Reinforcement Learning

CISC 7026 - Introduction to Deep Learning

Steven Morad

University of Macau

Lecture Goal: Provide a proper understanding of the theoretical foundations of reinforcement learning

~~**Lecture Goal**: Provide a proper understanding of the theoretical foundations of reinforcement learning~~

~~**Lecture Goal**: Provide a proper understanding of the theoretical foundations of reinforcement learning~~

Lecture Goal: Give you enough information to begin learning RL on your own

What is RL?

What is RL?

How does reinforcement learning (RL) differ from supervised or unsupervised learning?

What is RL?

How does reinforcement learning (RL) differ from supervised or unsupervised learning?

In supervised and unsupervised learning, we know the answer

What is RL?

How does reinforcement learning (RL) differ from supervised or unsupervised learning?

In supervised and unsupervised learning, we know the answer

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{y}$$

What is RL?

How does reinforcement learning (RL) differ from supervised or unsupervised learning?

In supervised and unsupervised learning, we know the answer

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{y}$$

$$f^{-1}(f(\boldsymbol{x}, \boldsymbol{\theta}), \boldsymbol{\theta}) = \boldsymbol{x}$$

What is RL?

How does reinforcement learning (RL) differ from supervised or unsupervised learning?

In supervised and unsupervised learning, we know the answer

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{y}$$

$$f^{-1}(f(\boldsymbol{x}, \boldsymbol{\theta}), \boldsymbol{\theta}) = \boldsymbol{x}$$

In reinforcement learning, we do not know the answer!

What is RL?

How does reinforcement learning (RL) differ from supervised or unsupervised learning?

In supervised and unsupervised learning, we know the answer

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{y}$$

$$f^{-1}(f(\boldsymbol{x}, \boldsymbol{\theta}), \boldsymbol{\theta}) = \boldsymbol{x}$$

In reinforcement learning, we do not know the answer!

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = ?$$

What is RL?

How does reinforcement learning (RL) differ from supervised or unsupervised learning?

In supervised and unsupervised learning, we know the answer

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{y}$$

$$f^{-1}(f(\boldsymbol{x}, \boldsymbol{\theta}), \boldsymbol{\theta}) = \boldsymbol{x}$$

In reinforcement learning, we do not know the answer!

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = ?$$

What does this mean?

What is RL?

Example: You train a model f to play chess

What is RL?

Example: You train a model f to play chess

$$f : X \times \Theta \mapsto Y$$

What is RL?

Example: You train a model f to play chess

$$f : X \times \Theta \mapsto Y$$

$$X \in \text{Position of pieces on the board}$$

What is RL?

Example: You train a model f to play chess

$$f : X \times \Theta \mapsto Y$$

$$X \in \text{Position of pieces on the board}$$

$$Y \in \text{Where to put piece}$$

What is RL?

Example: You train a model f to play chess

$$f : X \times \Theta \mapsto Y$$

$$X \in \text{Position of pieces on the board}$$

$$Y \in \text{Where to put piece}$$

What is RL?

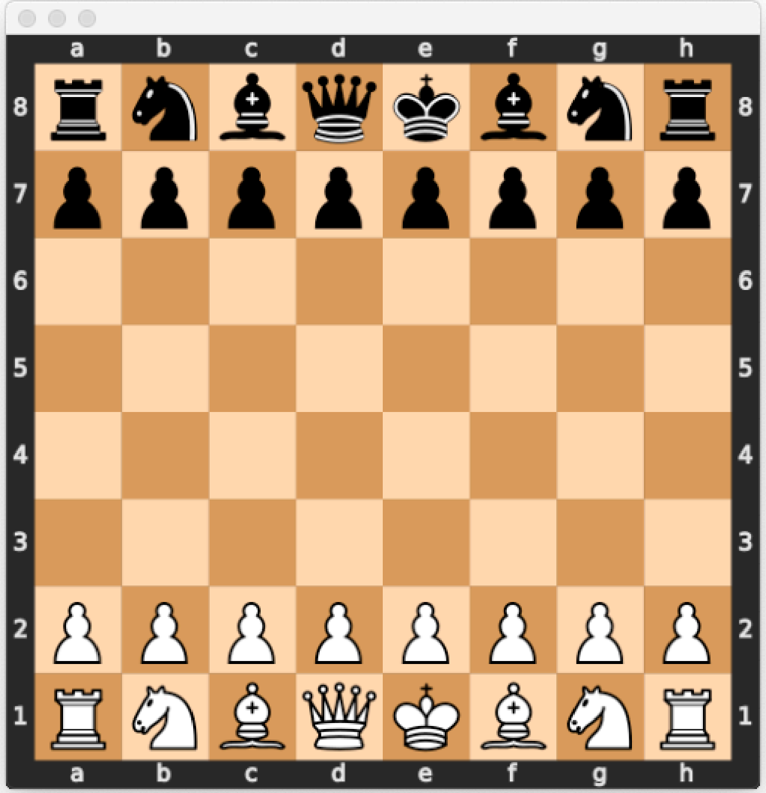
$X \in$ Position of pieces on the board $\quad$ $Y \in$ Where to put piece

