - start by doing a tutorial to refresh your knowledge

Importantly, you need to stay up to date reviewing and understanding the content, as most lectures build on previous knowledge.

Introduction to Python

You need to be confident programming in Python. Test your skills, refresh your knowledge or get up to speed doing these tutorials:

- Interactive Basics Tutorial: https://www.learnpython.org/
- Python Basics Tutorial: https://inst.eecs.berkeley.edu/~cs188/su24/projects/proj0/python-basics/
- CodeAcademy free course: https://www.codecademy.com/learn/learn-python-3
- Python Practice Puzzles (on Ed): https://edstem.org/au/courses/32360/lessons

Introduction to probability theory

The content around reinforcement learning requires some understanding of basic probability theory.

- Essentially just the main three axioms of probability, conditional probability and the product rule.

If you do not have this or require a brush up, please consult the following

https://gibberblot.github.io/rl-notes/appendix/intro-to-probability-theory.html

Consultation

If you don’t understand…or have a question…

→ Use Ed Discussion forum: COMP90054 Semester 1

- ED is your primary consultation platform
- Please be active in answering as well as posting questions!

Readings

COMP90054 Reinforcement Learning page - includes readings for each module and slides from live lectures

- Available as html book at: https://comp90054.github.io/reinforcement-learning/

Primary Textbooks

Classical Planning

- *A Concise Introduction to Models and Methods for Automated Planning*, Hector Geffner and Blai Bonet, Springer Nature (2013) - available via institutional (University of Melbourne) login
- *An Introduction to the Planning Domain Definition Langauge*, Patrik Haslum, Nir Lipovetzky, Daniele Magazzeni, and Christian Muise, Springer (2019) - available via institutional login

Reinforcement Learning

- *Reinforcement Learning, An Introduction* by Richard Sutton and Andrew Barto, Second Edition MIT Press (2020) (available for download at http://www.incompleteideas.net/book/RLbook2020.pdf)

Additional Text

Introduction to Search

- *Artificial Intelligence: A Modern Approach*, Stuart Russell and Peter Norvig, 2020 - In particular Chapter 3 & 4 on search - available via institutional login

Learning & Planning

Learning & Planning

Two fundamental problems in sequential decision making

Planning:

- A model of the environment is known
- The agent performs computations with its model (without any external interaction)
- The agent improves its behaviour (policy) through search, deliberation, reasoning, and introspection

Reinforcement Learning (RL):

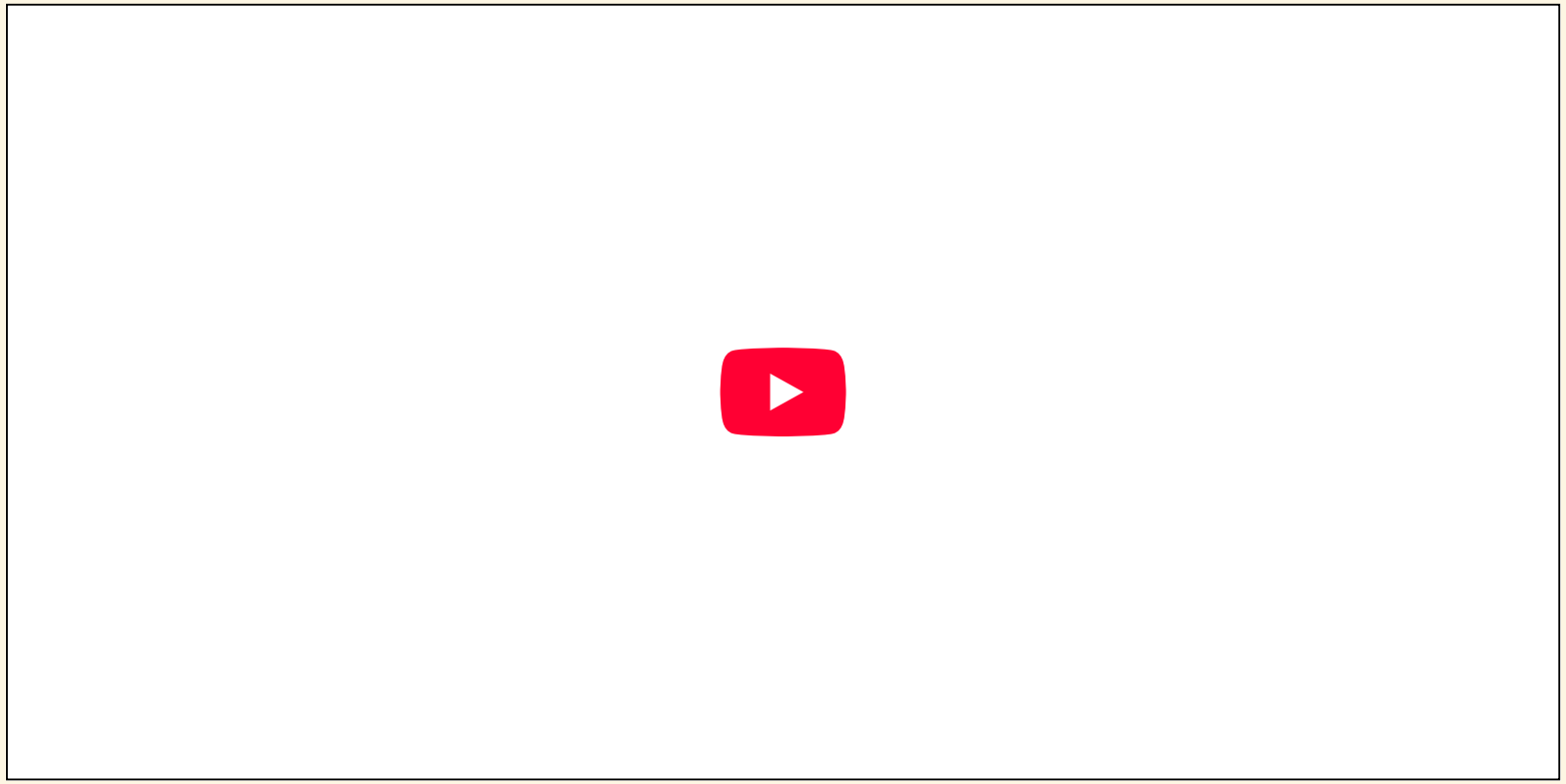
- The environment is initially unknown
- The agent interacts with the environment
- The agent improves its policy (behaviour)

Examples

- Making a humanoid robot walk
- Fine tuning LLMs using human/AI feedback
- Optimising operating system routines
- Controlling a power station
- Managing an investment portfolio

- Competing in the International Mathematical Olympiad
- Multi-agent pathfinding by robots in a warehouse

