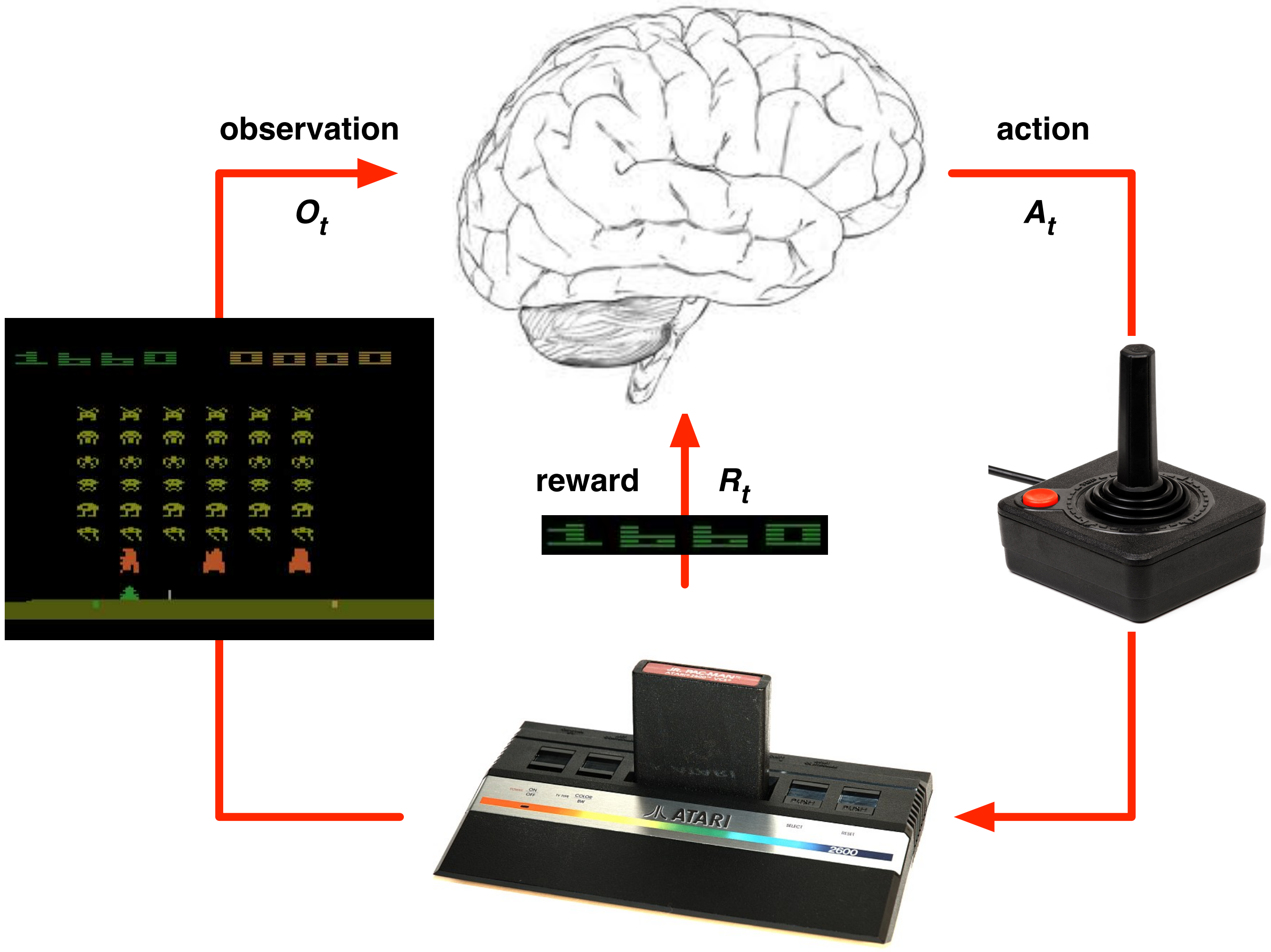
Introduction to Reinforcement Learning

Ali Gooya

Reference for slides:

David Silver (Deep Mind)

