Basic RL

Pablo Samuel Castro

neuromatch academy

A history of RL

In reverse chronological order

neuromatch academy

2022

ChatGPT

2016-2021

AlphaGo

Stratospheric balloon control

Nuclear plasma control

Chip design

ChatGPT

2015

DQN

Stratospheric balloon control

AlphaGo

Chip design

Nuclear plasma control

ChatGPT

1998

Sutton & Barto
(The Book)

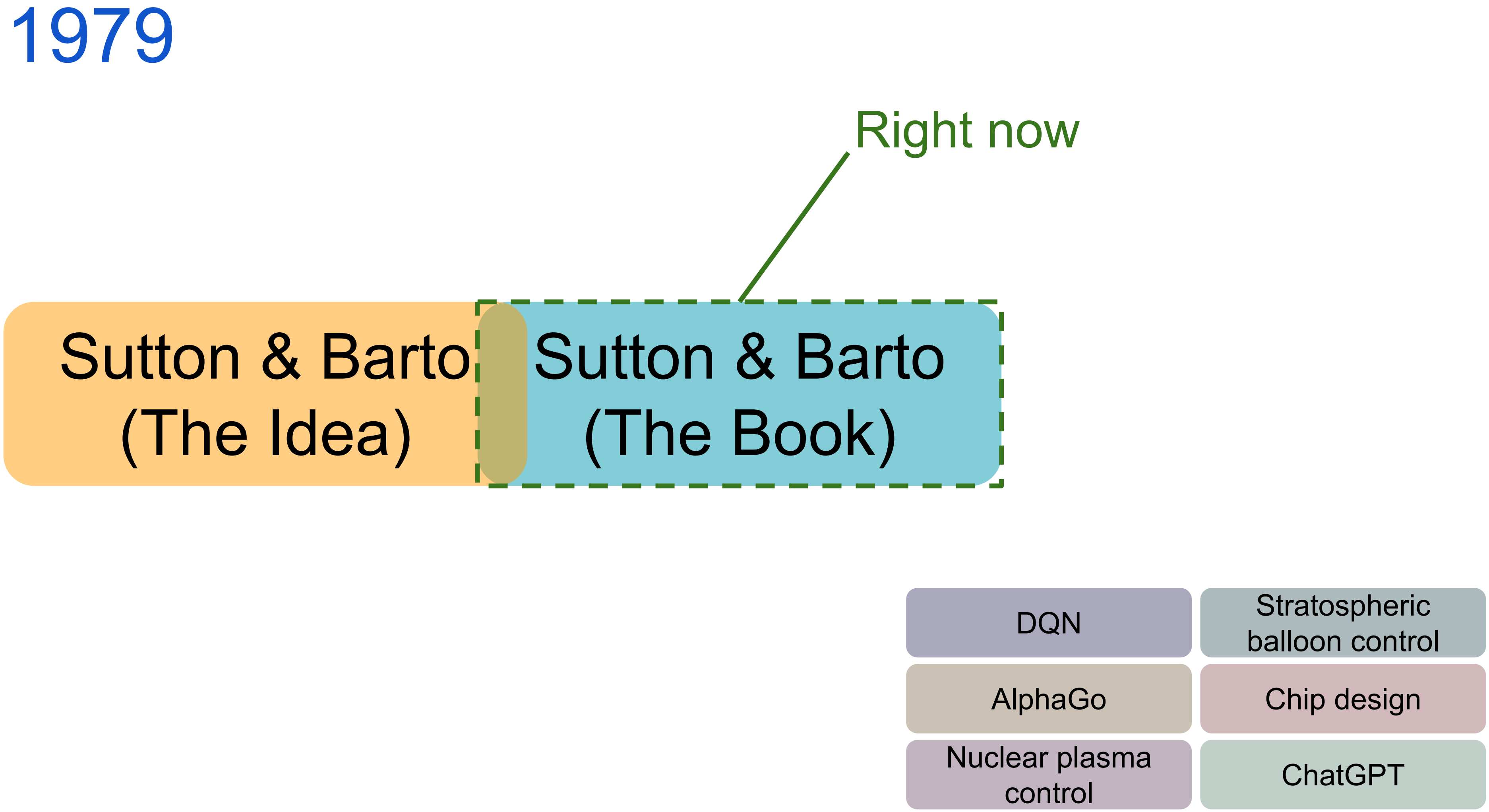
DQN	Stratospheric balloon control
AlphaGo	Chip design
Nuclear plasma control	ChatGPT

1979

Sutton & Barto
(The Idea)