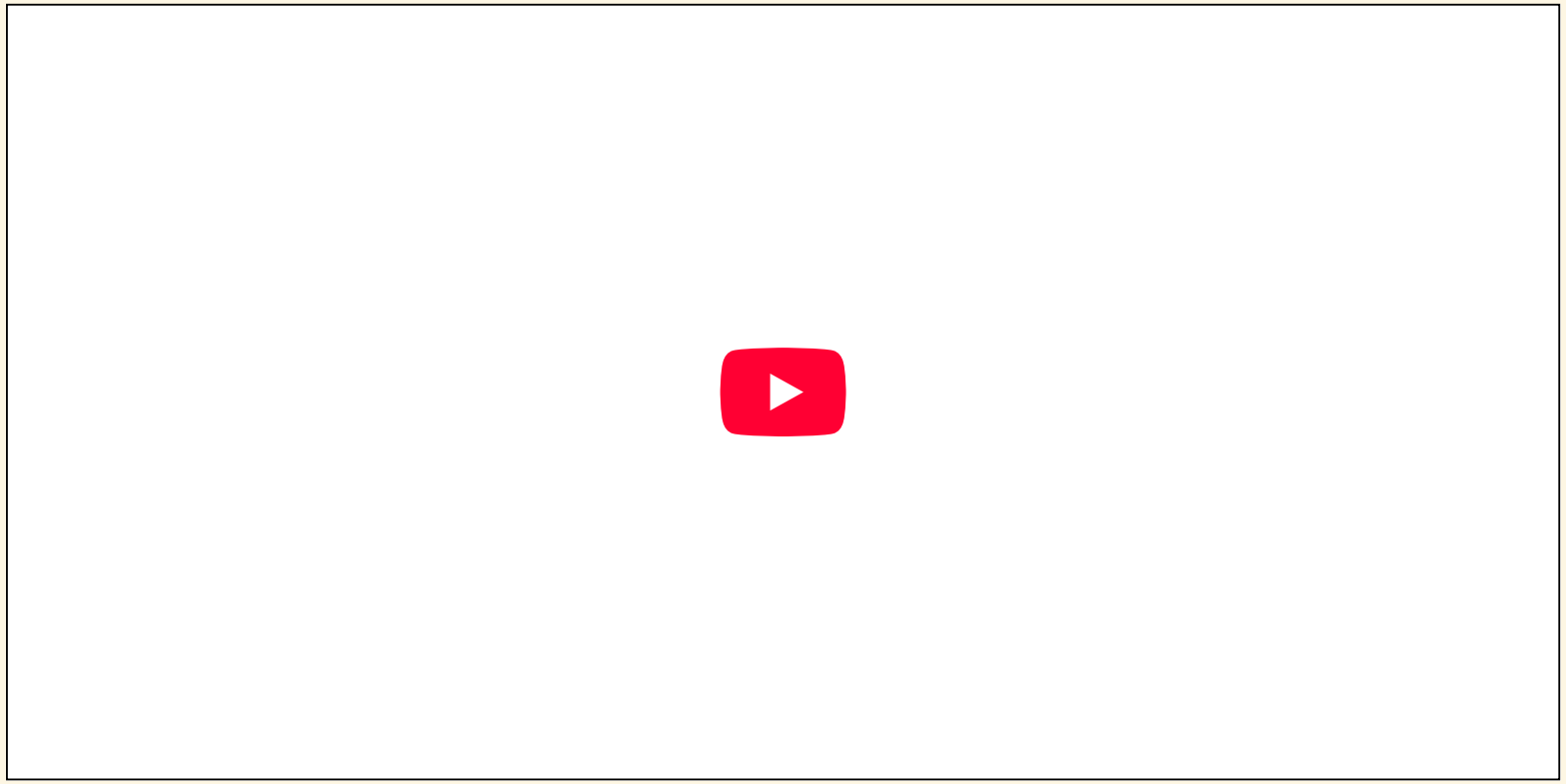
Example: Make a humanoid robot walk



Atlas demonstrates policies using RL based on human motion capture and animation - *Boston Dynamics 2025*