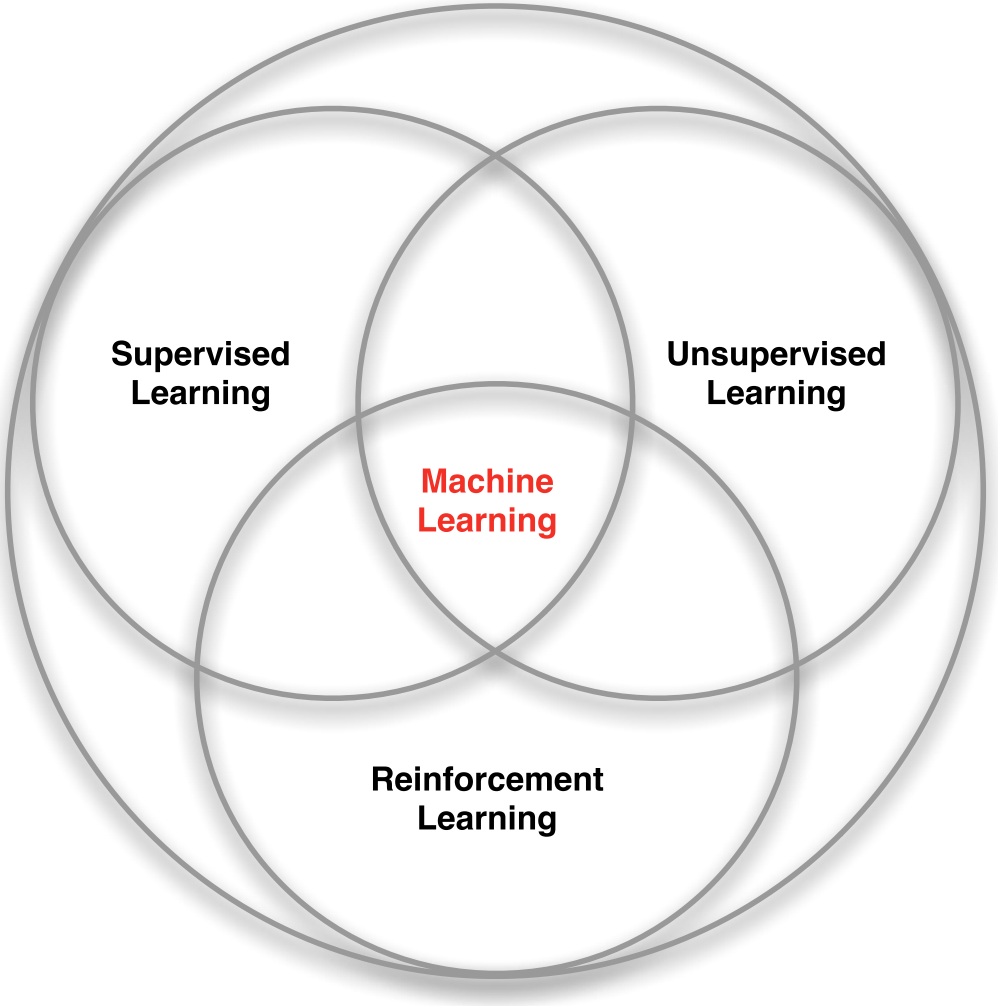
Atari example: RL



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

Characteristics of RL

- What makes RL different from other forms of ML?
- No examples of right actions – different from Supervised Learning
- Reward signal provides some form of information – different from Unsupervised Learning.
- Time matters, sequential data
- Agent action → data to be seen in the future.