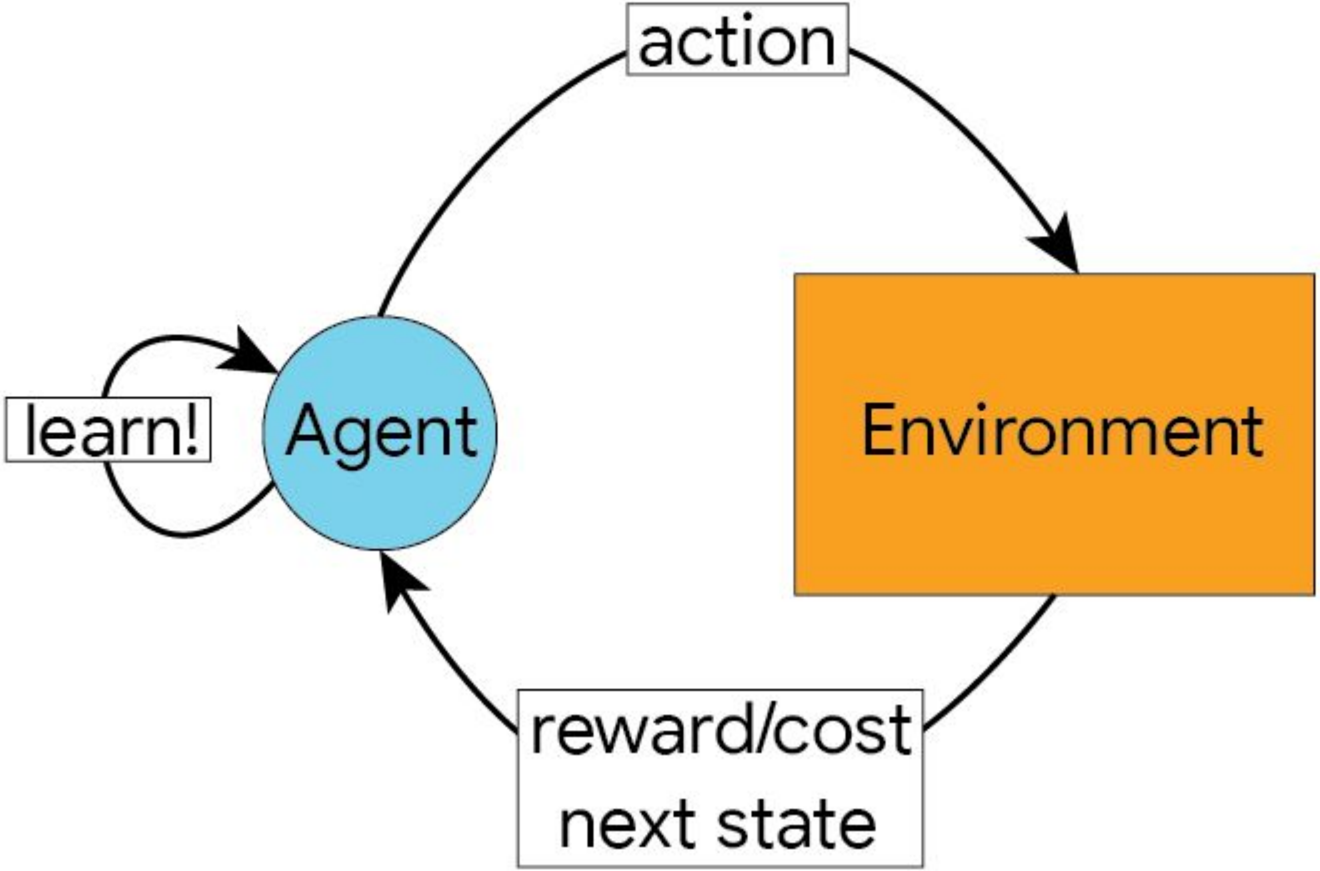
Sutton & Barto
(The Book)

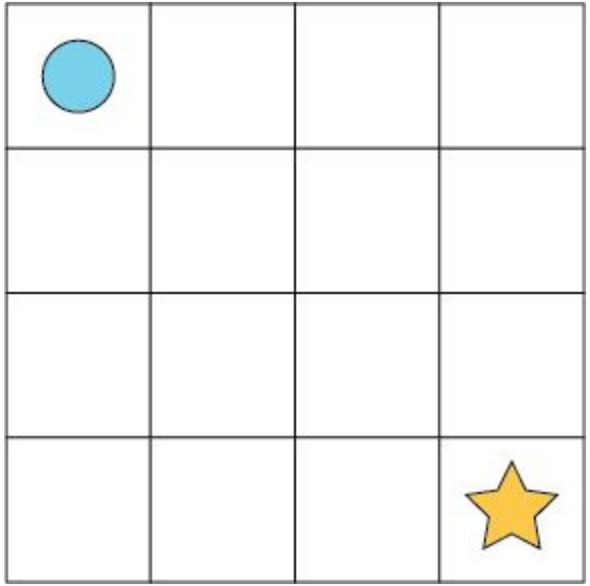
DQN	Stratospheric balloon control
AlphaGo	Chip design
Nuclear plasma control	ChatGPT


1979
Right now
Sutton & Barto
(The Idea)
Sutton & Barto
(The Book)
DQN
Stratospheric balloon control
AlphaGo
Chip design
Nuclear plasma control
ChatGPT

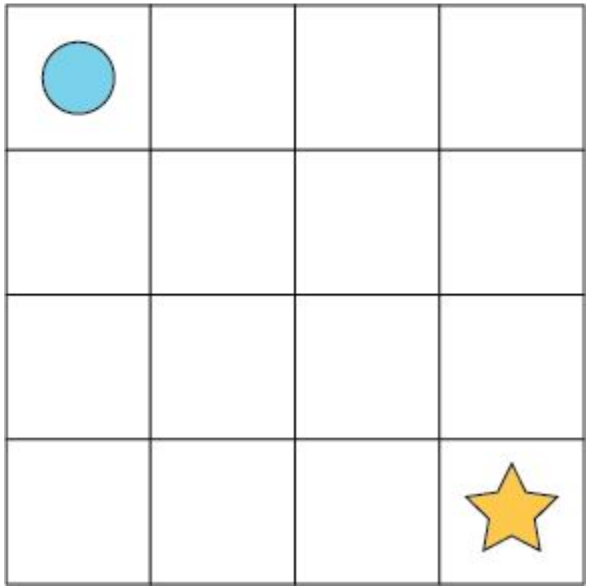
What is RL?