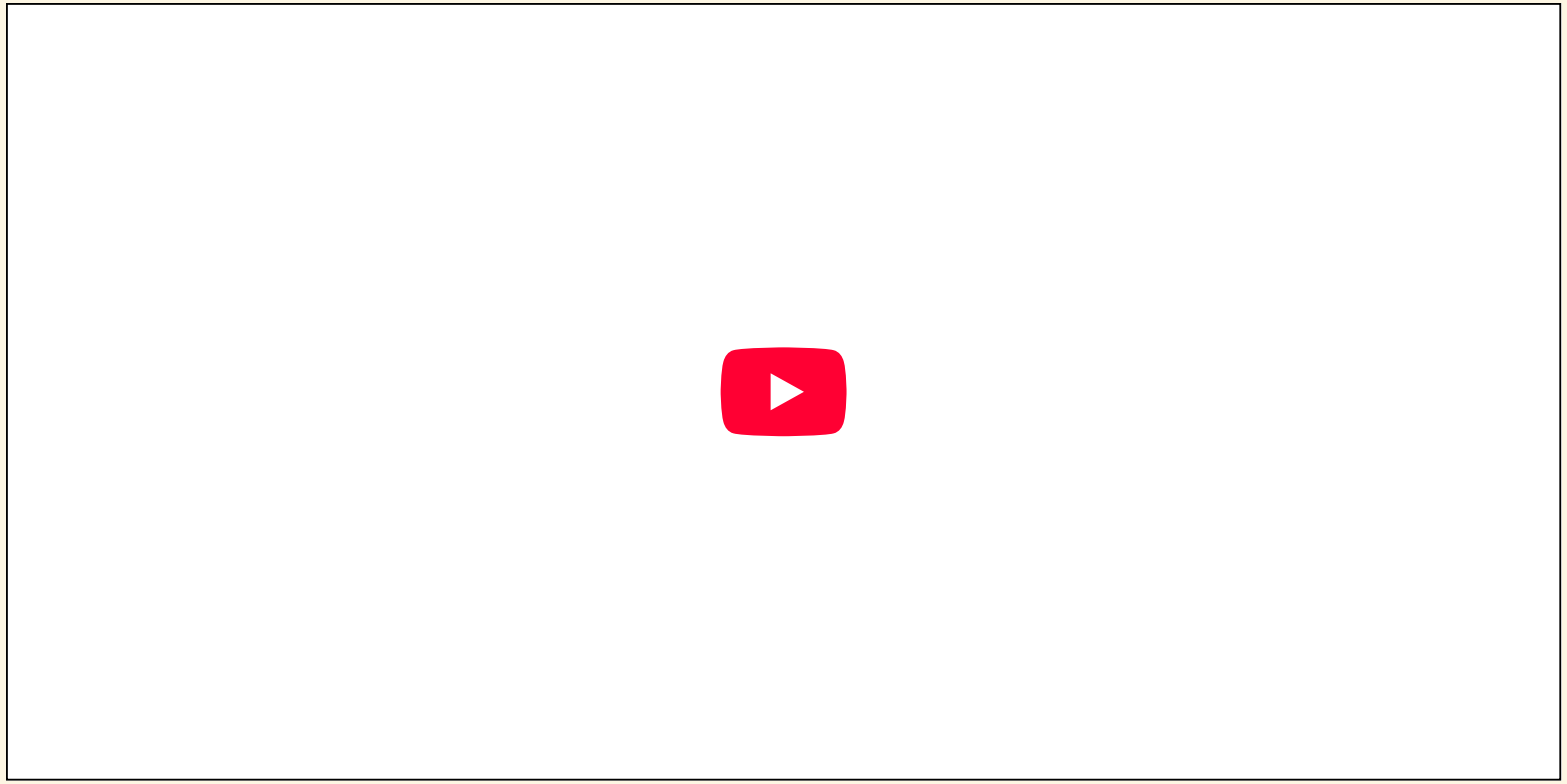
youtube.com/watch?v=I44_zbEwz_w

Example: Fine tuning LLMs using human/AI feedback



Proximal policy update (PPO) used by *ChatGPT 3.5 & Agentic AI - Chat GPT 'Operator' & Claude's 'Computer Use'* modes.

youtube.com/watch?v=VPRSBzXzavo



Group relative policy optimisation (GRPO) (more stable than PPO) used in *DeepSeek's* latest models.

youtube.com/watch?v=xT4jxQUl0X8

Example: Optimising operating system routines

2023, Daniel J. Mankowitz, et al. Faster sorting algorithms discovered using deep reinforcement learning, *Nature*, Vol 618, pp. 257-273

DeepMind's AlphaDev, a deep reinforcement learning agent, has discovered faster sorting algorithms, outperforming previously known human benchmarks.

- The sort routine is called up to a trillion times a day worldwide
- These newly discovered algorithms have already been integrated into the *LLVM standard C++ sort library*.

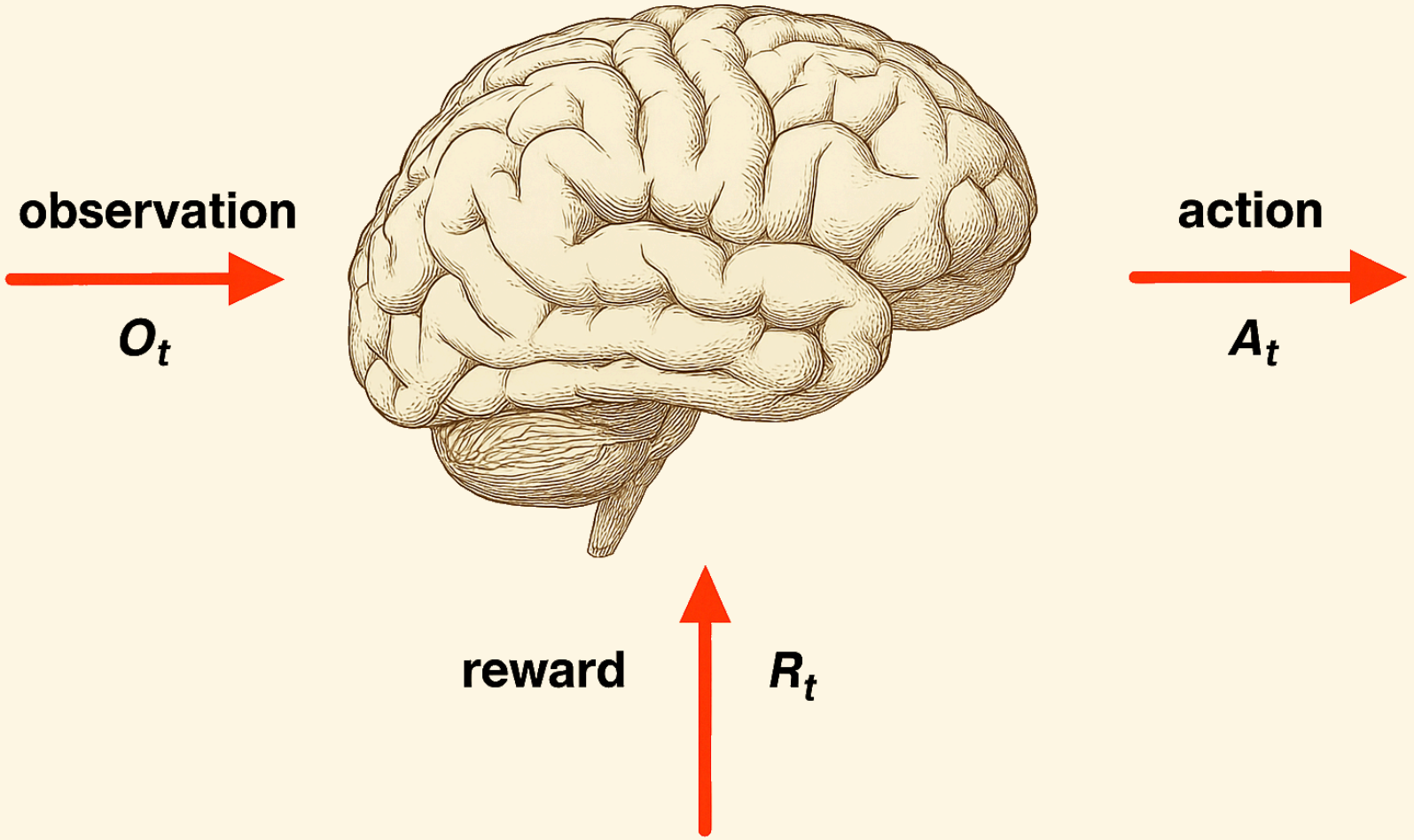
Example: Compete in International Mathematical Olympiad

https://deepmind.google/discover/blog/ai-solves-imo-problems-at-silver-medal-level/