What is RL?

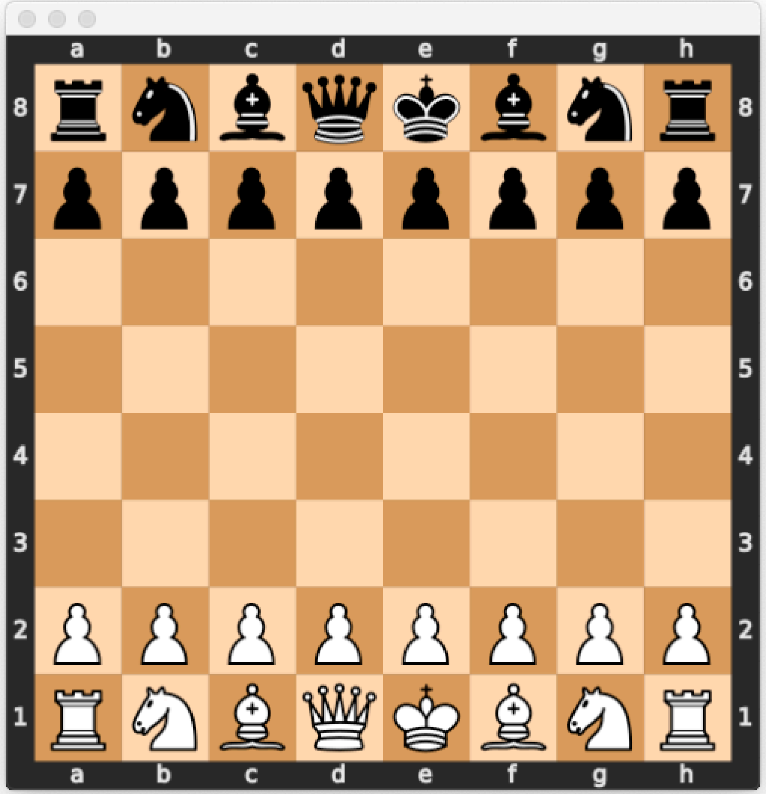
$X \in$ Position of pieces on the board $\qquad Y \in$ Where to put piece

What is RL?

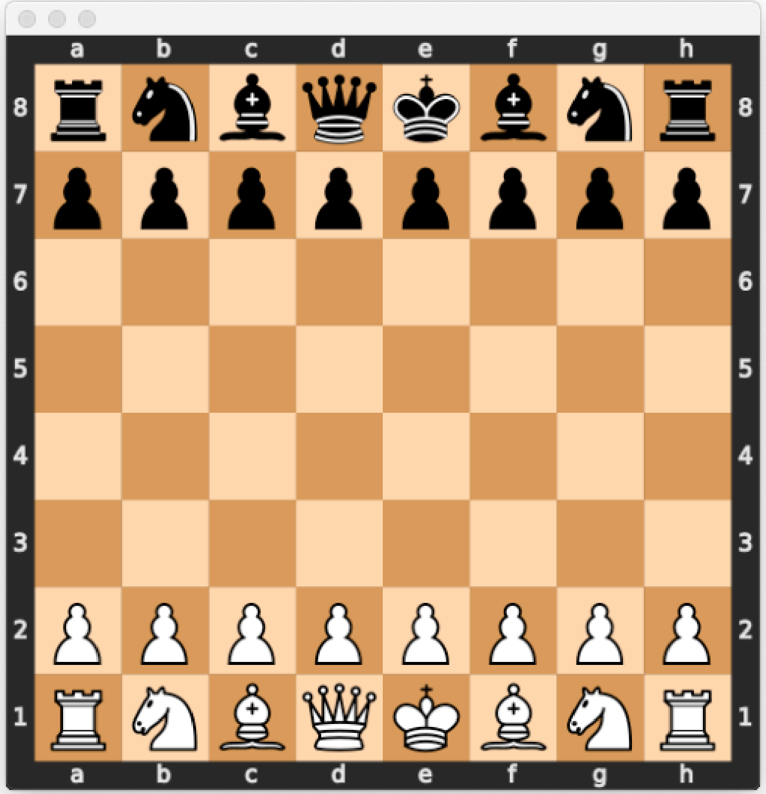
$X \in$ Position of pieces on the board $\qquad$ $Y \in$ Where to put piece