An illustrative toy example

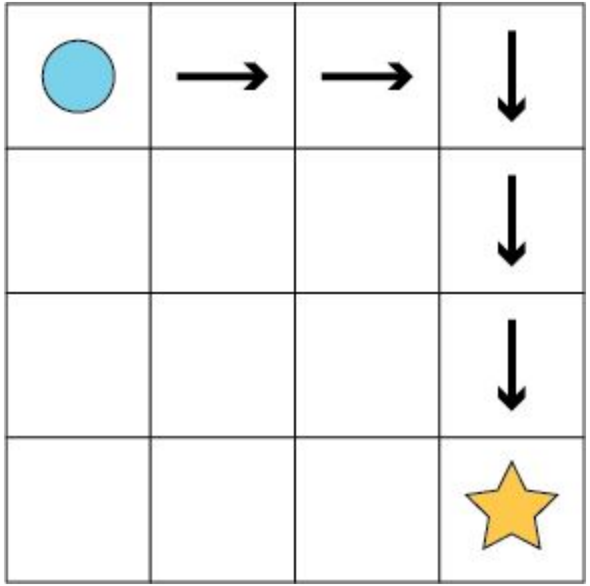

action
learn!
Agent
Environment
reward/cost
next state



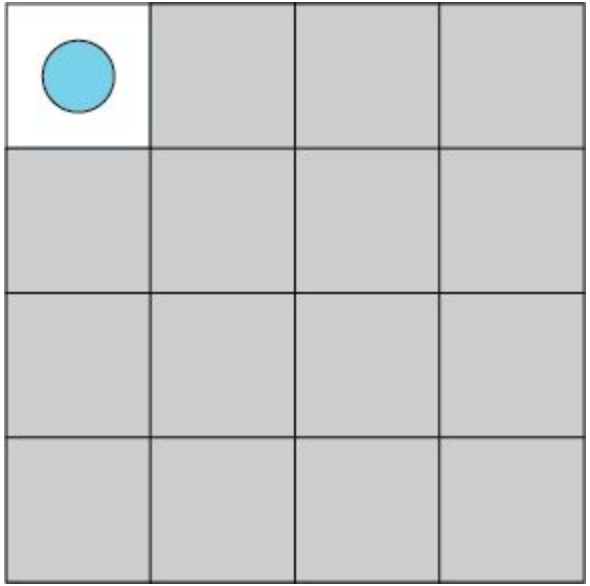
Known model: Planning

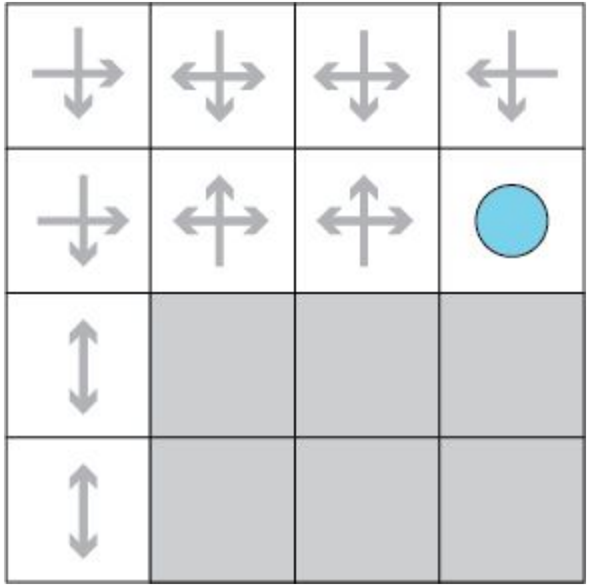


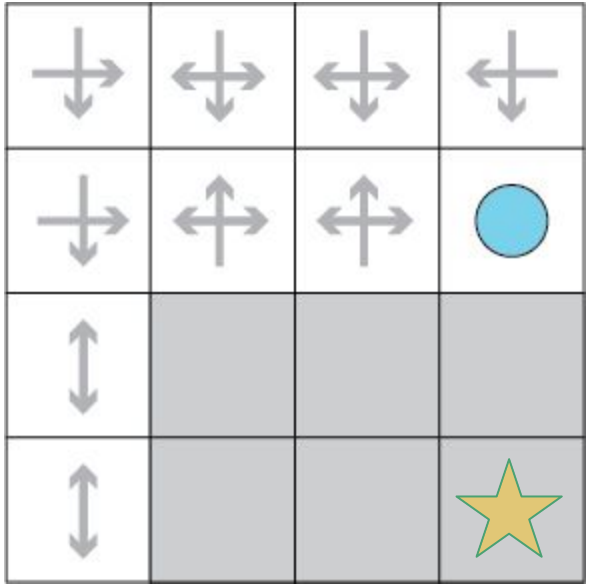
Known model: Planning



Unknown model: Reinforcement Learning!







Coding exercise 1

01 Installs and imports

02 Code a shortest-path planner for GridWorld

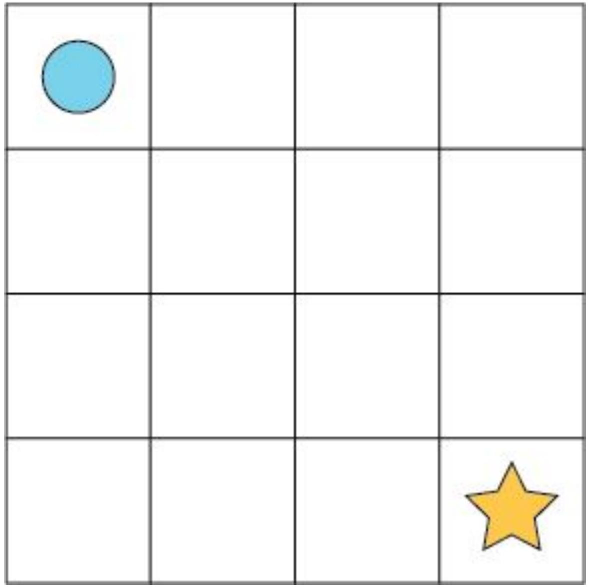
What is RL?