Reward

- 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

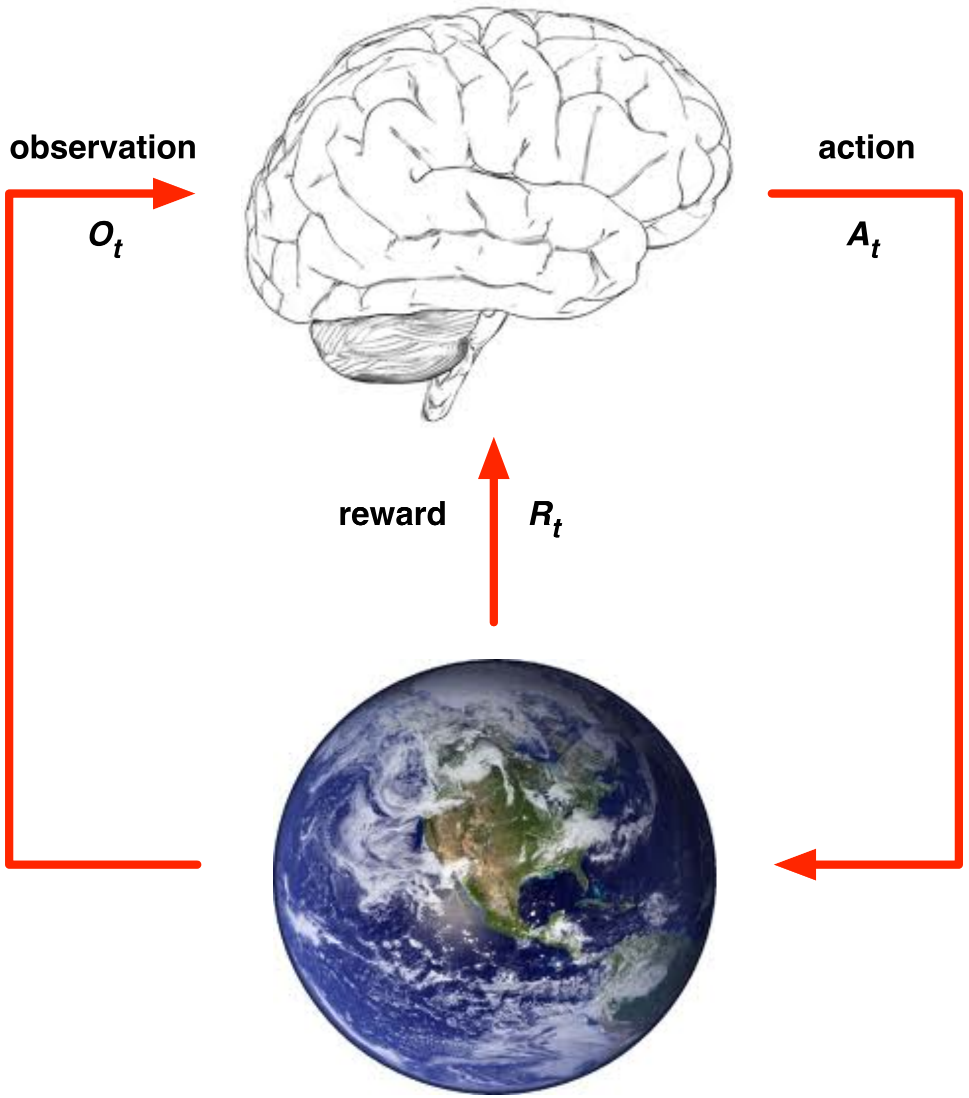
Reinforcement learning is based on the reward hypothesis

Definition (Reward Hypothesis)

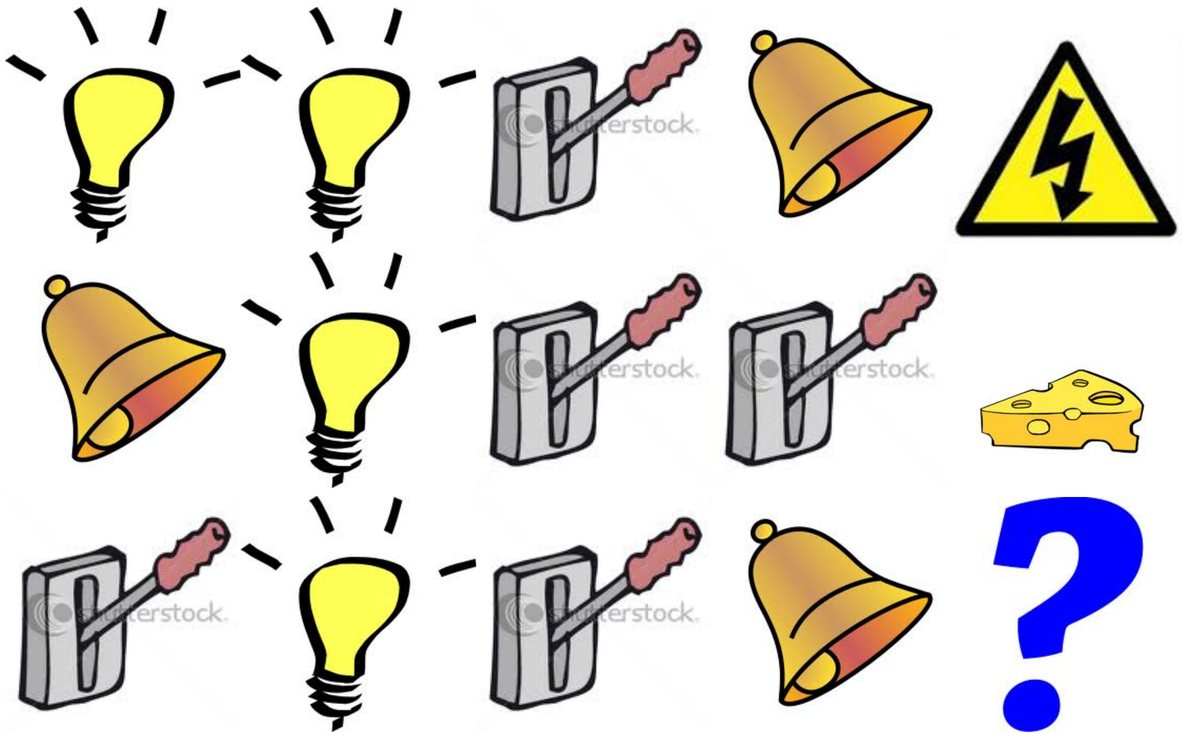
All goals can be described by the maximisation of expected cumulative reward