What is RL?

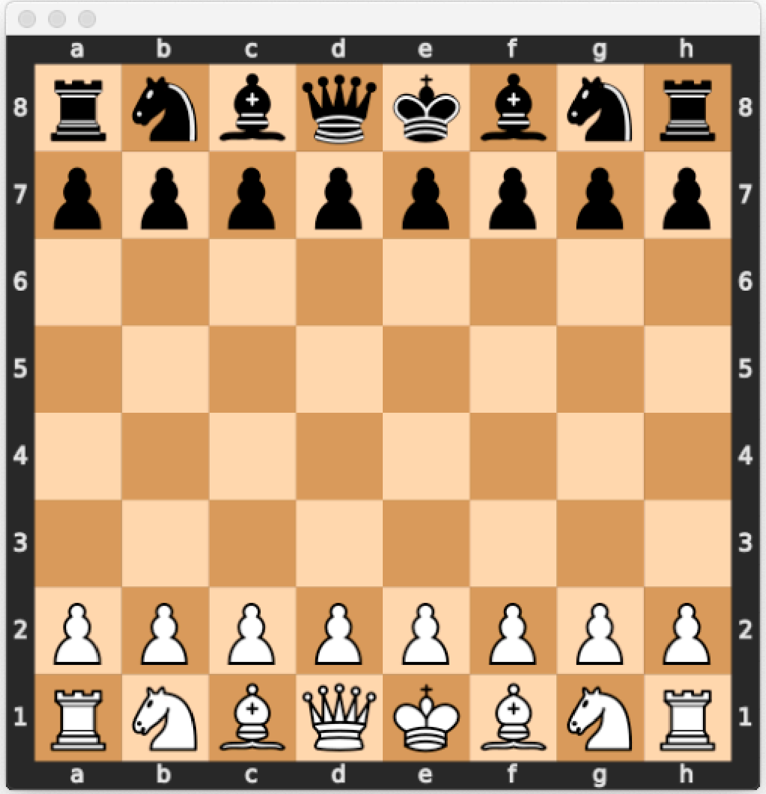


What is RL?



We do not know the answer

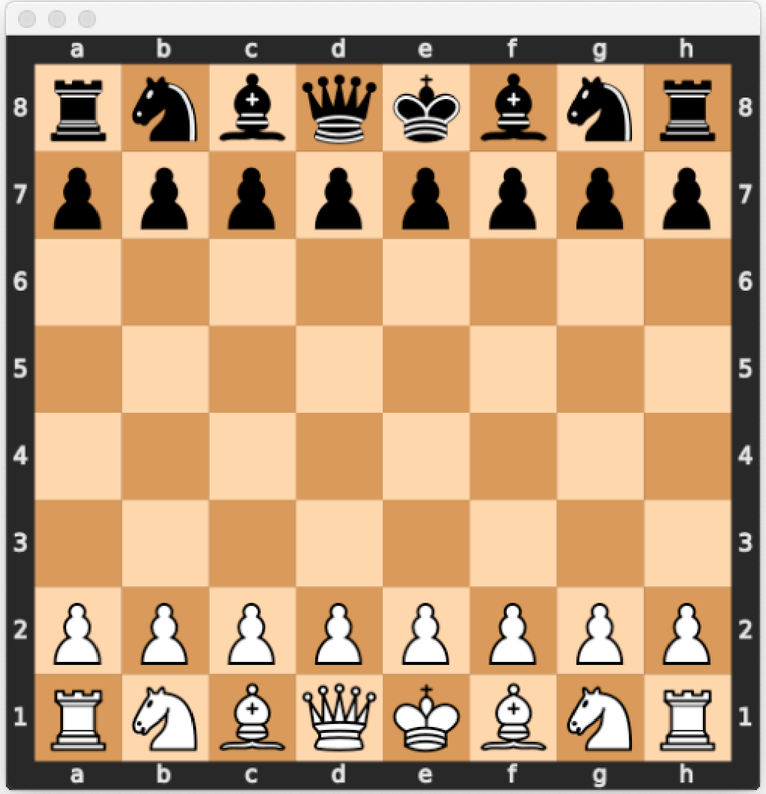
What is RL?