Formal definitions

neuromatch academy

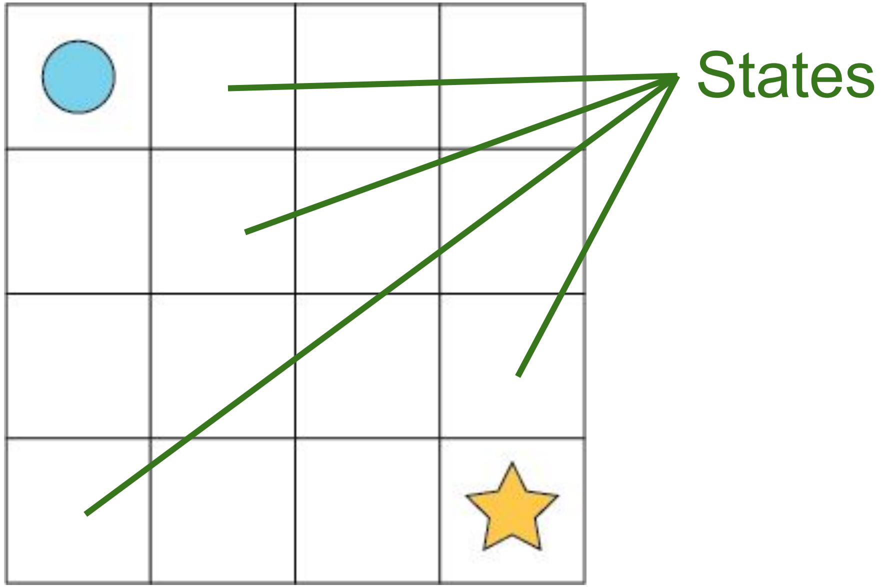
Markov decision processes

We define an MDP: $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$



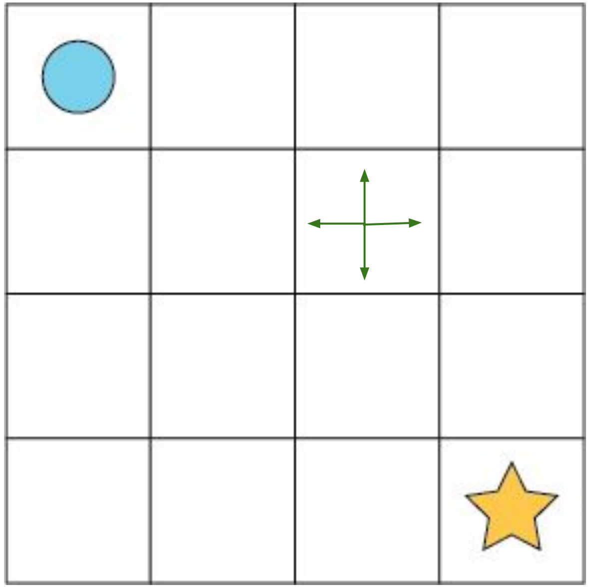
Markov decision processes

We define an MDP: $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$



Markov decision processes

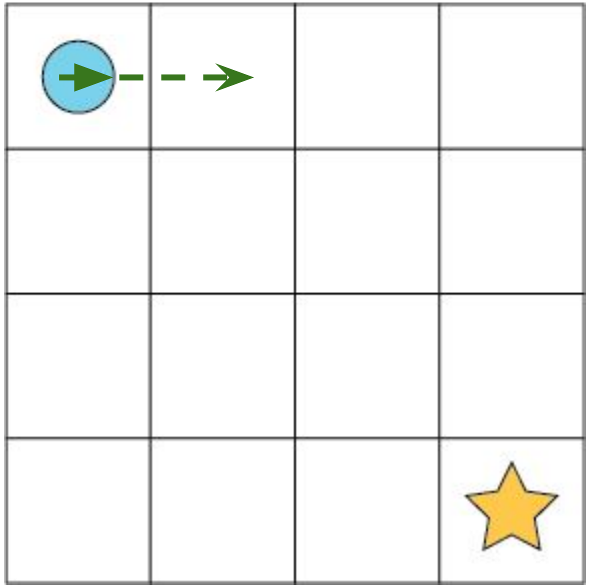
We define an MDP: $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$



