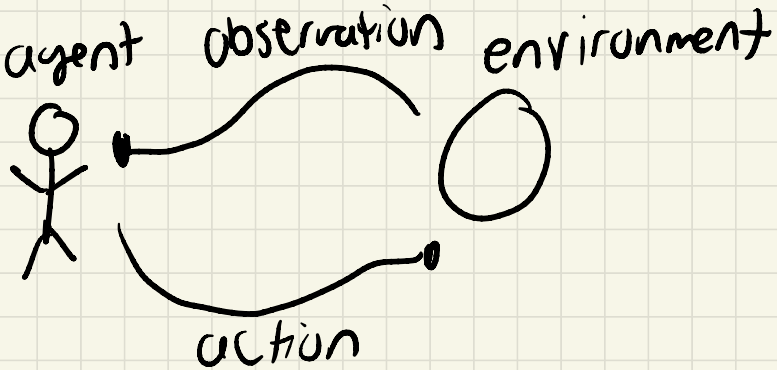
Intelligence := to be able to learn to make decisions to achieve goals

what is RL?

- People/animals learn by interacting with env.
  - ↳ active
  - ↳ sequential
- we are goal-directed
- we can learn without examples of optimal behaviour. → we optimise reward signal



Goal: optimise sum of rewards, through repeated interaction

RL is based on the reward hypothesis!

Any goal can be formalized as the outcome of maximizing a cumulative reward.

Examples of RL

- Fly helicopter
- Manage investment portfolio
- Control power station
- Make a robot walk
- Play video or board games.

If the goal is to learn via interaction, it is RL.

2 distinct reasons to learn: RL can do both.

1) Find solutions

2) Adapt online, deal with unforeseen circumstances.

RL is

- science and framework of learning to make decisions from interaction.

↳ need to think about
- time
- long term consequences of action
- actively gathering experience
- predicting the future
- dealing with uncertainty

- Huge potential scope
- A formalisation of the AI problem

formalising the RL problem

- At each step t, the agent:
  - Receives observation O_t (and reward R_t)
  - Executes action A_t
- the environment:
  - Receives action A_t
  - Emits observation O_{t+1} (reward R_{t+1})

} could be continuous time.

Rewards → Scalar feedback signal

- Indicates how well agent is doing at step t -

- agent goal is to maximize cumulative reward
  - ↳ return. (future reward)

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \dots$$

Values

- Expected cumulative reward, from a state s, the value → goal: maximise this.

$$v(s) = \mathbb{E}[G_t \mid S_t = s]$$

$$= \mathbb{E}[R_{t+1} + R_{t+2} + \dots \mid S_t = s]$$

- the value depends on action agent take
- Rewards and values define utility of states and action
- Returns can be defined recursively

$$G_t = R_{t+1} + G_{t+1}$$

$$v(s) = \mathbb{E}[R_{t+1} + v(S_{t+1}) \mid S_t = s]$$

- Goal: Select actions to maximise value.

Mapping from states to actions := policy

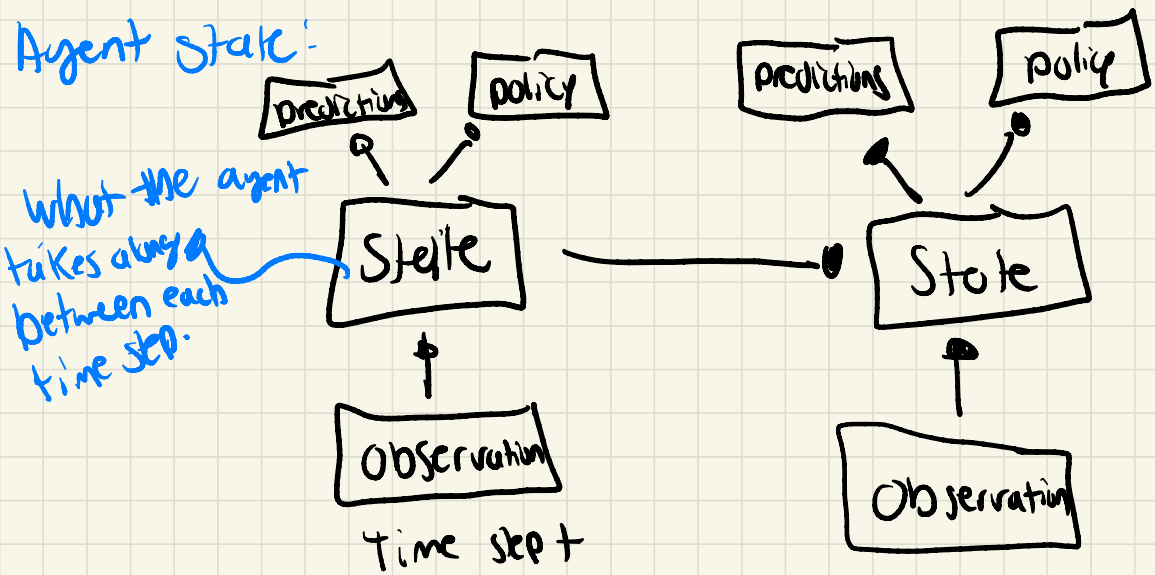
$$q(s,a) = \mathbb{E}[G_t \mid S_t = s, A_t = a]$$
$$= \mathbb{E}[R_{t+1} + R_{t+2} + \dots \mid S_t = s, A_t = a]$$

