Reinforcement Learning

Misaj Sharafudeen

Classes of Learning Problems

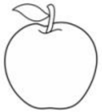
Supervised Learning

Data: (x, y)
x is data, y is label

Goal: Learn function to map

$$x \rightarrow y$$

Apple example:



This thing is an apple.

Unsupervised Learning

Data: x
x is data, no labels!

Goal: Learn underlying structure

Apple example:

This thing is like the other thing.

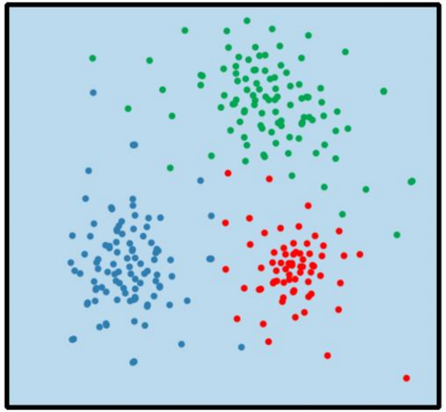
Reinforcement Learning

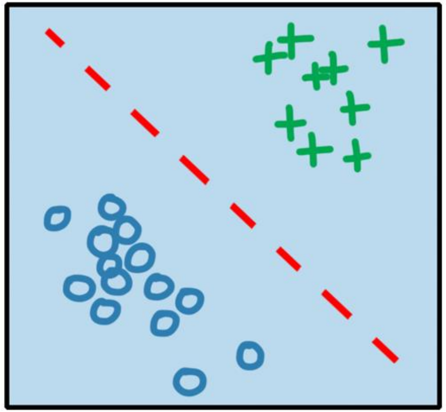
Data: state-action pairs

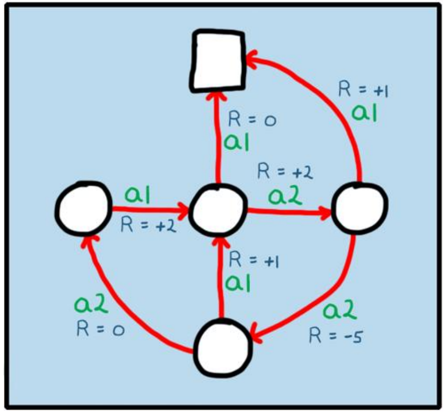
Goal: Maximize future rewards over many time steps

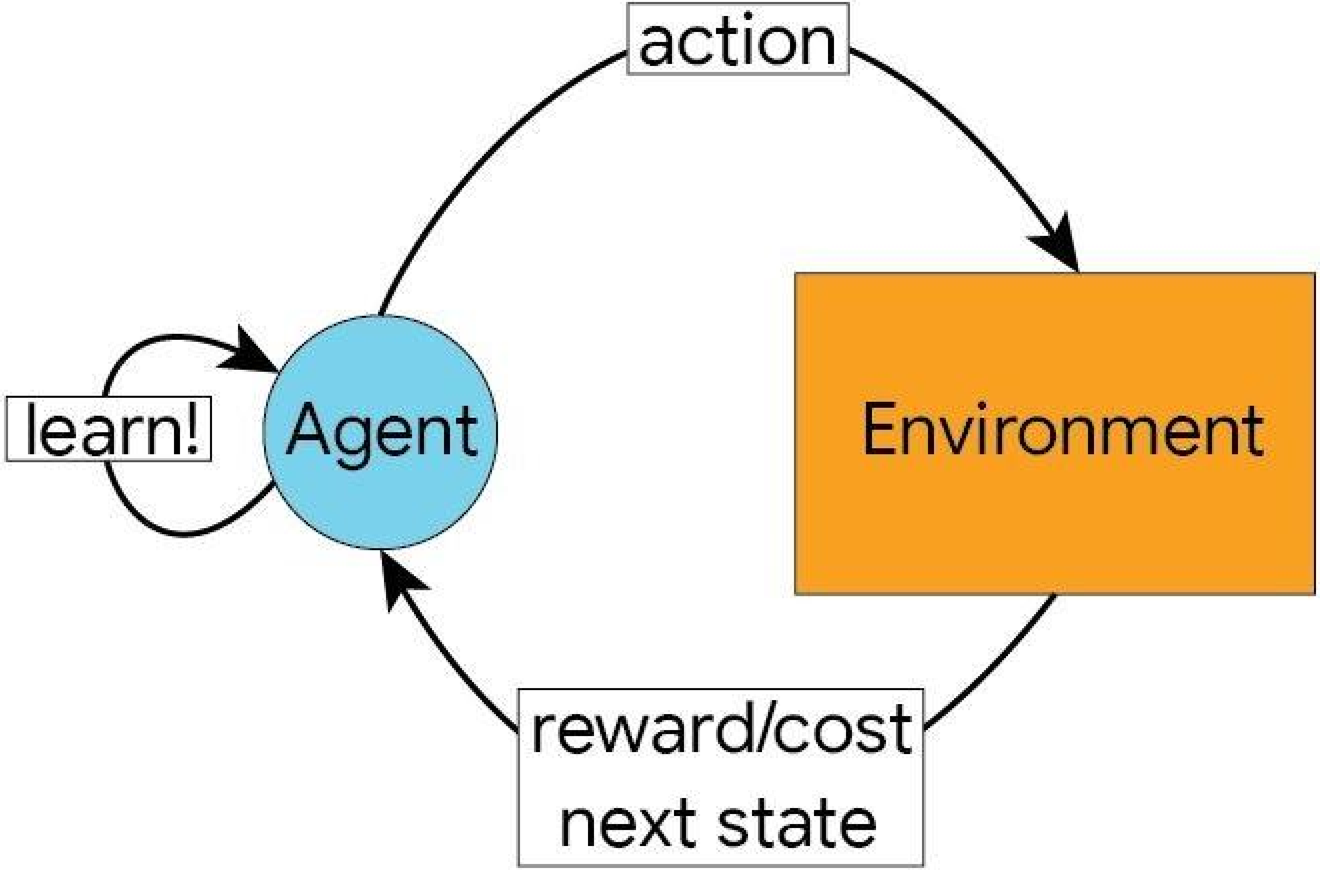
Apple example:

Eat this thing because it will keep you alive.