Actions

Markov decision processes

We define an MDP: $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

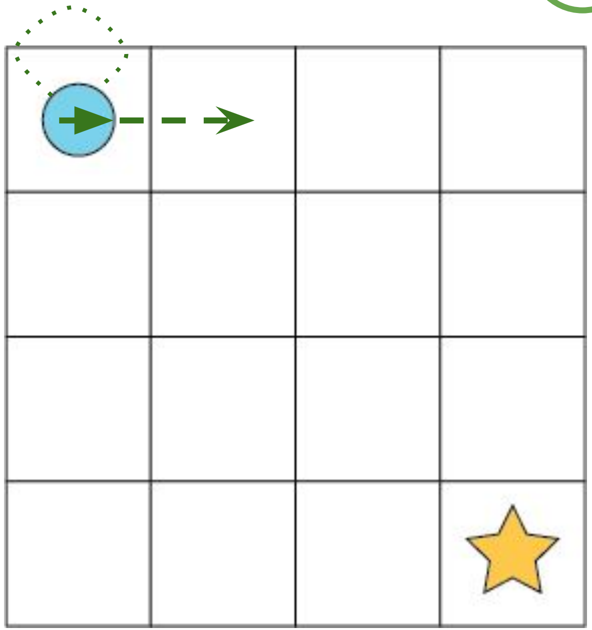


Transition dynamics

$\mathcal{P} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$

Markov decision processes

We define an MDP: $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

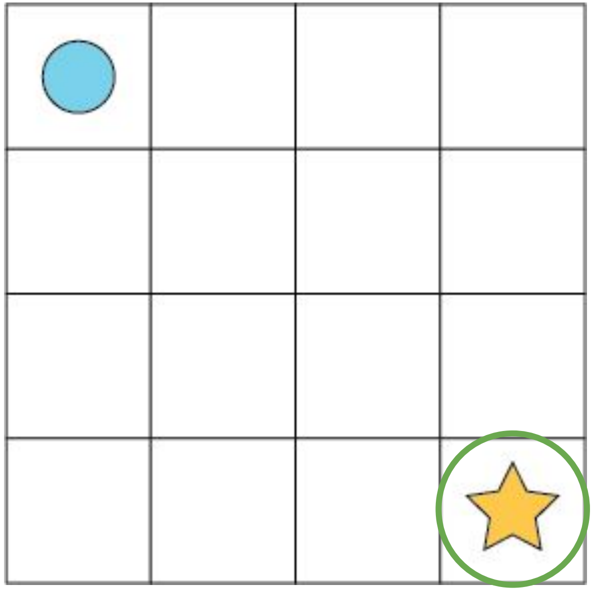


Transition dynamics

$\mathcal{P} : \mathcal{S} \times \mathcal{A} \rightarrow Dist(\mathcal{S})$

Markov decision processes

We define an MDP: $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

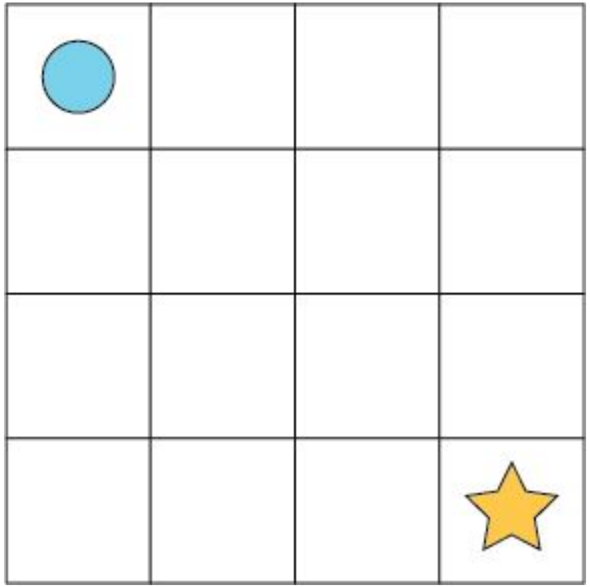


Reward function

$$\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$$

Markov decision processes

We define an MDP: $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

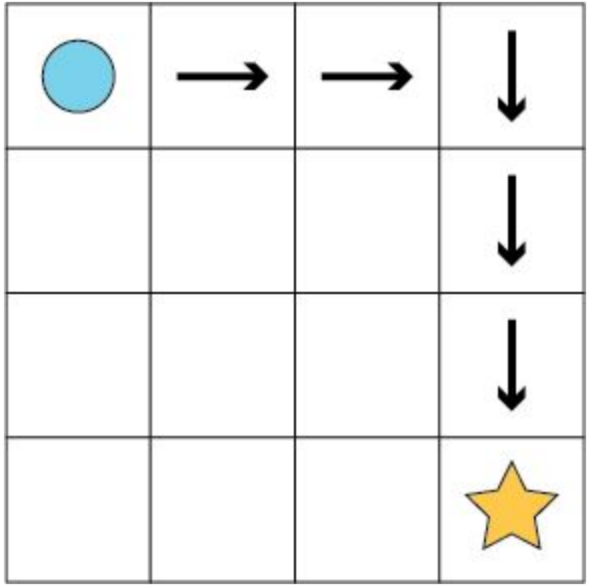


Discount factor ("don't wait too long")

Markov decision processes

We define an MDP: $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

A behaviour policy: $\pi : \mathcal{S} \rightarrow Dist(\mathcal{A})$



Coding exercise 2

01 Complete MDP class for GridWorld

02 Create “policy table” to represent π

03 Add shortest-path algorithm to use π

Markov decision processes

We define an MDP: $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

A behaviour policy: $\pi : \mathcal{S} \rightarrow Dist(\mathcal{A})$

with its respective value function:

$$V^{\pi}(s) = \mathbb{E}_{a \sim \pi(s)} \left[\mathcal{R}(s, a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}(s,a)} V^{\pi}(s') \right]$$

Markov decision processes

We define an MDP: $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

A behaviour policy: $\pi : \mathcal{S} \rightarrow Dist(\mathcal{A})$

with its respective value function:

$$V^{\pi}(s) = \mathbb{E}_{a \sim \pi(s)} \left[\mathcal{R}(s, a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}(s,a)} V^{\pi}(s') \right]$$

One-step reward

Markov decision processes

We define an MDP: $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

A behaviour policy: $\pi : \mathcal{S} \rightarrow Dist(\mathcal{A})$

with its respective value function:

$$V^{\pi}(s) = \mathbb{E}_{a \sim \pi(s)} \left[\mathcal{R}(s, a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}(s,a)} V^{\pi}(s') \right]$$

One-step reward

Discounted expected future rewards

Markov decision processes

We define an MDP: $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

A behaviour policy: $\pi : \mathcal{S} \to Dist(\mathcal{A})$

with its respective value function:

$$V^{\pi}(s) = \mathbb{E}_{a \sim \pi(s)} \left[\mathcal{R}(s, a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}(s,a)} V^{\pi}(s') \right]$$

$$V^{\pi}(s) = \sum_{t=0}^{\infty} [\gamma^t R(s_t, a_t) | s_0 = s, \pi]$$

Markov decision processes

We define an MDP: $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

A behaviour policy: $\pi : \mathcal{S} \to Dist(\mathcal{A})$

with its respective value function:

$$V^\pi(s) = \mathbb{E}_{a \sim \pi(s)} \left[\mathcal{R}(s, a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}(s,a)} V^\pi(s') \right]$$

we're typically interested in the optimal value function:

$$V^*(s) = \max_\pi \sum_{t=0}^{\infty} [\gamma^t R(s_t, a_t) | s_0 = s, \pi]$$

Markov decision processes

We define an MDP: $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

A behaviour policy: $\pi : \mathcal{S} \to Dist(\mathcal{A})$

with its respective value function:

$$V^{\pi}(s) = \mathbb{E}_{a \sim \pi(s)} \left[\mathcal{R}(s, a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}(s,a)} V^{\pi}(s') \right]$$

we're typically interested in the optimal value function:

$$V^{*}(s) = \max_{a \in \mathcal{A}} \left[\mathcal{R}(s, a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}(s,a)} V^{*}(s') \right]$$