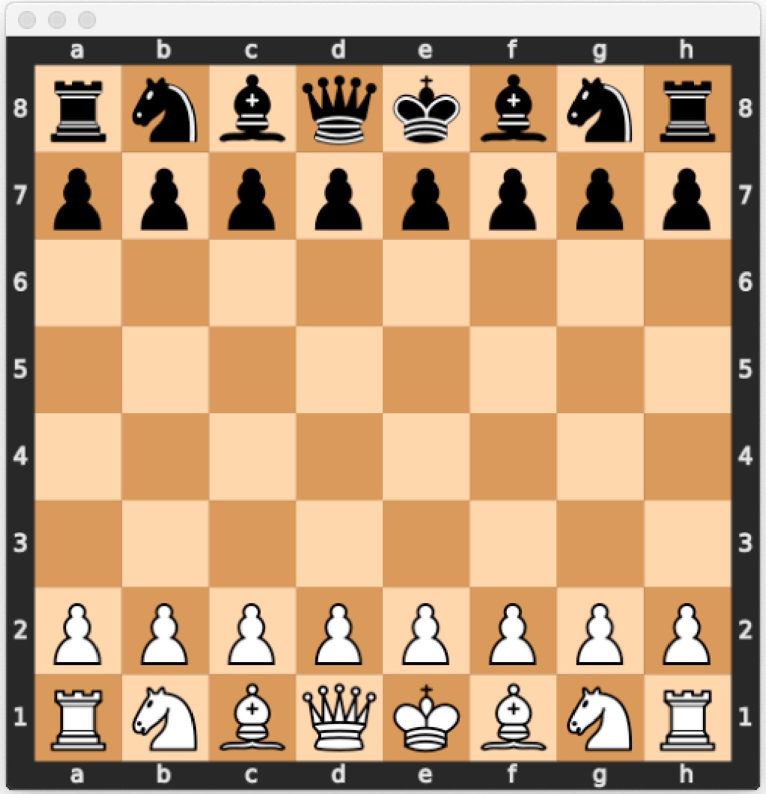
We do not know the answer

But RL can tell us!

What is RL?

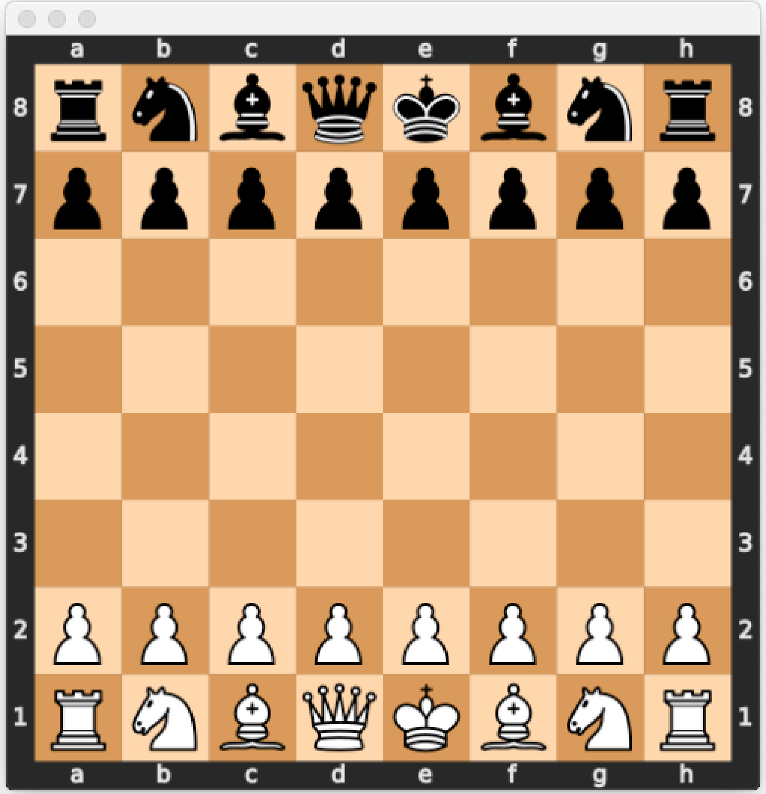


What is RL?