R = +1
a1
R = 0
a1
R = +2
a1
a2
R = +2
R = +1
a1
a2
R = 0
a2
R = -5


action
learn!
Agent
Environment
reward/cost
next state

Key Concepts

Agent: takes actions.

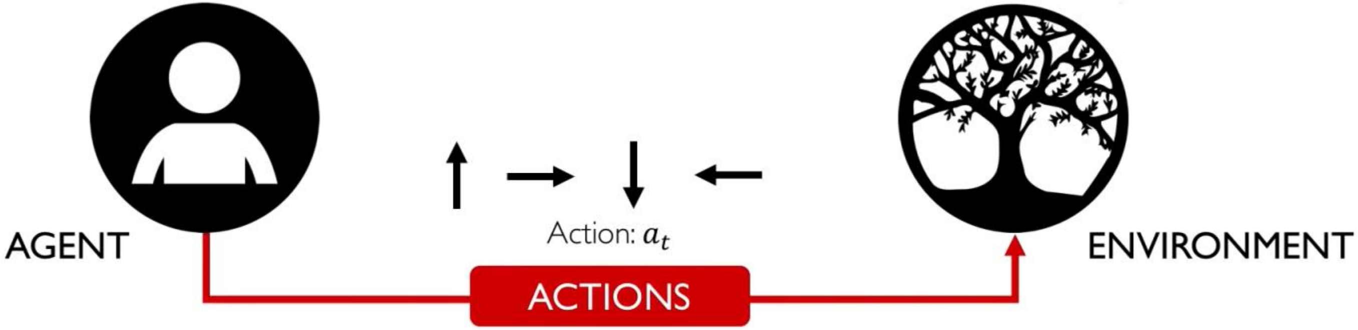
Key Concepts

Environment: the world in which the agent exists and operates.

Key Concepts

Action: a move the agent can make in the environment.

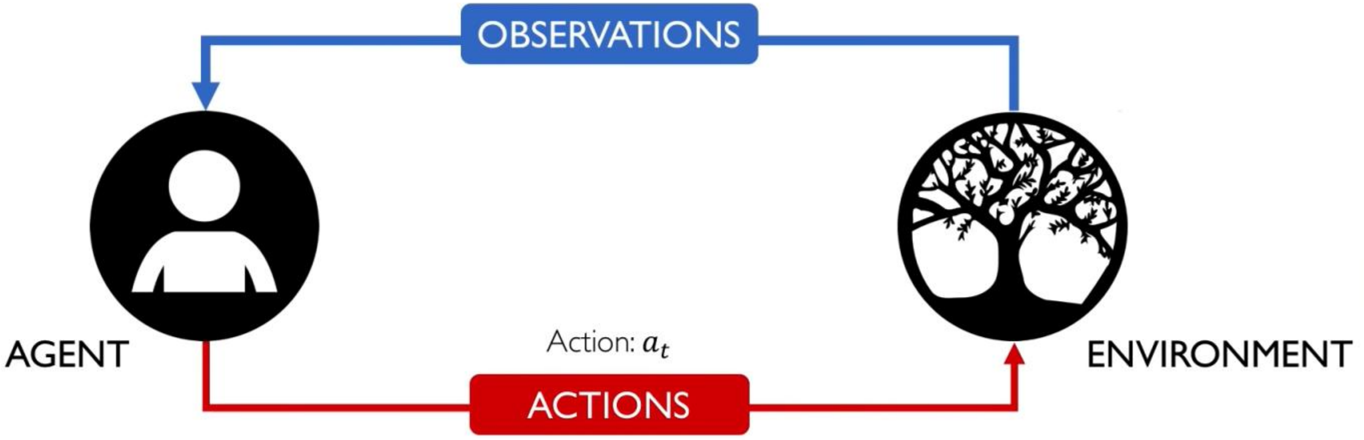
Key Concepts



Action: a move the agent can make in the environment.

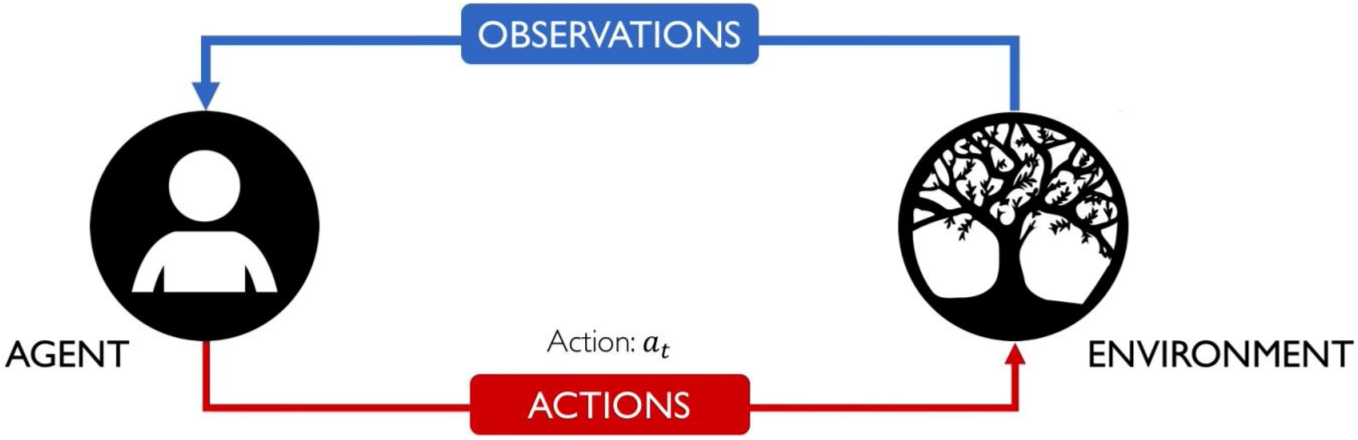
Action space A: the set of possible actions an agent can make in the environment