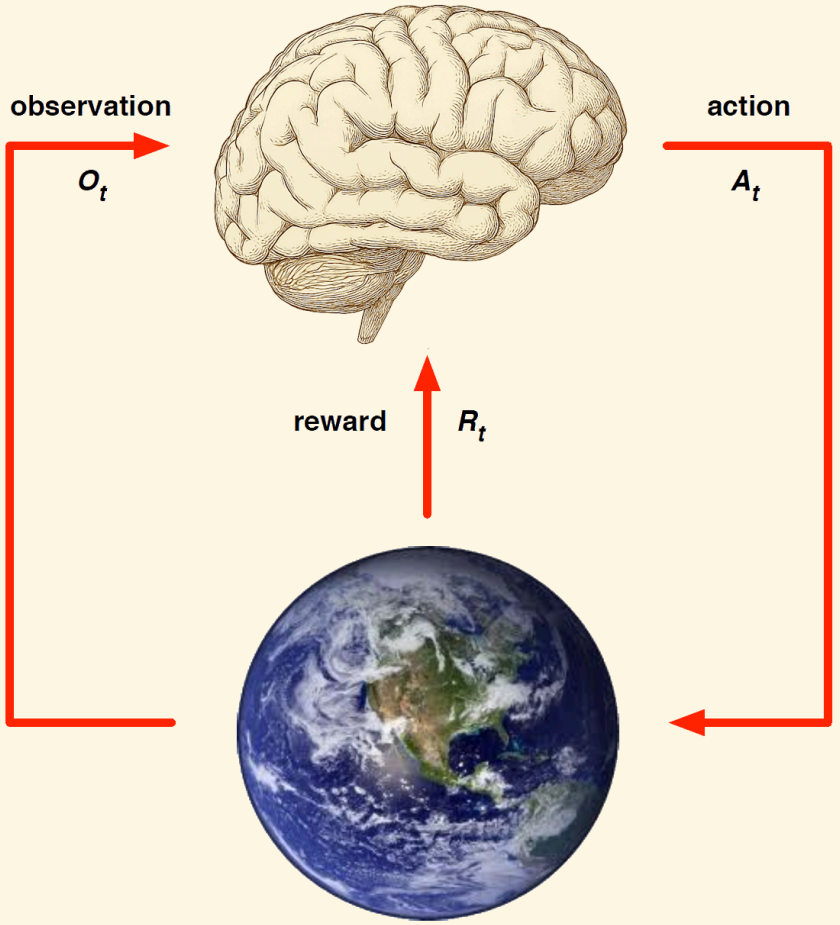
- AlphaProof achieved a sliver medal using a new reinforcement-learning based system for formal maths reasoning (see link above).

Agent model

Agent



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 sequence of observations, actions, rewards

$$H_t = O_0, R_0, A_0, \ldots A_{t-1}, O_t, R_t$$

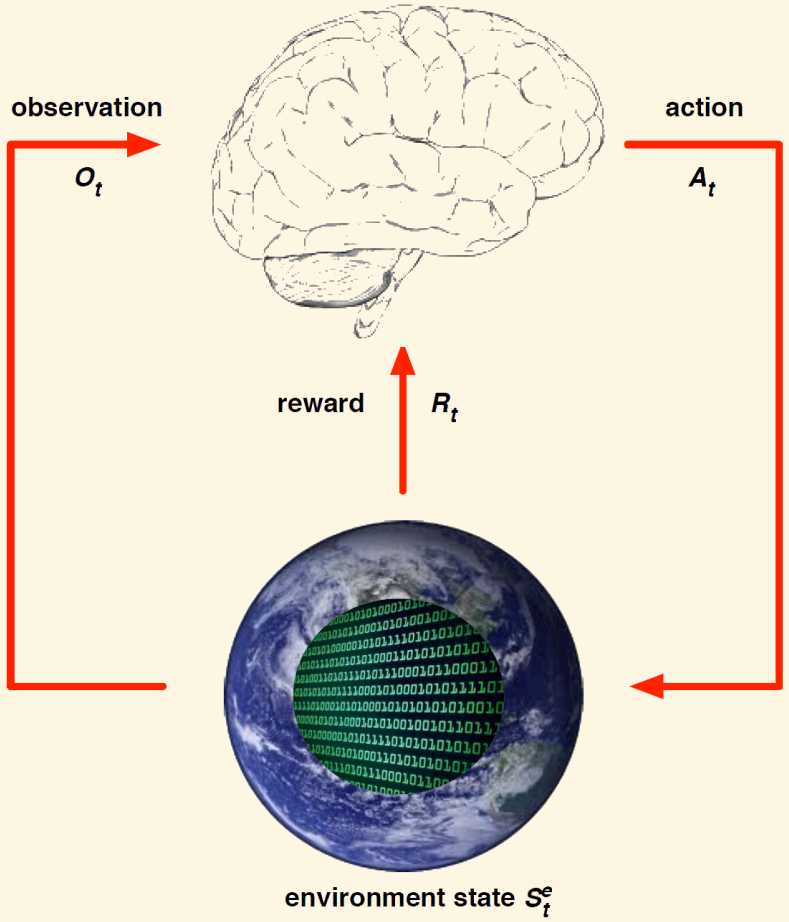
- i.e. the stream of a robot's actions, observations and rewards up to time t

What happens next depends on the history:

- The agent selects actions, and
- the environment selects observations/rewards.

State is the information used to determine what happens next.

Environment State



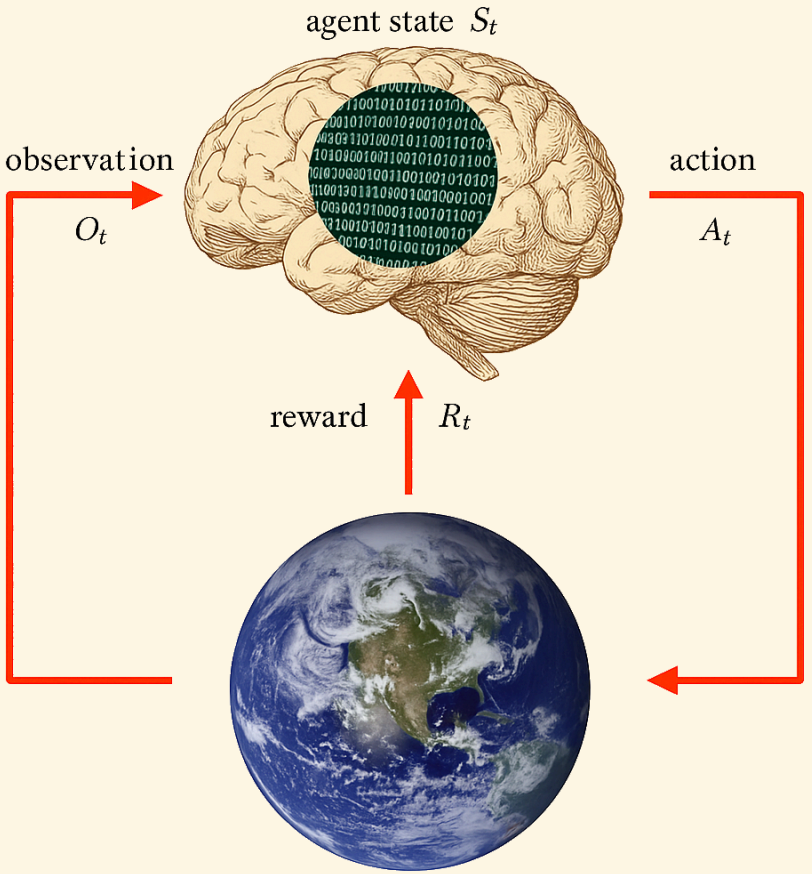
The environment state S_t^e is the environment's private representation