Value functions

$$V^{\pi}(s) = \mathbb{E}_{a \sim \pi(s)} \left[\mathcal{R}(s,a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}(s,a)} V^{\pi}(s') \right]$$

$$Q^{\pi}(s,a) = R(s,a) + \gamma \mathbb{E}_{s' \sim P(s,a)} [V^{\pi}(s')]$$

$$V^{*}(s) = \max_{a \in \mathcal{A}} \left[\mathcal{R}(s,a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}(s,a)} V^{*}(s') \right]$$

$$Q^{*}(s,a) = R(s,a) + \gamma \mathbb{E}_{s' \sim P(s,a)} [V^{*}(s')]$$

Behaviour policies

$$V^*(s) = \max_{a \in \mathcal{A}} \left[\mathcal{R}(s,a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}(s,a)} V^*(s') \right]$$

$$Q^*(s,a) = R(s,a) + \gamma \mathbb{E}_{s' \sim P(s,a)} [V^*(s')]$$

$$\pi^*(s) = \arg\max_{a \in \mathcal{A}} Q^*(s,a)$$

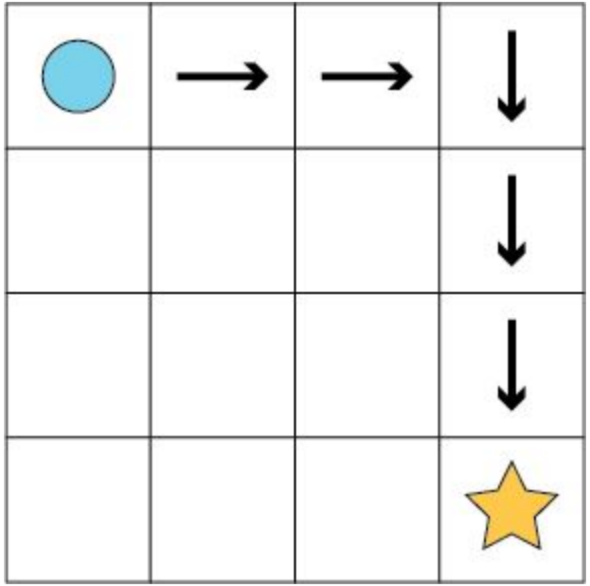
Behaviour policies

$$V^*(s) = \max_{a \in \mathcal{A}} \left[\mathcal{R}(s,a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}(s,a)} V^*(s') \right]$$

$$Q^*(s,a) = R(s,a) + \gamma \mathbb{E}_{s' \sim P(s,a)} [V^*(s')]$$

$$\pi^*(s) = \arg\max_{a \in \mathcal{A}} Q^*(s,a)$$

$$\pi^*$$



Coding exercise 3

01 Create SxA table (Q) to encode number of steps to goal (assume goal is known)

02 Extract π from Q table

03 Modify to include discount factor

How do we find π^*?

neuromatch academy

$$V^*(s) = \max_{a \in \mathcal{A}} \left[\mathcal{R}(s, a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}(s,a)} V^*(s') \right]$$

$$V^*(s) = \max_{a \in \mathcal{A}} \left[\mathcal{R}(s, a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}(s,a)} V^*(s') \right]$$

$$V^0(s) = 0$$

$$V^*(s) = \max_{a \in \mathcal{A}} \left[\mathcal{R}(s,a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}(s,a)} V^*(s') \right]$$

$$V^0(s) = 0$$

$$V^1(s) = \max_{a \in \mathcal{A}} \left[\mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s,a)(s') V^0(s') \right]$$

$$V^*(s) = \max_{a \in \mathcal{A}} \left[\mathcal{R}(s,a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}(s,a)} V^*(s') \right]$$

$$V^0(s) = 0$$

$$V^1(s) = \max_{a \in \mathcal{A}} \left[\mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s,a)(s') V^0(s') \right]$$

$$V^2(s) = \max_{a \in \mathcal{A}} \left[\mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s,a)(s') V^1(s') \right]$$

$$V^*(s) = \max_{a \in \mathcal{A}} \left[\mathcal{R}(s, a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}(s,a)} V^*(s') \right]$$

$$V^0(s) = 0$$

$$V^1(s) = \max_{a \in \mathcal{A}} \left[\mathcal{R}(s, a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s, a)(s') V^0(s') \right]$$

$$V^2(s) = \max_{a \in \mathcal{A}} \left[\mathcal{R}(s, a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s, a)(s') V^1(s') \right]$$

$$\vdots$$

$$V^*(s) = \max_{a \in \mathcal{A}} \left[\mathcal{R}(s,a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}(s,a)} V^*(s') \right]$$

$$V^0(s) = 0$$

$$V^1(s) = \max_{a \in \mathcal{A}} \left[\mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s,a)(s') V^0(s') \right]$$

$$V^2(s) = \max_{a \in \mathcal{A}} \left[\mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s,a)(s') V^1(s') \right]$$

$$\vdots$$

$$V^*(s) = \max_{a \in \mathcal{A}} \left[\mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s,a)(s') V^*(s') \right]$$

$$V^*(s) = \max_{a \in \mathcal{A}} \left[\mathcal{R}(s,a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}(s,a)} V^*(s') \right]$$

$$V^0(s) = 0$$

$$V^1(s) = \max_{a \in \mathcal{A}} \left[\mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s,a)(s') V^0(s') \right]$$

$$V^2(s) = \max_{a \in \mathcal{A}} \left[\mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s,a)(s') V^1(s') \right]$$

$$\vdots$$

$$V^*(s) = \max_{a \in \mathcal{A}} \left[\mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s,a)(s') V^*(s') \right]$$