Key Concepts



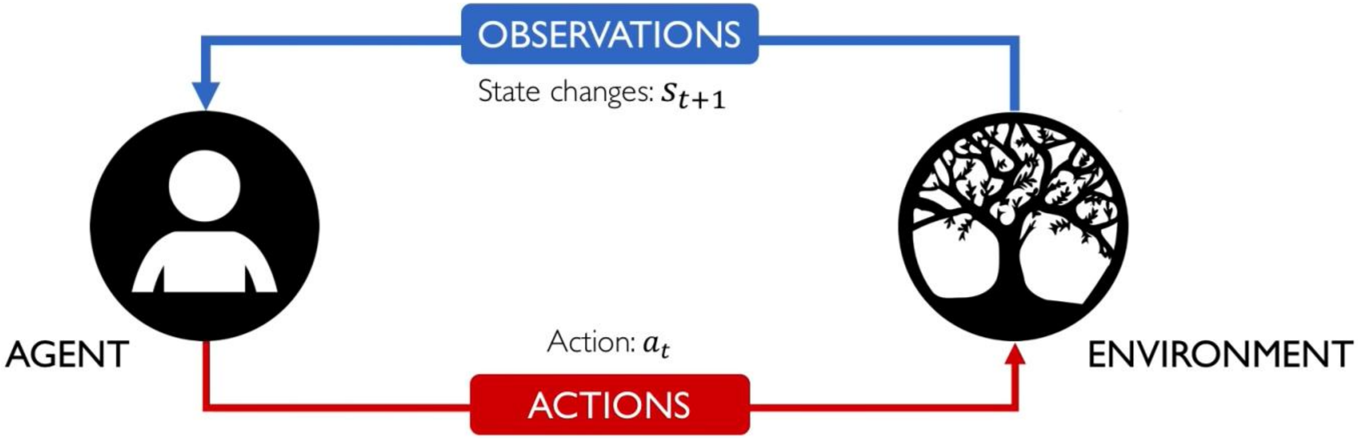
Observations: of the environment after taking actions.

Key Concepts



Observations: of the environment after taking actions.

Key Concepts

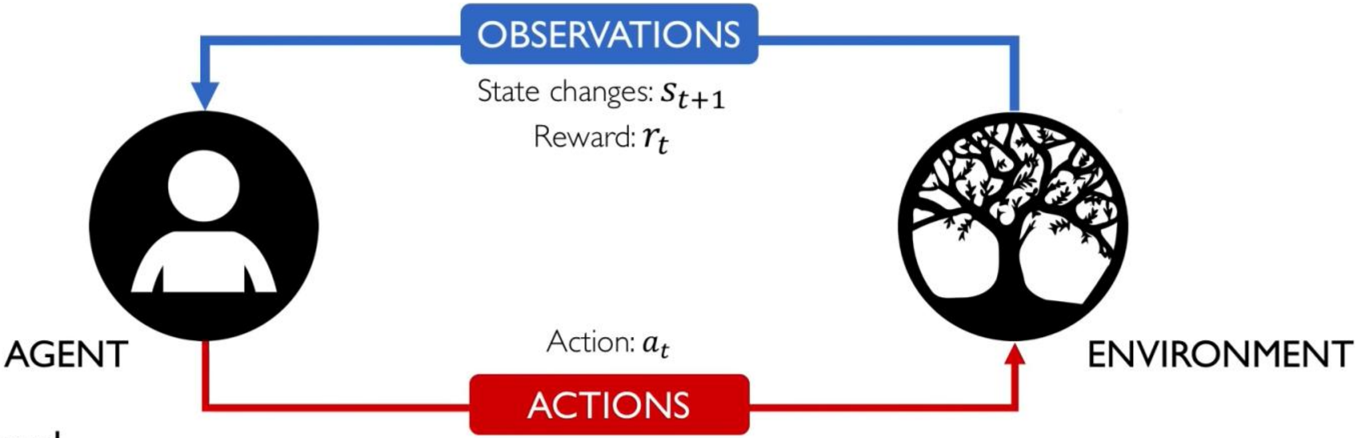


State: a situation which the agent perceives.

Key Concepts

Reward: feedback that measures the success or failure of the agent's action.

Key Concepts



Total Reward
(Return)

$$R_t = \sum_{i=t}^{\infty} r_i = r_t + r_{t+1} \dots + r_{t+n} + \cdots$$

Key Concepts

Discounted Total Reward **(Return)**

$$R_t = \sum_{i=t}^{\infty} \gamma^i r_i = \gamma^t r_t + \gamma^{t+1} r_{t+1} \dots + \gamma^{t+n} r_{t+n} + \cdots$$