History and State

- History: $H_t = O_1, R_1, A_1, \ldots, A_{t-1}, O_t, R_t$
- State (S_t): Useful information from history
- Generally speaking, agent and environment can have their own states.
- Environment uses $S_t^e = g(H_t)$ to determine R_t and O_t
- Agent uses $S_t^a = f(H_t)$ to determine A_t



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?

Observability of environments

- Fully observable environment: agent directly observes the environment's state

$$O_t = S_t^a = S_t^e$$

- Formally this is a Markov Decision Process (MDP) - which is simplest form of RL.
- Partial observable environment (POMDP)
  - Robot with a camera vision isn't told its absolute location – no GPS
  - Trading agent only observes the current prices
- In POMDP, Agent must construct its own state, for example: $S_t^a = H_t$

Major components of an RL agent

- An RL agent may include one or more of these components:
  - Policy: How agent determines actions from state?
  - Value function: How good a state or an action is?
  - 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: $\pi(a|s) = \mathbb{P}[A_t = a|S_t = s]$

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.

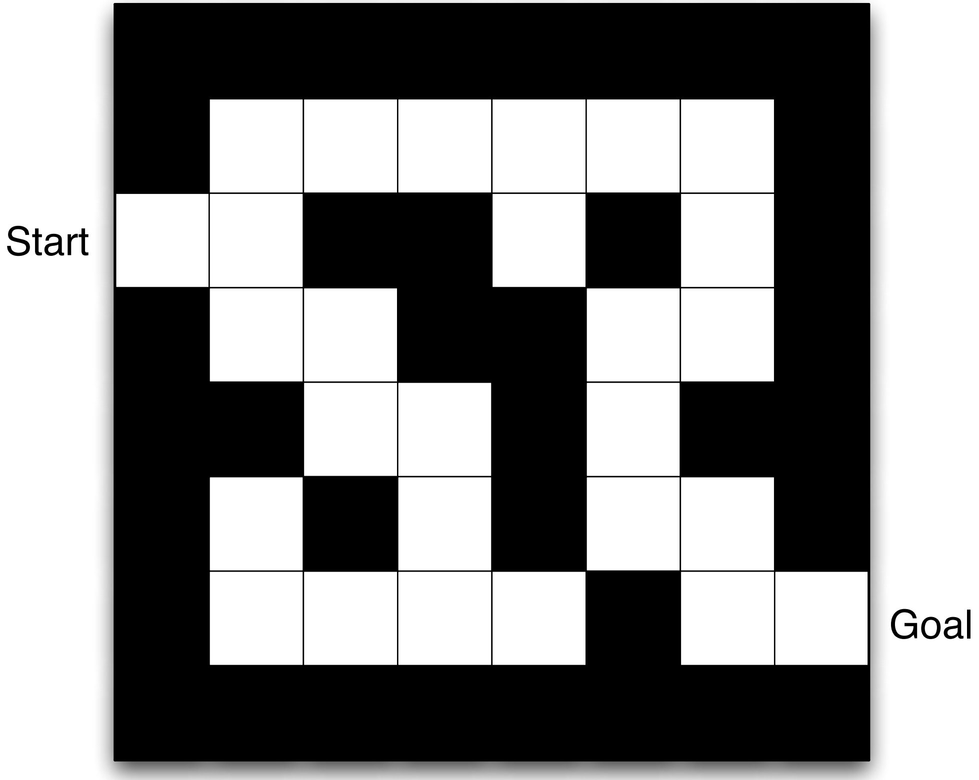
$$v_\pi(s) = \mathbb{E}_\pi \left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \mid S_t = s \right]$$