Value Iteration

$$V^0(s) = 0$$

$$V^1(s) = \max_{a \in \mathcal{A}} \left[\mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s,a)(s') V^0(s') \right]$$

$$V^2(s) = \max_{a \in \mathcal{A}} \left[\mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s,a)(s') V^1(s') \right]$$

$$\vdots$$

$$V^*(s) = \max_{a \in \mathcal{A}} \left[\mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s,a)(s') V^*(s') \right]$$

Value Iteration

Bellman backup

$$V^0(s) = 0$$

$$V^1(s) = \max_{a \in \mathcal{A}} \left[\boxed{\mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s,a)(s')V^0(s')} \right]$$

$$V^2(s) = \max_{a \in \mathcal{A}} \left[\mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s,a)(s')V^1(s') \right]$$

$$\vdots$$

$$V^*(s) = \max_{a \in \mathcal{A}} \left[\mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s,a)(s')V^*(s') \right]$$

Theorem: Value iteration converges to a fixed point

Theorem: Value iteration converges to a fixed point

$$T(V)(s) = \max_{a \in \mathcal{A}} \left[R(s,a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}(s,a)} V(s') \right]$$

Theorem: Value iteration converges to a fixed point

$$T(V)(s) = \max_{a \in \mathcal{A}} \left[R(s,a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}(s,a)} V(s') \right]$$

$$\|V - V'\|_\infty = \max_{s \in \mathcal{S}} |V(s) - V'(s)|$$

Theorem: Value iteration converges to a fixed point

$$T(V)(s) = \max_{a \in \mathcal{A}} \left[R(s,a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}(s,a)} V(s') \right]$$

$$\|V - V'\|_\infty = \max_{s \in \mathcal{S}} |V(s) - V'(s)|$$

$$\|T(V) - T(V')\|_\infty \leq \gamma \|V - V'\|_\infty$$

Theorem: Value iteration converges to a fixed point

$$T(V)(s) = \max_{a \in \mathcal{A}} \left[R(s,a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}(s,a)} V(s') \right]$$

$$\|V - V'\|_\infty = \max_{s \in \mathcal{S}} |V(s) - V'(s)|$$

$$\|T(V) - T(V')\|_\infty \leq \gamma \|V - V'\|_\infty$$

Since $\|\cdot\|_\infty$ is a complete metric space, by Banach's fixed point theorem, **T** converges to a fixed point:

$$T(V^*) = V^*$$

Value Iteration

$$V^0 \to V^*$$

Value Iteration

$$V^0 \rightarrow V^*$$

$$Q^*(s,a) = \mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s,a)(s')V^*(s')$$

Value Iteration

$$V^0 \to V^*$$

$$Q^*(s,a) = \mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s,a)(s') V^*(s')$$

$$\pi^*(s) = \arg\max_{a \in \mathcal{A}} Q^*(s,a)$$

Value Iteration

1. Initialize **Q** arbitrarily (e.g. set to 0 for each state **s** and action **a**)

2. While **Q** is changing:

    - $$Q(s,a) = \mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s,a)(s') \max_{a' \in \mathcal{A}} Q(s',a')$$

3. For every state **s:**

    - $$\pi(s) = \arg\max_{a \in \mathcal{A}} Q^*(s,a)$$

4. Return **π**

Coding exercise 4

01 Code up value iteration to compute Q*

02 Extract V* from Q*

03 Extract π* from Q*

04 Visualize V*

Value Iteration

$$V^0 \rightarrow V^*$$

$$Q^*(s,a) = \mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s,a)(s')V^*(s')$$

$$\boxed{\pi^*(s)} = \arg\max_{a \in \mathcal{A}} Q^*(s,a)$$

If this is what we're after…
Isn't this kind of indirect?

Policy Iteration

1. Initialize **π** arbitrarily (e.g. for each state **s**, pick a random action **a**)

2. While **π** is changing:

   - $$Q(s,a) = \mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s,a)(s') Q(s', \pi(s'))$$
   - $$\pi(s) = \arg\max_{a \in \mathcal{A}} Q(s,a)$$

3. Return **π**

Coding exercise 5

01 Code up policy iteration

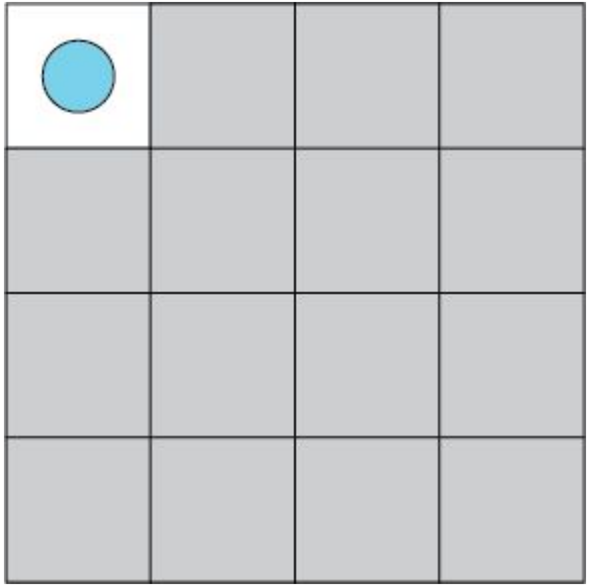
02 Compare performance with value iteration

But there's a problem

We're assuming we know

- the full state space $\mathcal{S}$
- the reward function $\mathcal{R}$
- the transition dynamics $\mathcal{P}$

Unknown model: Reinforcement Learning!