γ: discount factor; $0 < \gamma < 1$

Maximize Reward

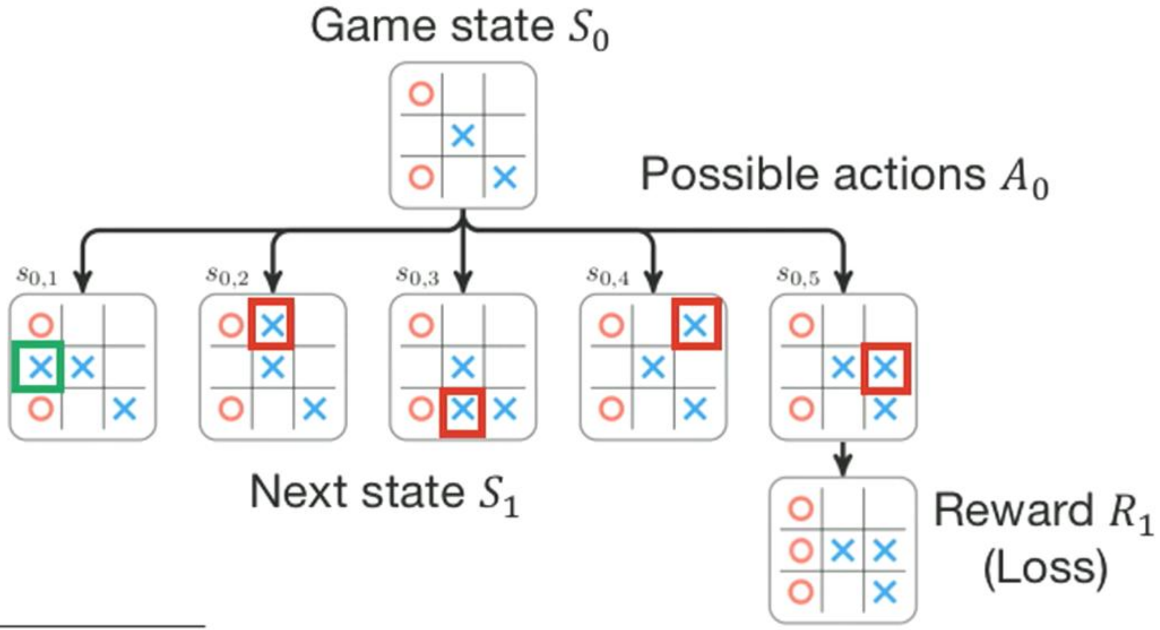


Figure by Tim Wheeler, tim.hibal.org

What is Reinforcement Learning?

- Learning from interaction
- Goal-oriented learning
- Learning about, from, and while interacting with an external environment
- Learning what to do—how to map situations to actions—so as to maximize a numerical reward signal

Key Features of RL

- Learner is not told which actions to take
- Trial-and-Error search
- Possibility of delayed reward (sacrifice short-term gains for greater long-term gains)
- The need to ***explore*** and ***exploit***
- Considers the whole problem of a goal-directed agent interacting with an uncertain environment

Defining the Q-function

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots$$

Total reward, R_t, is the discounted sum of all rewards obtained from time t

Quality Function- How useful a given action is in gaining some future reward

Defining the Q-function

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots$$

Total reward, R_t, is the discounted sum of all rewards obtained from time t

$$Q(s_t, a_t) = \mathbb{E}[R_t | s_t, a_t]$$

The Q-function captures the **expected total future reward** an agent in state, s, can receive by executing a certain action, a

How to take actions given a Q-function?

$$Q(s_t, a_t) = \mathbb{E}[R_t | s_t, a_t]$$

(state, action)

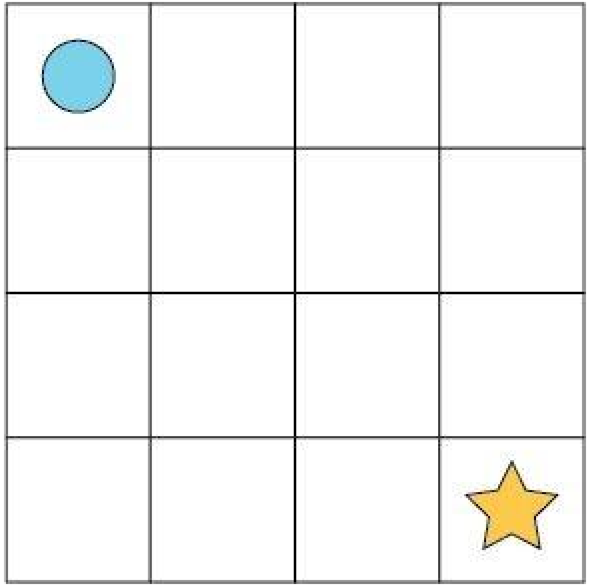
Ultimately, the agent needs a **policy $\boldsymbol{\pi(s)}$,** to infer the **best action to take** at its state, s

Strategy: the policy should choose an action that maximizes future reward

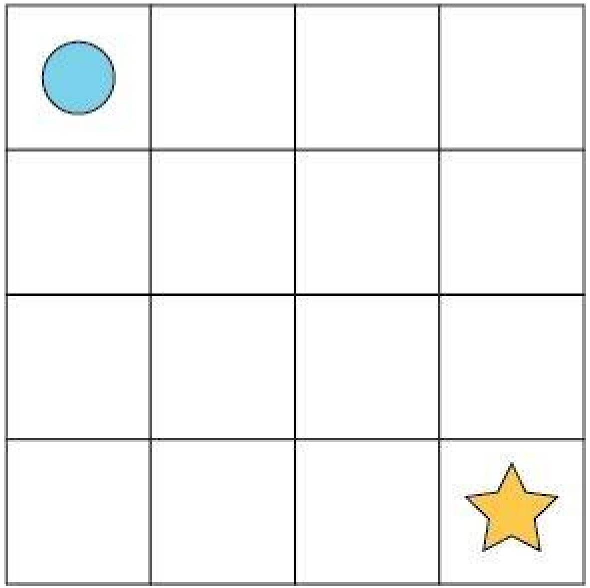
$$\pi^*(s) = \underset{a}{\operatorname{argmax}}\, Q(s, a)$$

What is RL?