Model

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

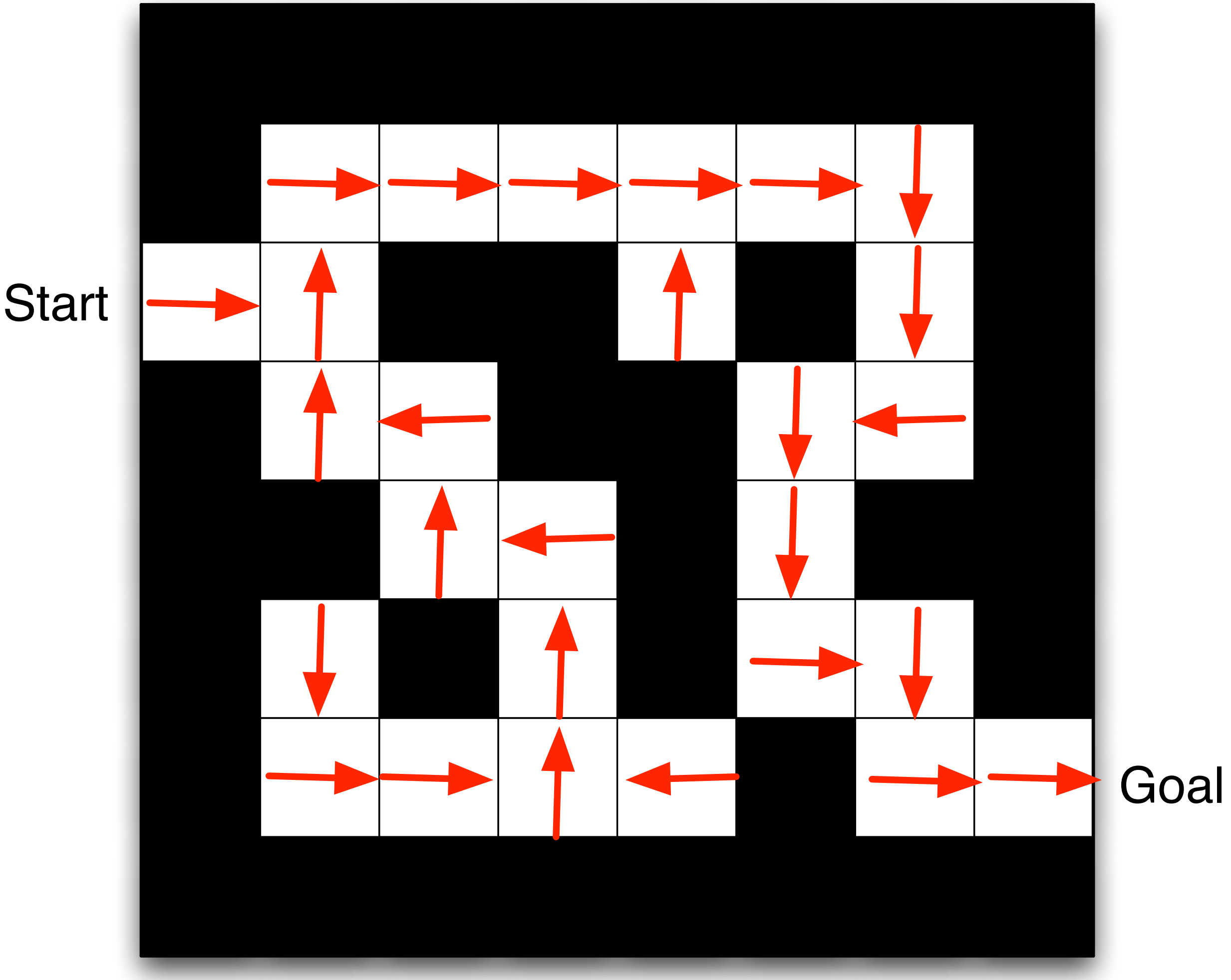
$$\mathcal{P}^a_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$$
$$\mathcal{R}^a_s = \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right]$$