Temporal differences

- Let's say we have some estimate of Q-values
- And now let's say we observe $s, a \to s', r$
- The **temporal difference** is:

$$r + \gamma \max_{a' \in \mathcal{A}} Q(s', a') - Q(s, a)$$

Temporal differences

- Let's say we have some estimate of Q-values
- And now let's say we observe $s, a \rightarrow s', r$
- The **temporal difference** is:

$$\underbrace{\boxed{r + \gamma \max_{a' \in \mathcal{A}} Q(s', a')}}_{\text{Bellman backup}} - Q(s, a)$$

Bellman backup

Temporal differences

- Let's say we have some estimate of Q-values
- And now let's say we observe $s, a \rightarrow s', r$
- The **temporal difference** is:

Current estimate

$$\underbrace{\boxed{r + \gamma \max_{a' \in \mathcal{A}} Q(s', a')}}_{\text{Bellman backup}} - \boxed{Q(s, a)}$$

Bellman backup

Q-learning

1. Initialize **Q** and **π**, pick a start state **s**
2. While learning
    a. Pick **a** according to **π**
    b. Send **a** to the environment and receive **s'** and **r**
    c. Compute TD-error:

    $$\delta = r + \gamma \max_{a' \in \mathcal{A}} Q(s', a') - Q(s, a)$$

    d. Update the estimates for Q:

    $$Q(s, a) = Q(s, a) + \alpha\delta$$

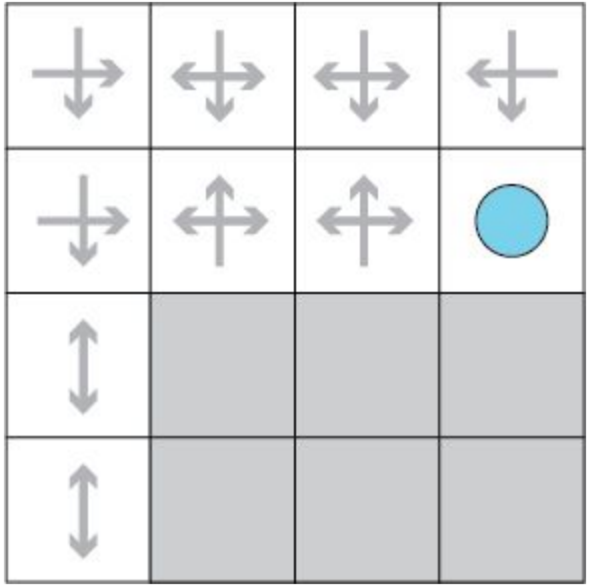
    e. $\pi(s) = \arg\max_{a \in \mathcal{A}} Q(s, a)$

    f. Update **s** = **s'**

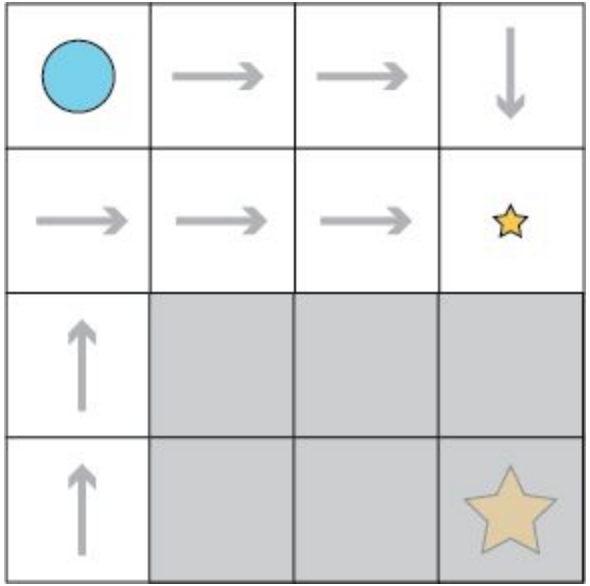
Coding exercise 6

01 Code up Q-learning

02 Test it! Did it work? If not, why not?



Exploration



Exploration: ε-greedy

- With probability $1 - \varepsilon$:
  - Select the action according to $\boldsymbol{\pi}$
- With probability ε:
  - Select a random action

Q-learning

1. Initialize **Q** and **π**, pick a start state **s**
2. While learning
   a. Pick **a** according to **π**
   b. Send **a** to the environment and receive **s'** and **r**
   c. Compute TD-error:

$$\delta = r + \gamma \max_{a' \in \mathcal{A}} Q(s', a') - Q(s, a)$$

   d. Update the estimates for Q:

$$Q(s, a) = Q(s, a) + \alpha\delta$$

   e. $\pi(s) = \arg\max_{a \in \mathcal{A}} Q(s, a)$
   f. Update **s** = **s'**

Watkins, 1989
Watkins & Dayan, 1992

Q-learning

1. Initialize **Q** and **π**, pick a start state **s**
2. While learning
   a. Pick **a** according to **π** **(plus any exploration strategy)**
   b. Send **a** to the environment and receive **s'** and **r**
   c. Compute TD-error:

   $$\delta = r + \gamma \max_{a' \in \mathcal{A}} Q(s', a') - Q(s, a)$$

   d. Update the estimates for Q:

   $$Q(s, a) = Q(s, a) + \alpha\delta$$

   e. $\pi(s) = \arg\max_{a \in \mathcal{A}} Q(s, a)$

   f. Update **s** = **s'**

Google

Coding exercise 7

01 Modify Q-learning to include ε-greedy exploration

02 Try different values of ε, what happens?