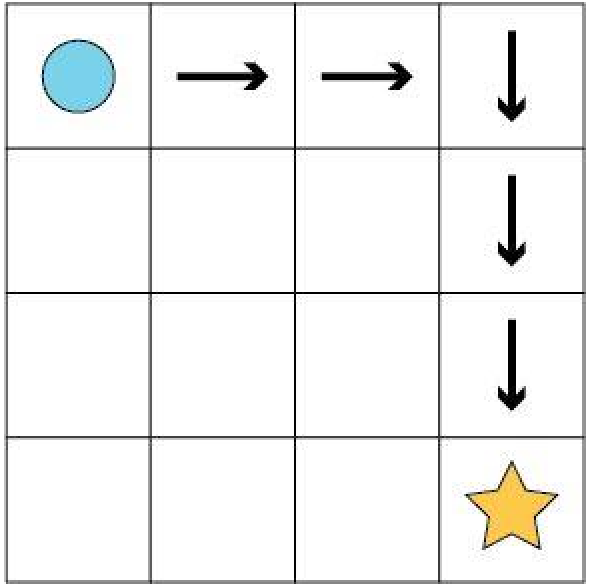
An illustrative toy example



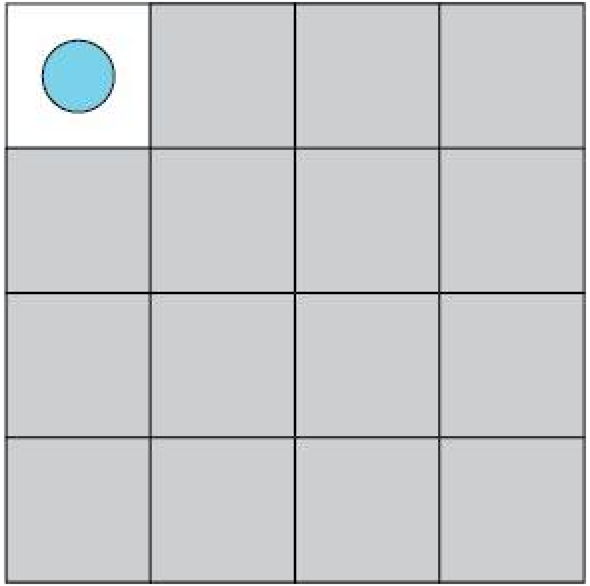
Known model: Planning

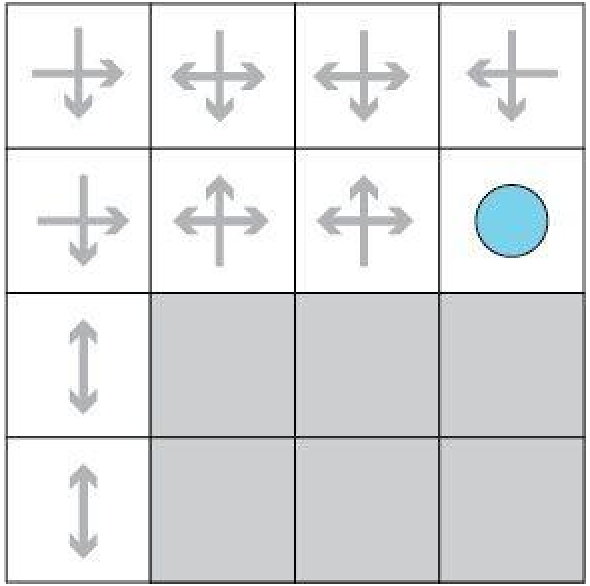


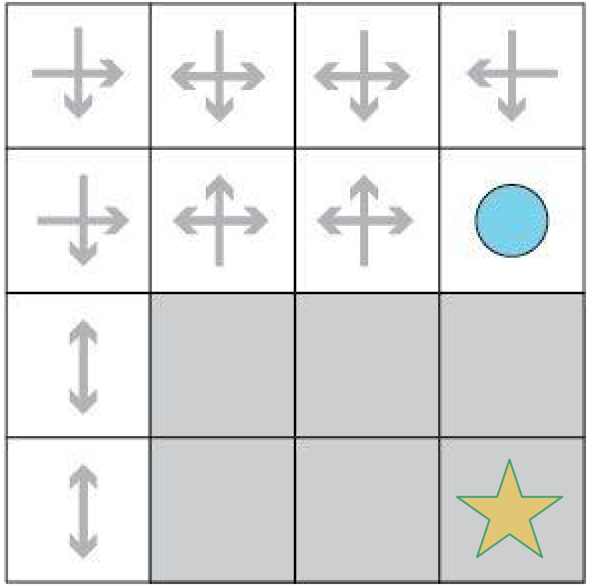
Known model: Planning



Unknown model: Reinforcement Learning!





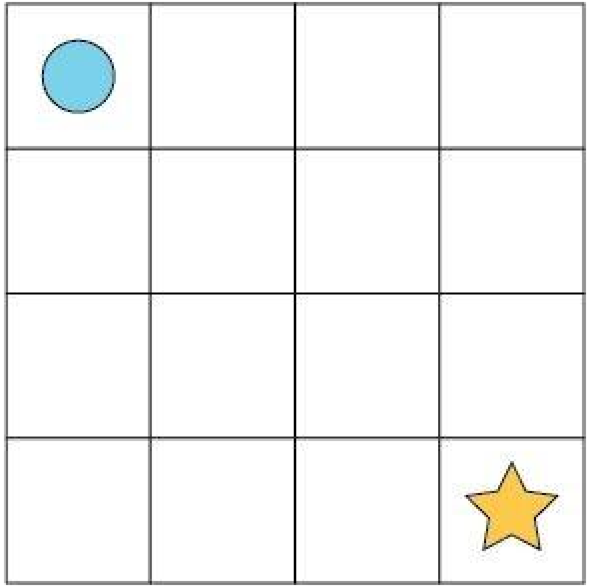


What is RL?

Formal Definitions

Markov decision processes

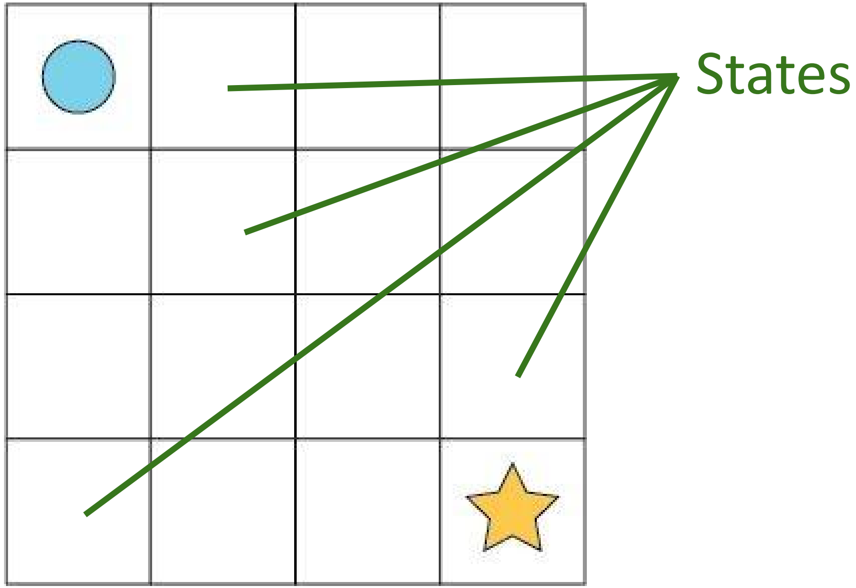
We define an MDP: $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$