Maze Example



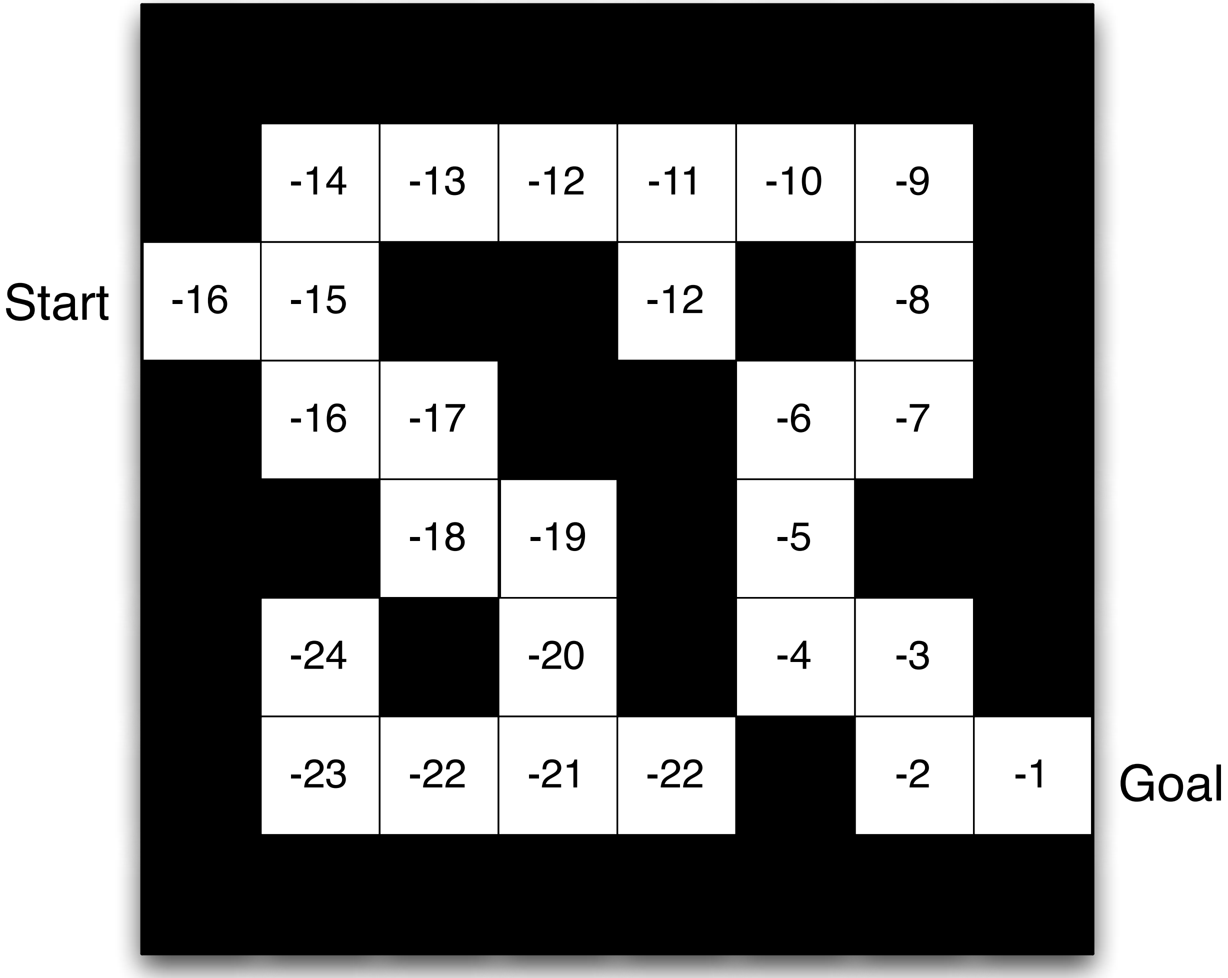
- Rewards: -1 per time-step
- Actions: N, E, S, W
- States: Agent's location