- i.e. environment uses to determine 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 info

Agent State

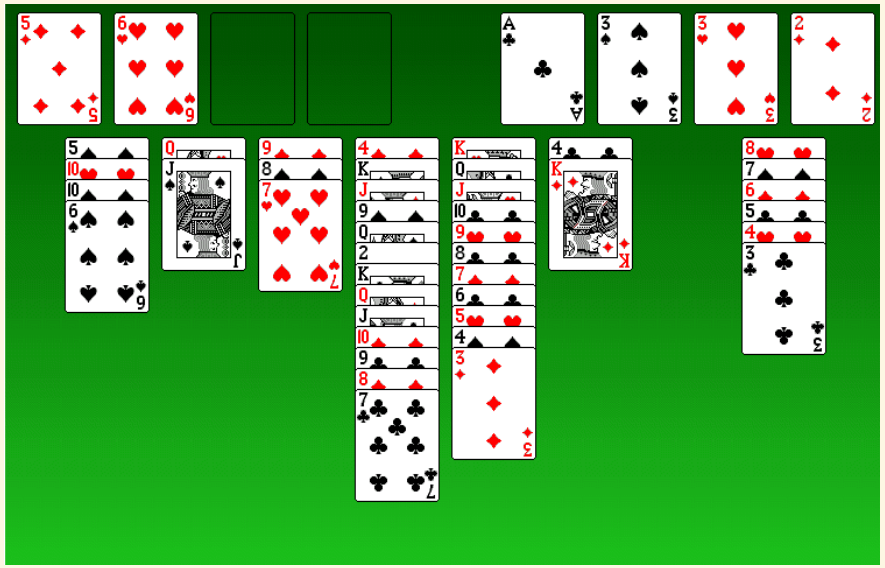


The agent state S_t^a is the agent's *internal* representation

- i.e. information the agent uses to pick the next action
- i.e. information used by reinforcement learning algorithms

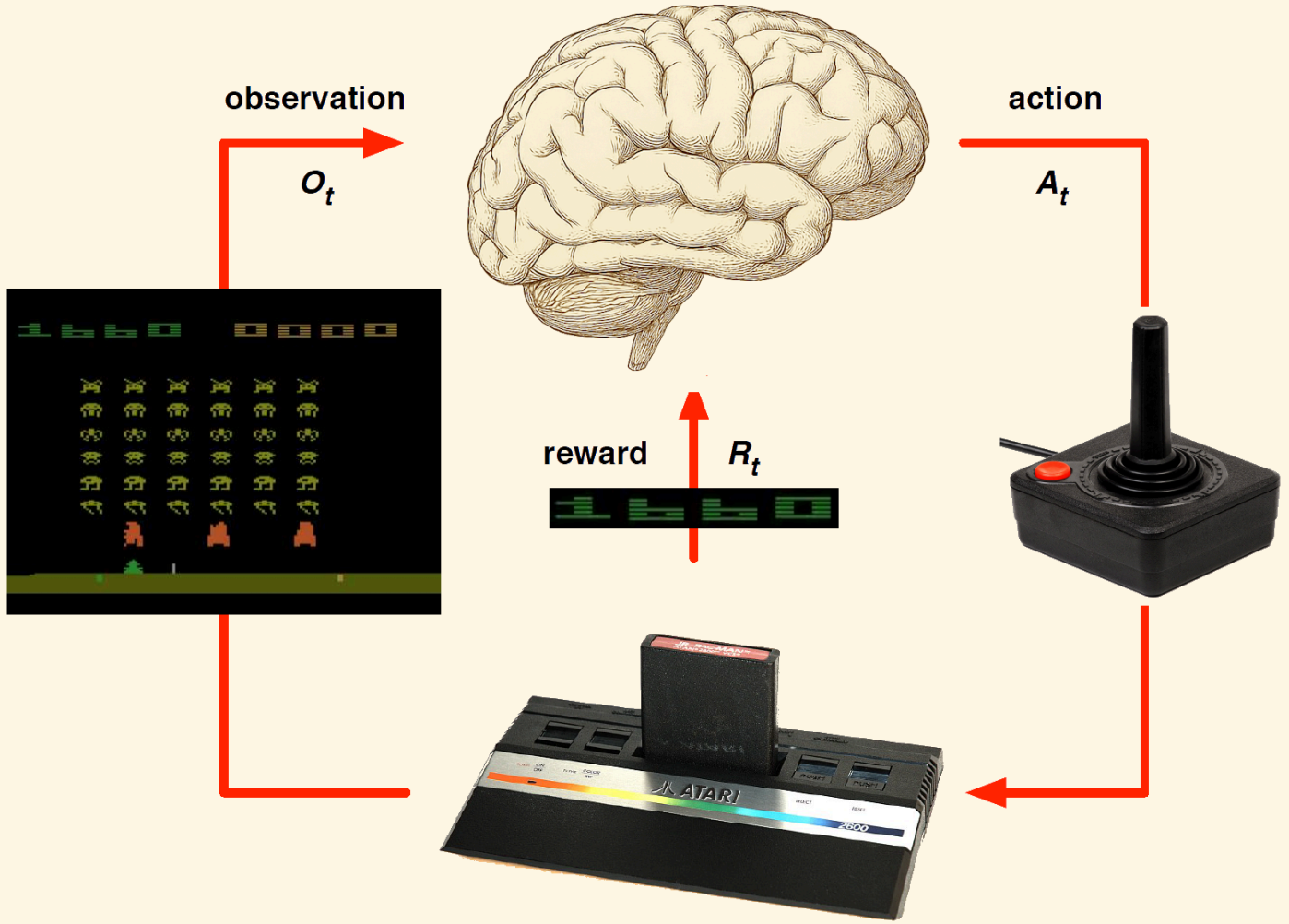
It can be any function of history: $S_t^a = f(H_t)$

Patience Example: Classical Planning



- States: Card positions (position Jspades=Qhearts).
- Actions: Card moves (move Jspades Qhearts freecell4).
- Initial state: Start configuration.
- Goal (reward) states: All cards ‘home’.
- Solution: Card moves solving this game.