Markov decision processes

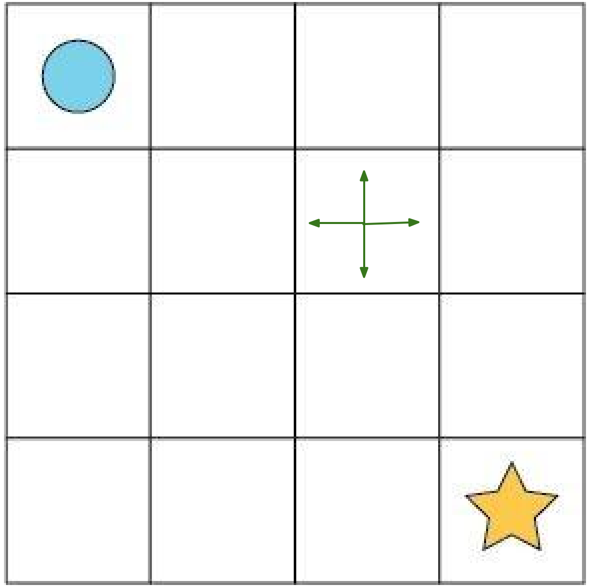
We define an MDP:

$$\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$$



Markov decision processes

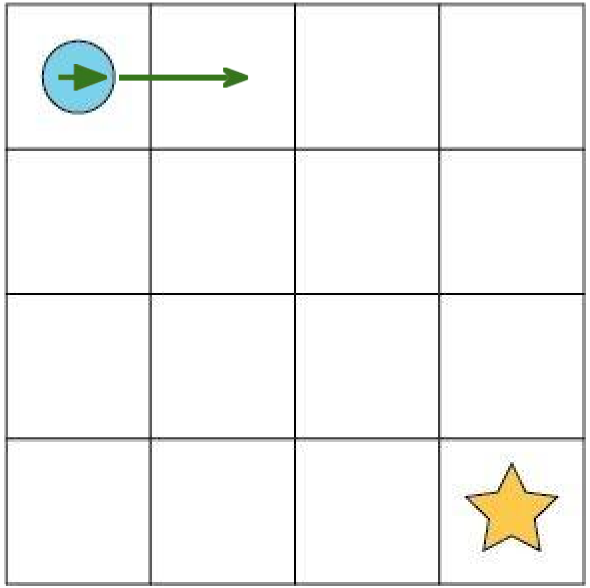
We define an MDP: $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$



