# Introduction to Reinforcement Learning

EMAT31530/Nov 2020/Xiaoyang Wang

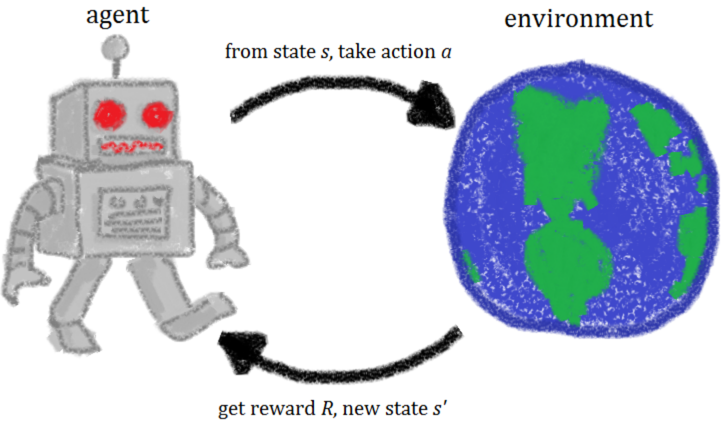
In MDP,

we know the state transition function $P(s'|a, s)$

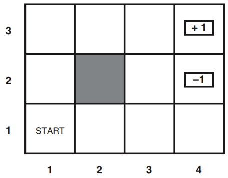
we know the reward function $R(s)$


Learning by interaction

[https://www.youtube.com/user/stanfordhelicopter]
[https://research-football.dev]
[https://www.technologyreview.com/2020/03/02/905593/ai-robot-learns-to-walk-autonomously-reinforcement-learning/]


agent
environment
from state $s$, take action $a$
get reward $R$, new state $s'$

- Actions have non-deterministic effects: initially unknown
- Rewards / punishments are infrequent and often at the end of long sequences of actions
- Agent must decide what actions to take
- World is large and complex




### Reinforcement learning

Reinforcement learning is **on-line** methodology that we use when the **model of world is unknown** and/or **rewards are delayed**.

Learning what to do

- A learning agent is able to sense the state of its environment
- Take actions to affect the state
- A goal related to the state of the environment

### RL algorithm: template

For $t = 1, 2, 3, \ldots$
 Choose an action $a_t = \pi(s_t)$ - how?
 Receive reward $r_t$, observe new state $s_{t+1}$
 Update parameters related to action selection / policy - how?

Supervised Learning? Unsupervised Learning?

# Reinforcement learning: Naive approach

## Naive Approach

1. Act randomly for a while (or systematically explore all possible actions), collect data
2. Build an MDP: Learn Transition model and Reward function
3. Use value iteration, policy iteration, ...

Problems?

# Reinforcement learning: basic approaches

## Exploration vs Exploitation

Exploit: use your learning result to maximise expected utility now, according to your learned model

Explore: choose an action that will help you improve your model

- How to explore
  - choose a random action
  - choose an action you haven't chosen yet
  - choose and action that will take you to unexplored states
- How to exploit: follow policy
- When to explore

## Model-based reinforcement learning

- Learn MDP
- Solve the MDP to determine optimal policy
- Treat the difference between expected / actual reward as an error signal

## Model-free reinforcement learning

Don't learn a model, learn the (value) function directly: Q-learning, TD learning

# Model-based vs Model-free

Model-based

- Having a model allows the agent to *plan*
- The real model of the environment is usually not available to the agent, also difficult to learn
- Model bias
- High sample efficiency

Model-free

- Low sample efficiency
- Directly working on the value function / policy
- In practice, easier to implement and tune

# Q-learning

An action-value function $Q(s, a)$, estimating $Q^*(s, a)$, says how good it is to be in a state, take an action, independent of the policy being followed.

## Algorithm

Parameters: step size $\alpha \in (0, 1]$, $\epsilon > 0$ for exploration

1. Initialise $Q(s, a)$ arbitrarily, except $Q(\text{terminal}, \cdot) = 0$
2. Choose actions using $Q$, e.g., $\epsilon -$ greedy.
3. On each time step

$$Q^{new}(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(R_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t))$$

4. Repeat step 2 and step 3

If desired, reduce the step-size parameter $\alpha$ over time

## Theorem

Q-learning control converges to the optimal action-value function.

# Q-learning

An action-value function $Q(s,a)$, estimating $Q^*(s,a)$, says how good it is to be in a state, take an action, independent of the policy being followed.

## Algorithm

Parameters: step size $\alpha \in (0,1]$, $\epsilon > 0$ for exploration

1. Initialise $Q(s,a)$ arbitrarily, except $Q(\text{terminal}, \cdot) = 0$
2. Choose actions using $Q$, e.g., $\epsilon -$ greedy.
3. On each time step

$$Q^{new}(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(R_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t))$$

4. Repeat step 2 and step 3

If desired, reduce the step-size parameter $\alpha$ over time

## Theorem

Q-learning control converges to the optimal action-value function.

# Q table

actions

| states | $a_0$ | $a_1$ | $\cdots$ |
|---|---|---|---|
| $s_0$ | $Q(s_0, a_0)$ | $Q(s_0, a_1)$ | $\cdots$ |
| $s_1$ | $Q(s_1, a_0)$ | $Q(s_1, a_1)$ | $\cdots$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |

# Q table

| | Up | Down | Left | Right |
|---|---|---|---|---|
| (1,1) | | | | |
| (1,2) | | | | |
| (1,3) | | | | |
| (2,1) | | | | |
| ... | | | | |
| (4,2) | | | | |
| (4,3) | | | | |

# Q table

| | Up | Down | Left | Right |
|---|---|---|---|---|
| (1,1) | | | | |
| (1,2) | a | b | c | d |
| (1,3) | | | | |
| (2,1) | | | | |
| ... | | | | |
| (4,2) | | | | |
| (4,3) | | | | |

$$\text{Action}[s = (1,2)] = \arg\max([a, b, c, d])$$

When state space and/or action space scale up?

# On-policy and Off-policy

**On-policy**

Evaluate and improve the same policy which is being used to generate data.

**Off-policy**

Evaluate and improve the policy which is different from the policy being used to generate data.

Q: Is Q-learning on-policy or off-policy?

$$Q^{new}(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(R_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t))$$

# Summary

- Reinforcement learning idea
- Basic approach
- Q-learning

Let's implement a Q-learning algorithm!