Atari Example



Atari Example: Planning

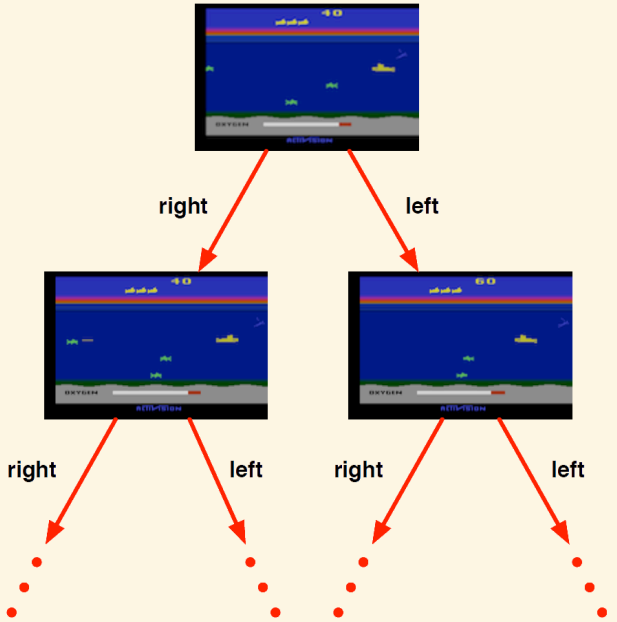
If rules of the game are *known*, can plan

- perfect model inside agent's brain

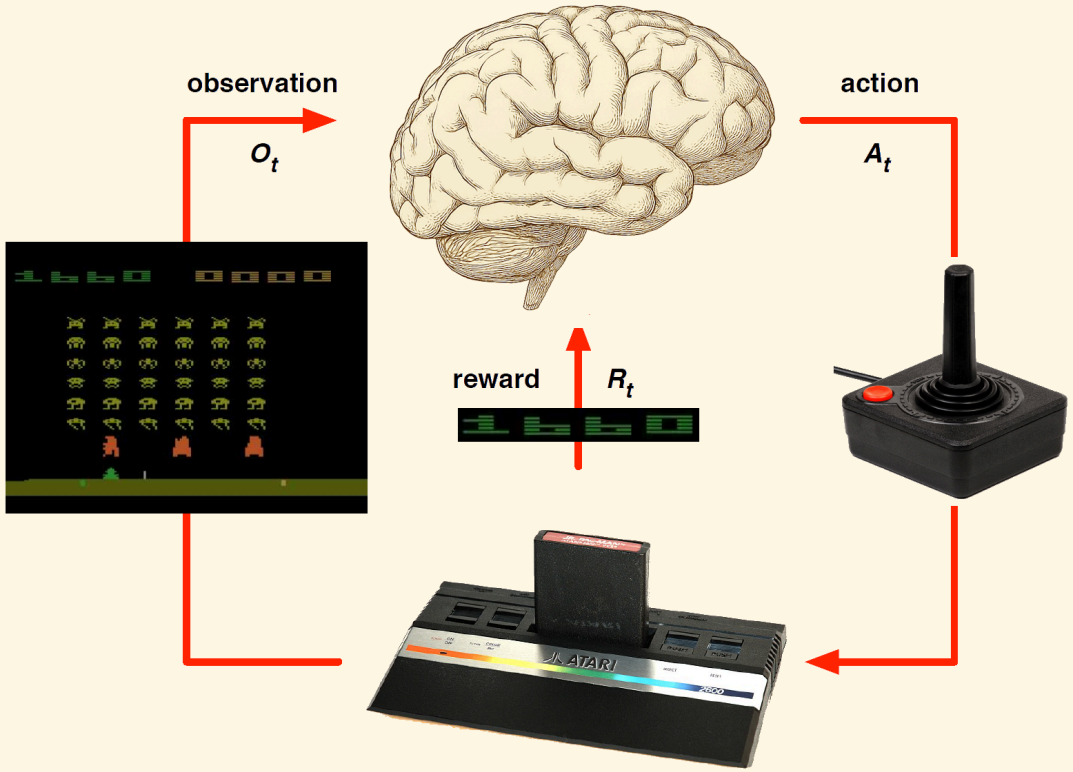
If I take action a from state s:

- what would the next state s' be?
- what would the score be?

Plan ahead for reward - stay alive, maximise score!



Atari Example: Learning



- If rules of the game are *unknown*, need to learn
- Pick joystick actions, only see pixels & scores
- Learn directly from interactive game-play

Rewards & Goals

Rewards & Goals

- 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.

Example of Rewards

- Make a humanoid robot walk
  - -ve reward for falling
  - +ve reward for forward motion
- Optimise sort routine in an operating system
  - -ve reward for execution time
  - +ve reward for throughput

- Control plasma in a stellarator in a fusion power station
  - +ve reward for containment of plasma
  - -ve reward for plasma crashing
- Control inventory in a warehouse
  - -ve reward for stock-out penalty (lost sales)
  - -ve reward for holding costs (inventory)
  - +ve reward for sales revenue

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)
- Re-stocking warehouse (might prevent a stock-outs in days or weeks)

Markov State

Markov State

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

Definition: Markov State

A state S_t is Markov if and only if

$$\mathbb{P}[S_{t+1}|S_t] = \mathbb{P}[S_{t+1} \mid S_0, \ldots, S_t]$$

- Markov states are key to both planning and learning

Markov State (continued)

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

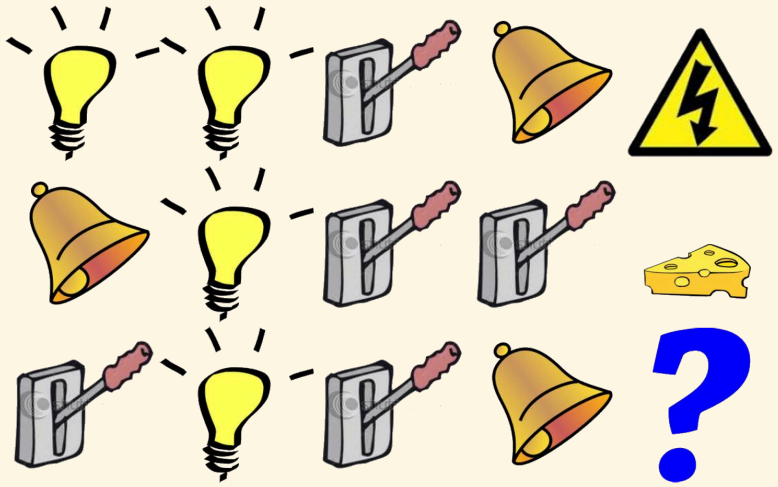
- The environment state S_t^e is Markov
- The history H_t is Markov

How is a Markov state dependent on time - in terms of the past, present and future?