Actions

Markov decision processes

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

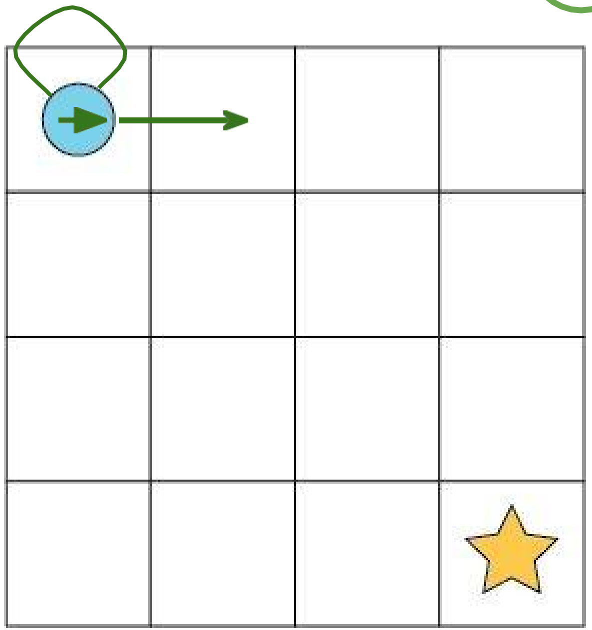


Transition dynamics

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

Markov decision processes

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

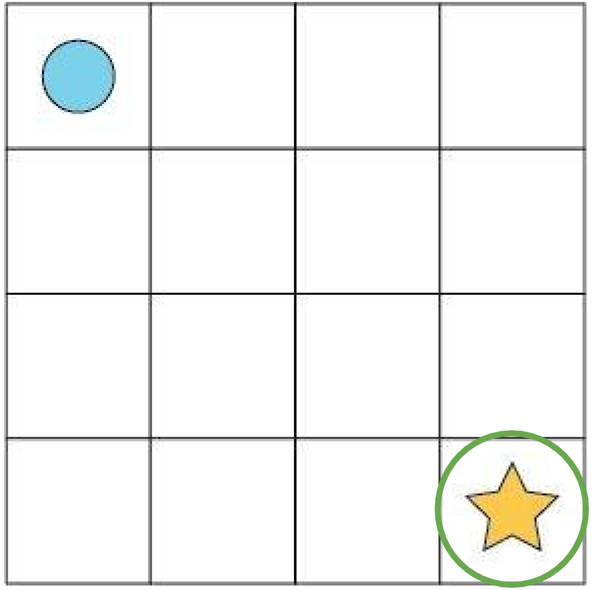


Transition dynamics

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

Markov decision processes

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

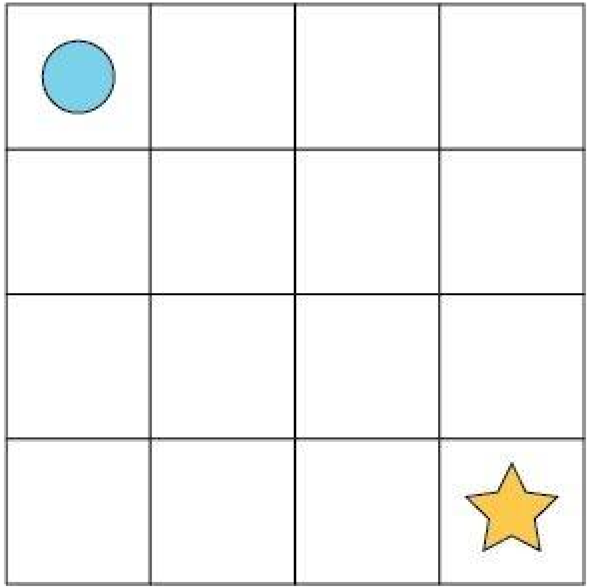


Reward function

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

Markov decision processes

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

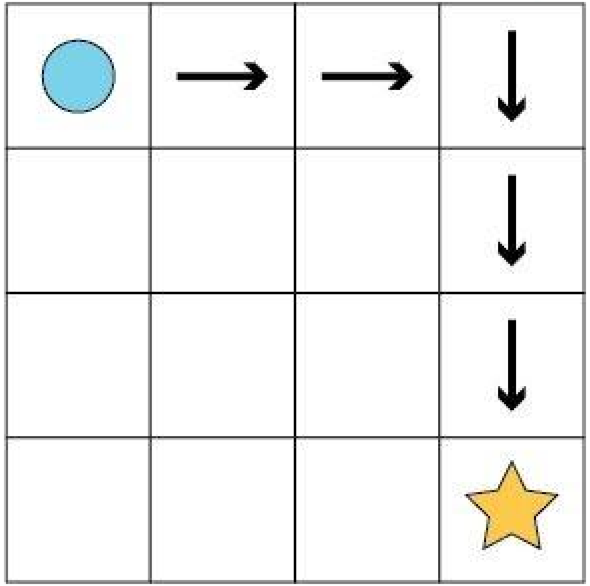


Discount factor (“don’t wait too long”)

Markov decision processes

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

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



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:

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

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

with its respective value function:

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

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

Markov decision processes

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

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

with its respective value function:

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

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

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

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

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

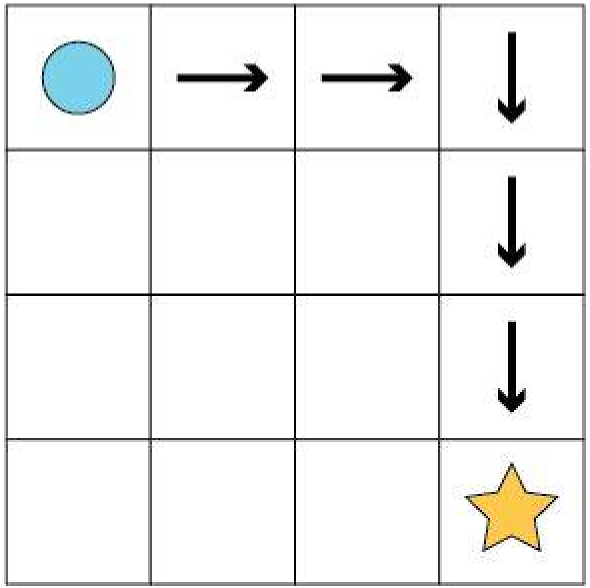
Behaviour policies

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

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

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

π^*



How do we find π^*?

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

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

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

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

$$\vdots$$

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

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

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

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

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

$$\vdots$$

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

Value Iteration

Bellman backup

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

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

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

$$\vdots$$

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

Value Iteration

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

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

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

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**

Value Iteration

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

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

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

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:

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

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

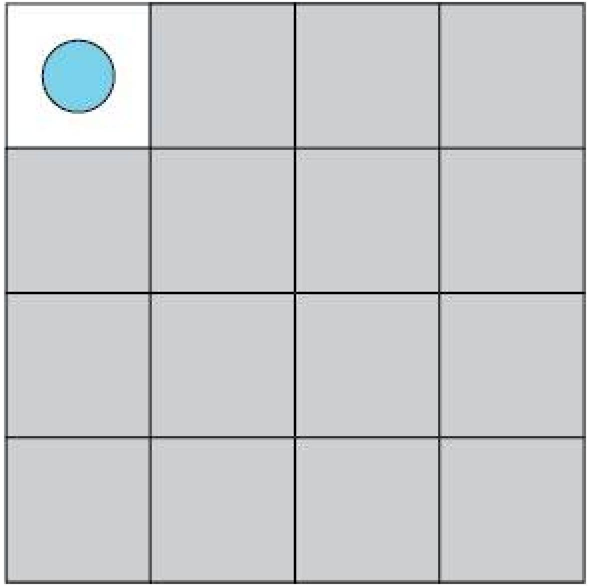
3. Return **π**

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 $s, a \rightarrow s', r$

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:

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:

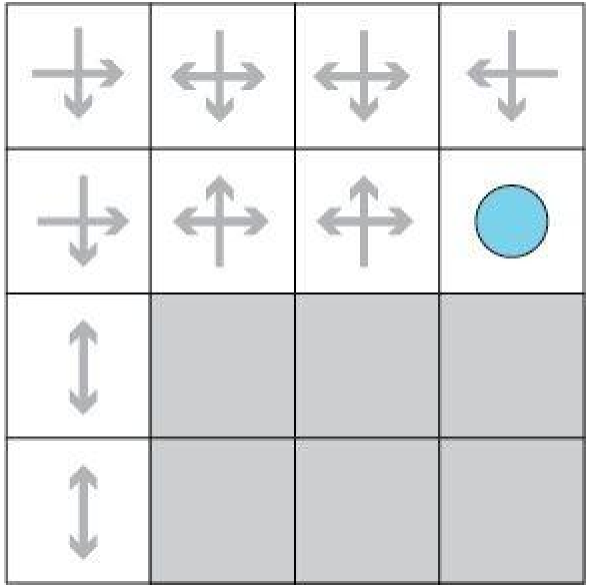
$$\delta = r + \gamma \max_{a' \in \mathcal{A}} Q(s', a') - Q(s, a)$$

   d. Update the estimates for Q:

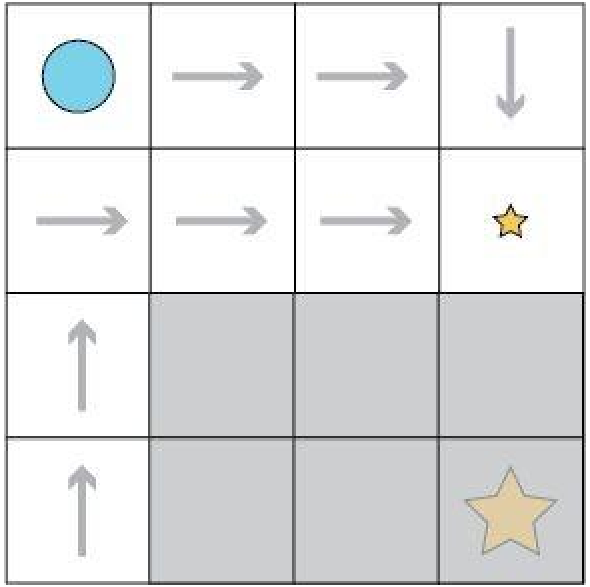
$$Q(s, a) = Q(s, a) + \alpha\delta$$

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

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



Exploration and Exploitation



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** **(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:

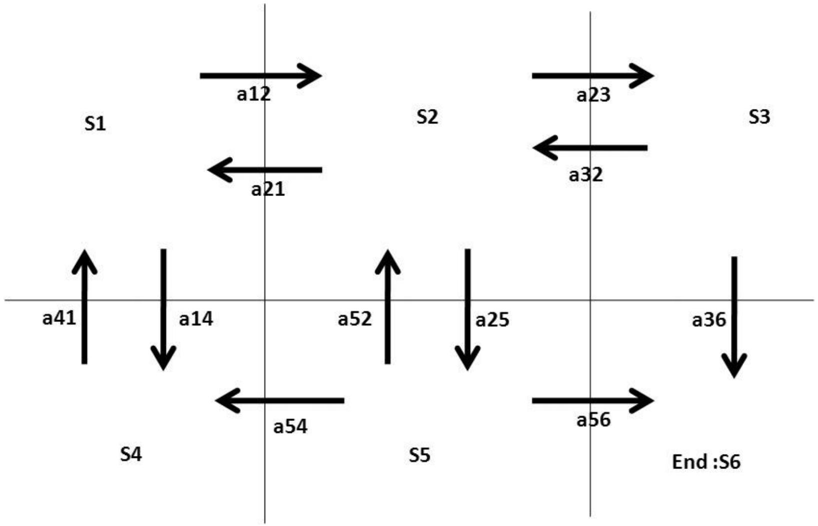
$$Q(s, a) = Q(s, a) + \alpha\delta$$

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

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

Q-learning- Example

Consider a modified maze problem
It is necessary to travel from box s1 to s6 in the best way available
The initial condition and the various travel paths are shown



Learning Automata

Defined as:

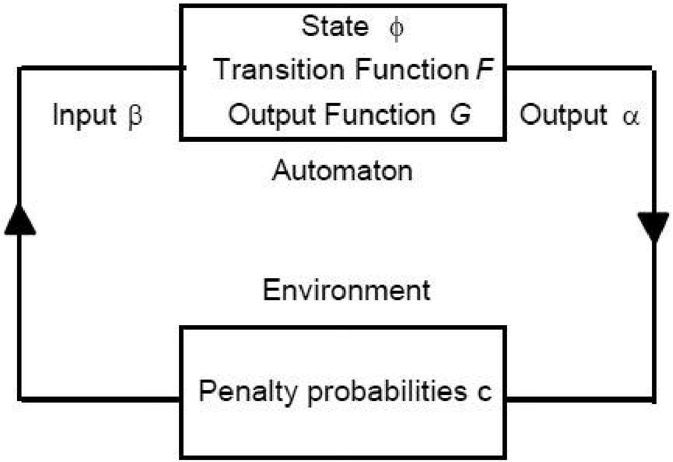
“The concept of learning automaton grew out as a fusion of works from psychologists, statisticians and operational researchers. The psychologists worked for modelling the observed behavior where the statisticians took effort to model the choice of experiments by considering the past observations, the operation researchers attempted to implement the optimal strategies in the context of the two-armed bandit problem. The system theorists group made rational decisions in random environments.”

Learning Automata

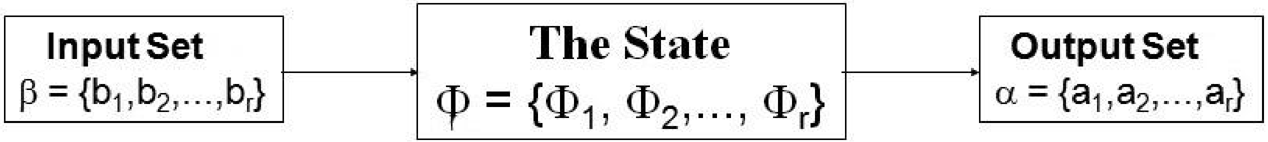
- Type of machine learning algorithm
- Select their current action based on past experiences from the environment
- Fall in the range of reinforcement learning
  - if the environment is stochastic
  - a Markov decision process (MDP) is used

Learning Automata

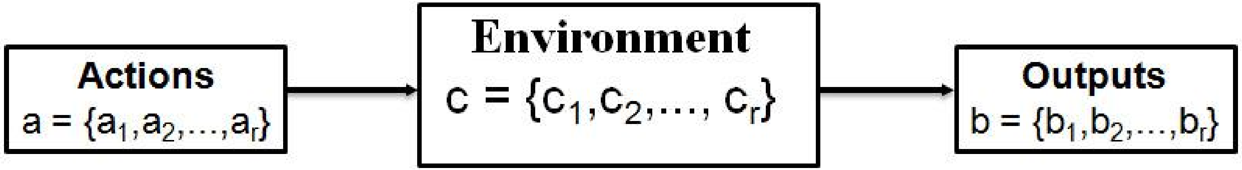
Automaton is a quintuple
- set of states
- set of actions
- inputs
- outputs
- transitions



Learning Automata



The Learning Automaton



The Environment

Learning is difficult?

The **blame attribution** problem is the problem of determining which action was responsible for a reward or punishment

The responsible action may have occurred a long time before the reward was received

The explore-exploit dilemma: if the agent has worked out a good course of actions, should it continue to follow these actions

Thank you
for your Attention!