Maze example: Policy



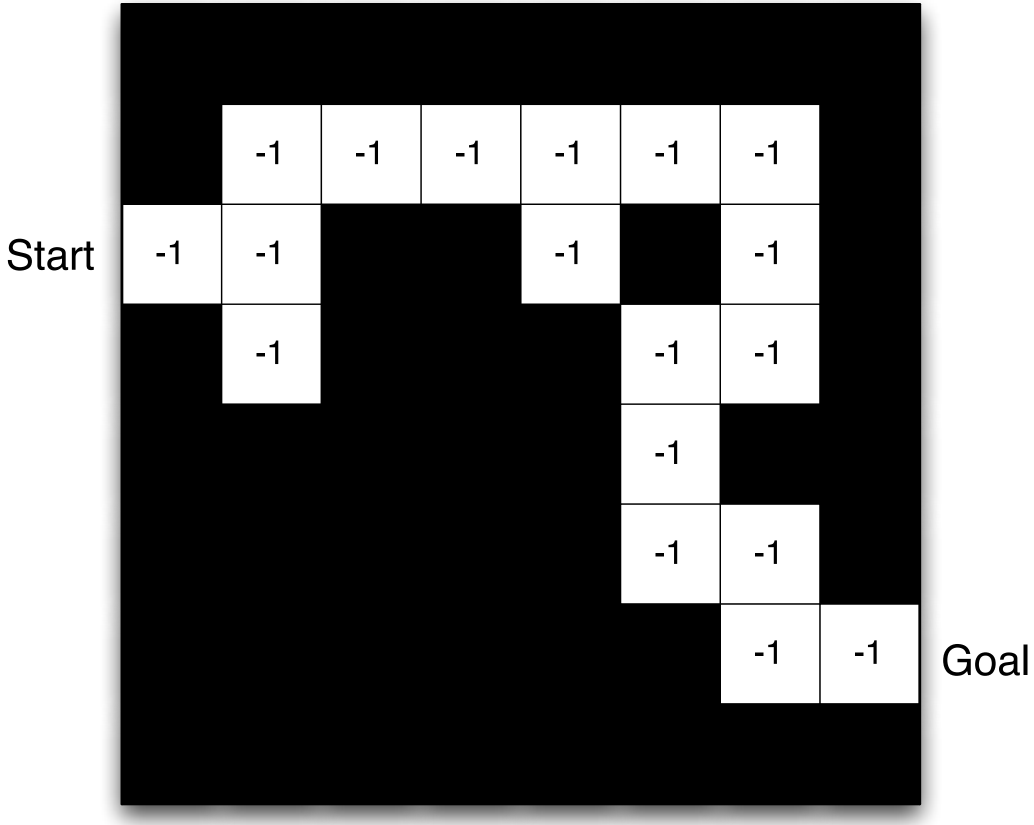
- Arrows represent policy $\pi(s)$ for each state s

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)

Learning and Planning

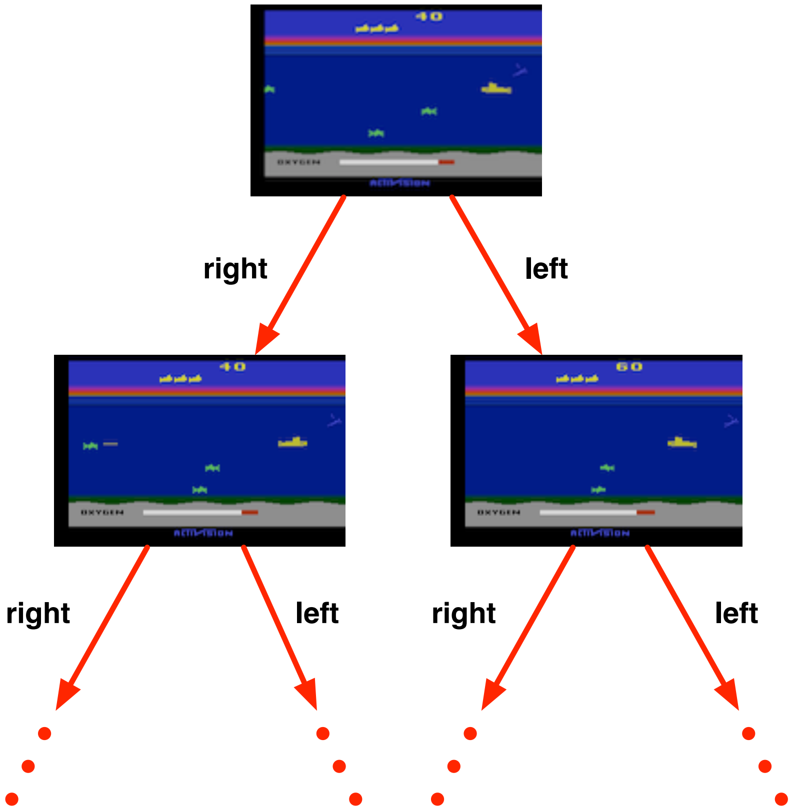
Two fundamental problems in sequential decision making