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

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

Rat Example



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

Shock

- What if agent state = counts for lights, bells and levers?

Cheese

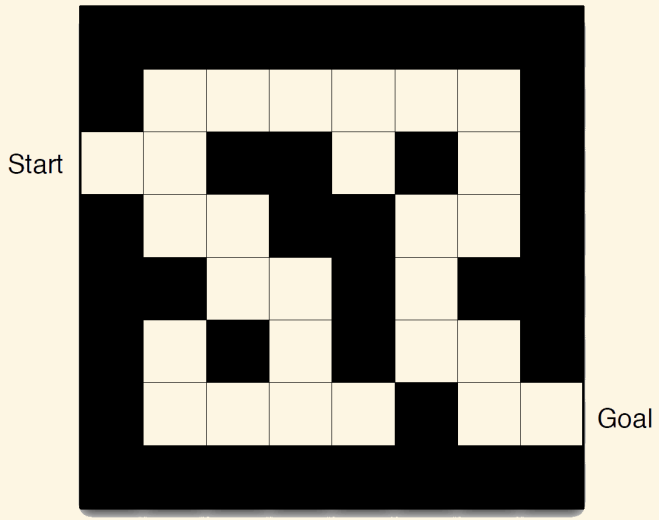
- What if agent state = complete sequence?

?

Components of an Agent

What are the components of a Markov planning and learning agent?

Example to help you think about question:



Components of an Agent

What are the components of a Markov planning and learning agent?

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

- Policy: agent's behaviour function
- Value function: how good is each state and/or action (rewards/goals)
- Model: agent's representation of the dynamics of the environment (including states, propositions (STRIPS) and/or probabilities (MRPs)

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

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.

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

Model

A model predicts what the environment will do next

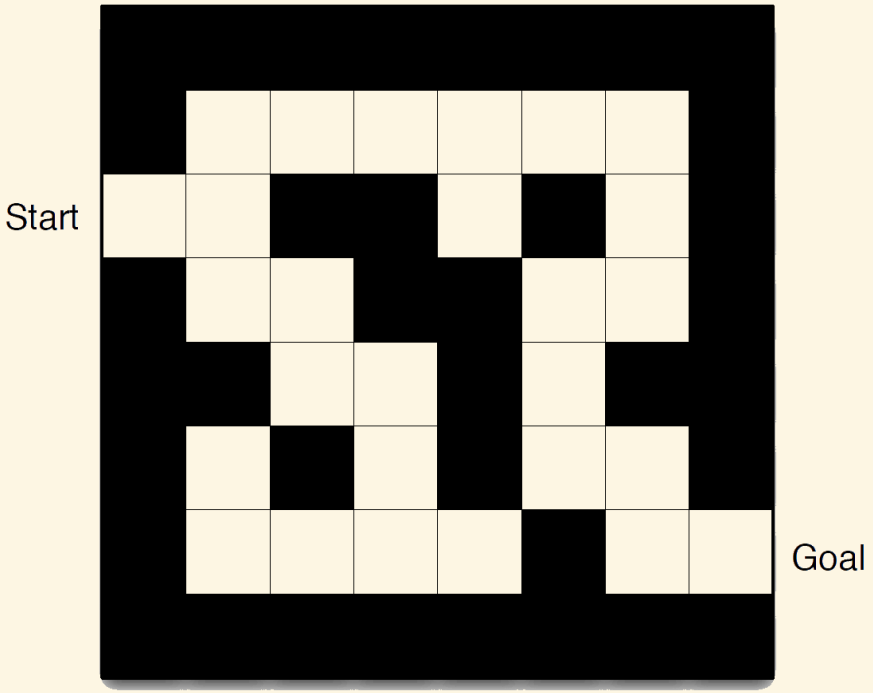
$\mathcal{P}$ predicts the *probability* of the next state

$\mathcal{R}$ predicts the *expectation* of the next *reward*, e.g.

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

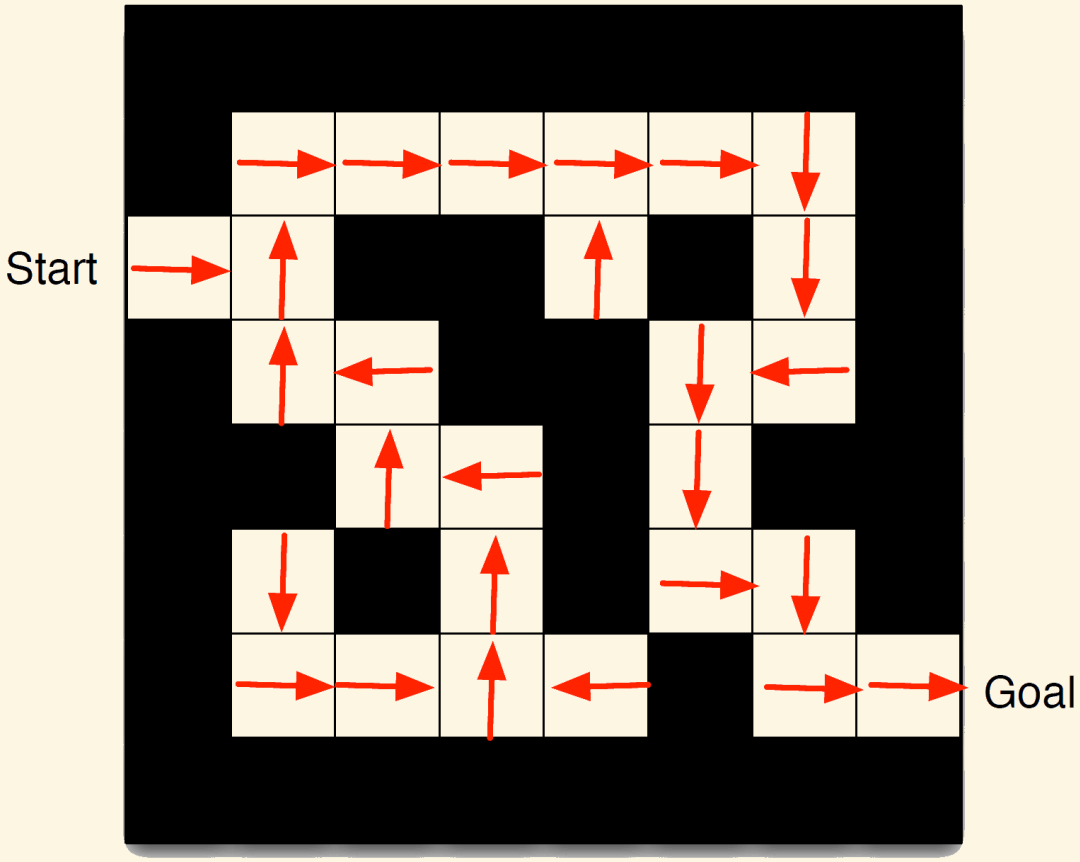
$$\mathcal{R}^a_s = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$$