Core concepts!

- Environment
- Reward
- Agent
  - agent state
  - policy
  - value function estimate?
  - model?

Inside the agent: the Agent State

Agent State:



The history := full sequence of observations/actions/rewards

$$H_t = O_0, A_0, R_1, O_1, \ldots, O_{t-1}, A_{t-1}, R_t, O_t$$

↳ used to construct the agent state S_t

Full observability := Agent sees full environment state (we could use environment state as state)

$S_t = O_t$ = environment state.

Markov decision processes (MDPs)

Decision process is Markov if:

$$p(r_{t+1} | s_t, a_t) = p(r_{t+1} | H_t, a_t)$$

↳ state contains everything we need to know from history.

→ once state is known, the history may be thrown away.

Typically, s_t is some compression of H_t

Partial observability: observations are not Markovian

Using observation as state would not be Markovian, this is called partially observable Markov decision process (POMDP)

Agent State ~ actions depend on state

↳ function of history

Generally: $S_{t+1} = u(S_t, A_t, R_{t+1}, O_{t+1})$

↳ state update function

To deal with partial observability, agent can construct suitable state representations

examples:

- Last observation: $S_t = O_t$ ↳ not enough
- Complete history: $S_t = H_t$ ↳ too large
- generic update: $S_t = u(S_{t-1}, A_{t-1}, R_t, O_t)$

↳ how to pick?

Fully markovian agent state not always feasible

more important: state allow good policies and value prediction

Inside the agent: the policy

Policy: defines agent's behavior

↳ map from agent state to action

Deterministic policy: $A = \pi(S)$

Stochastic policy: $\pi(A|S) = P(A|S)$

Inside the agent: Value estimates

the actual value function is the expected return

$$v_\pi(s) = \mathbb{E}[G_t \mid S_t = s, \pi]$$

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

γ: discount factor

$\gamma \in [0, 1]$

- value depends on policy
- used to evaluate the desirability of a state

↳ Trades off importance of immediate vs long-term rewards

Optimal value:

$$v_*(s) = \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a]$$

↳ does not depend on policy

↳ used to create algorithms.

Inside the agent! Model

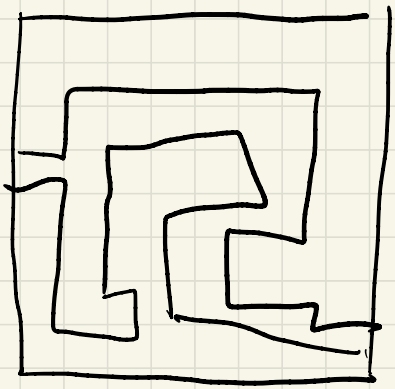
↳ predicts what env. will do

↳ could be stochastics.

$$p(s, a, s') \approx p(S_{t+1} = s' \mid S_t = s, A_t = a)$$

↳ probability of observing a next state given a state and action

Example



Rewards: -1 per time step

Actions: N, E, S, W

States: Agent location

Agent Categories

1) Value Based
   - No Policy
   - Value Function

2) Policy Based
   - Policy
   - No value Function

3) Actor Critic
   - Policy
   - Value function

- Model Free := No model
- Model Based := - optionally policy and/or value function
  - model

All components are functions

- Policies $\pi : S \to A$
- Value functions $v : S \to \mathbb{R}$
- models: $m : S \to S$ and/or $r : S \to \mathbb{R}$
- state update: $u : S \times O \to S$

Can use neural networks and use deep learning to learn