- 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 game: 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 vs 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 maximise reward
  - Exploitation Go to your favourite restaurant
  - Exploration Try a new restaurant

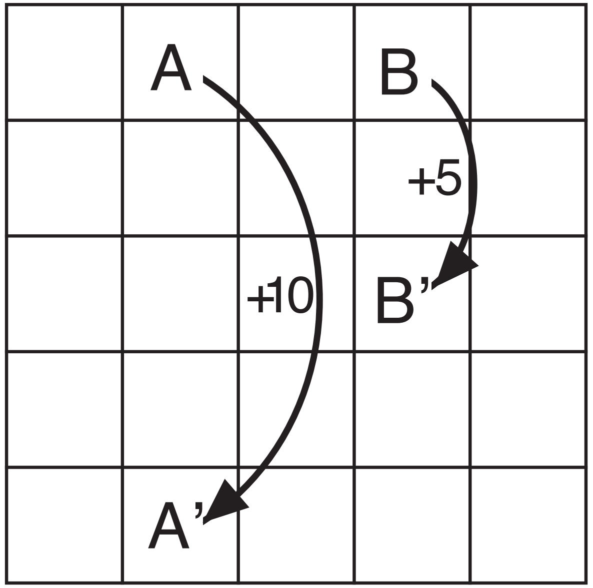
- It is usually important to explore as well as exploit

Prediction vs Control

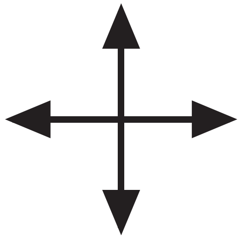
- Prediction: evaluate the future given a policy
- Control: optimise the future find the best policy

Grid world example: Prediction

With a reward value of -1 for every action, except for states A and B with immediate teleporting to A' and B', receiving +10 and +5 immediate rewards.



(a)



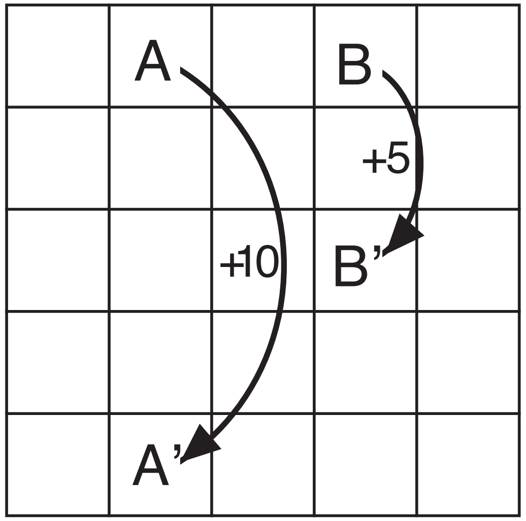
Actions

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?

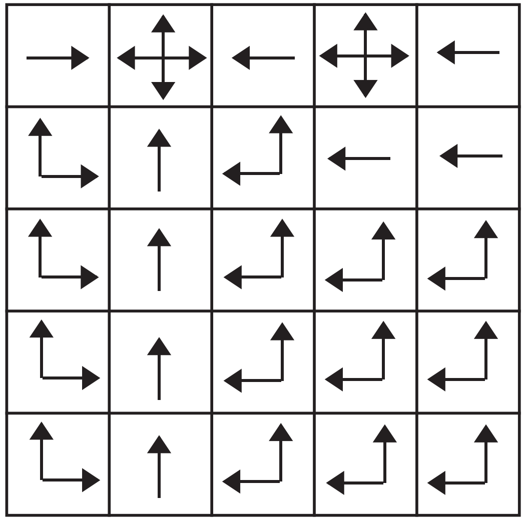
Grid world: control



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) π_*

What is the optimal value function over all possible policies?

What is the optimal policy?

Conclusion

- RL is different from both supervised and unsupervised learning
- In a Markov Decision Process form of RL the states of the agent and environment are the same
- In RL the environment is not initially known for the agent
- RL is different from planning (search algorithms)
- Agent performs both exploration and exploitation