Maze Example



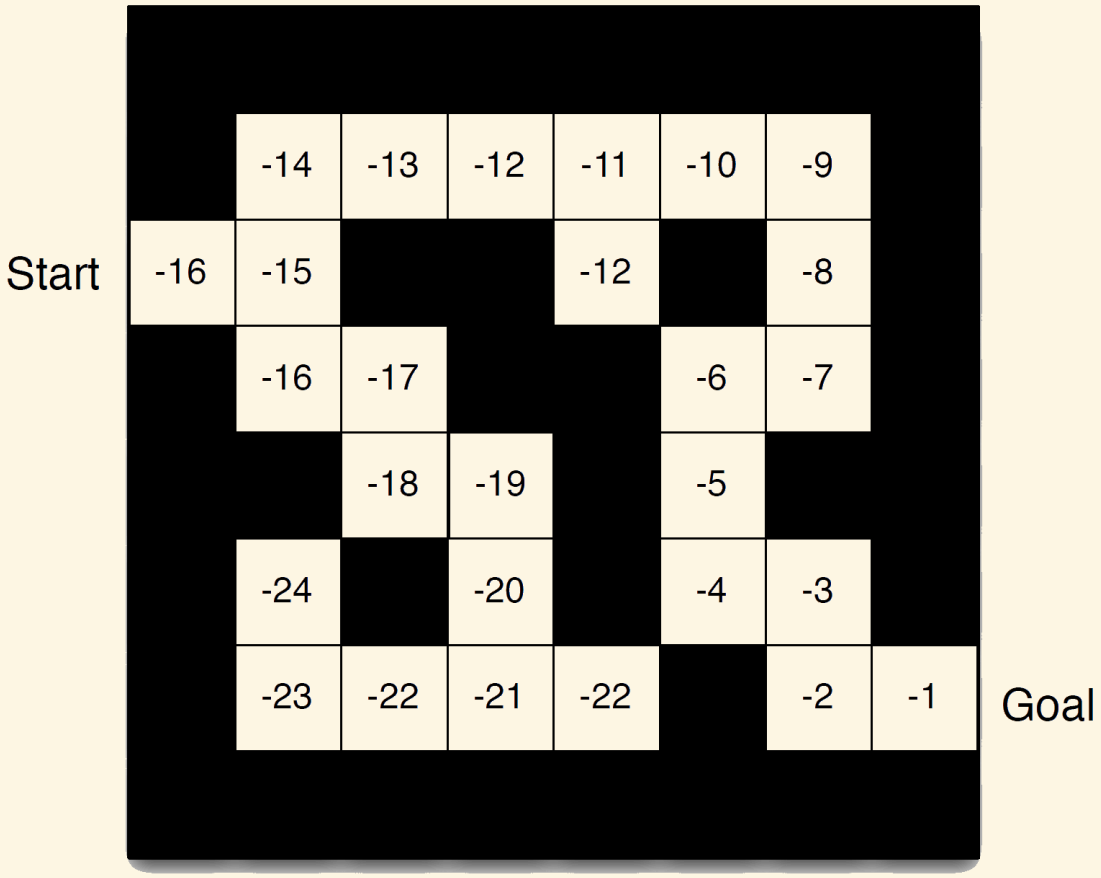
- Rewards: -1 per time-step
- Actions: N, E, S, W
- States: Agent's location

Maze Example: Policy



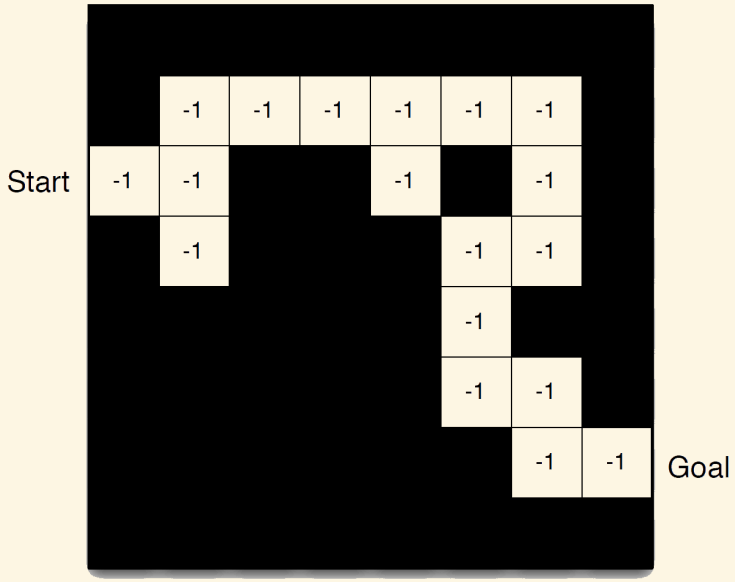
- Arrows represent policy $\pi(s)$ for each state s

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 $\mathcal{P}^a_{ss'}$

Numbers represent immediate reward $\mathcal{R}^a_s$ from each state s (same for all a)

Goals and Rewards

Classical planning typically relies on a model

- The agent aims to synthesise a plan toward achieving a goal
- A plan is a special case of a policy (for one set of initial conditions) toward achieving a goal.
- The goal is a special case of (more general) value achieved from reward.

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 and Exploitation (continued)

- *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
- Gold exploration
  Exploitation Drill core samples at the best known location
  Exploration Drill at a new location

Examples (continued)

- Game Playing
  Exploitation Play the move you believe is best
  Exploration Play an experimental move

Prediction and Control

Prediction: evaluate the future

- Given a best policy

Control: optimise the future

- find the best policy

We need to solve the prediction problem in order to solve the control problem

- i.e. we need evaluate all of our policies in order to work out which is the best one

Characteristics of Classical Planning & Reinforcement Learning

It is helpful to think about the differences:

What makes planning different from traditional machine learning paradigms?

- Sequence matters, i.e. it involves non-i.i.d. (independent and identically distributed) data
- …contemporary deep learning utilises sequence too, through e.g. attention based transformers

What makes reinforcement learning different from other machine learning paradigms?

- There is no supervisor, only a reward signal
- Feedback is delayed, not instantaneous
- …contemporary transformers utilise RL techniques (ChatGPT, DeepSeek, etc.)

What makes reinforcement learning different from planning?

- The outcomes of actions are non-deterministic
- Uses probabilistic representation
- Relies on (more general) rewards instead of goals