An answer gives us just one move

What is RL?



An answer gives us just one move

We need many moves to win

What is RL?

RL gives us the best sequence of moves to achieve a result

What is RL?

RL gives us the best sequence of moves to achieve a result

- Win a game of chess

What is RL?

RL gives us the best sequence of moves to achieve a result

- Win a game of chess
- Drive a customer to the store

What is RL?

RL gives us the best sequence of moves to achieve a result

- Win a game of chess
- Drive a customer to the store
- Cook a tasty meal

What is RL?

RL gives us the best sequence of moves to achieve a result

- Win a game of chess
- Drive a customer to the store
- Cook a tasty meal
- Surgically remove cancer from the patient

What is RL?

RL gives us the best sequence of moves to achieve a result

- Win a game of chess
- Drive a customer to the store
- Cook a tasty meal
- Surgically remove cancer from the patient
- Prevent climate change

What is RL?

RL gives us the best sequence of moves to achieve a result

- Win a game of chess
- Drive a customer to the store
- Cook a tasty meal
- Surgically remove cancer from the patient
- Prevent climate change
- Reduce human suffering

What is RL?

RL gives us the best sequence of moves to achieve a result

- Win a game of chess
- Drive a customer to the store
- Cook a tasty meal
- Surgically remove cancer from the patient
- Prevent climate change
- Reduce human suffering
- Find your own purpose (achieve conciousness)

What is RL?

https://www.youtube.com/watch?v=Zeyv1bN9v4A	GT
https://www.youtube.com/watch?v=kopoLzvh5jY&t=1s	H&S
https://www.youtube.com/watch?v=eHipy_j29Xw	DoTA

What is RL?

Real applications of RL:

What is RL?

Real applications of RL:

- Autonomous vehicles

What is RL?

Real applications of RL:

- Autonomous vehicles
- Video game NPCs

What is RL?

Real applications of RL:

- Autonomous vehicles
- Video game NPCs
- Behavior modeling in psychology/ecology/biology

What is RL?

Real applications of RL:

- Autonomous vehicles
- Video game NPCs
- Behavior modeling in psychology/ecology/biology
- Material and drug design

What is RL?

Real applications of RL:

- Autonomous vehicles
- Video game NPCs
- Behavior modeling in psychology/ecology/biology
- Material and drug design
- Finance

What is RL?

Real applications of RL:

- Autonomous vehicles
- Video game NPCs
- Behavior modeling in psychology/ecology/biology
- Material and drug design
- Finance
- Alignment in large language models

What is RL?

Real applications of RL:

- Autonomous vehicles
- Video game NPCs
- Behavior modeling in psychology/ecology/biology
- Material and drug design
- Finance
- Alignment in large language models
  - Artificial General Intelligence?

What is RL?

Real applications of RL:

- Autonomous vehicles
- Video game NPCs
- Behavior modeling in psychology/ecology/biology
- Material and drug design
- Finance
- Alignment in large language models
  - Artificial General Intelligence?
- Anywhere with cause and effect
  - Where you **change** the world by **interacting** with it

What is RL?

RL is more complicated than supervised learning

What is RL?

RL is more complicated than supervised learning

Instead of a model and dataset, we have an **agent** and **environment**

What is RL?

RL is more complicated than supervised learning

Instead of a model and dataset, we have an **agent** and **environment**

Agent

Environment

What is RL?

The agent receives a positive reward for doing good

What is RL?

The agent receives a positive reward for doing good

And a negative reward for doing bad

What is RL?

The agent receives a positive reward for doing good

And a negative reward for doing bad

What is RL?

The agent receives a positive reward for doing good

And a negative reward for doing bad

Eventually, the agent only does good behaviors

What is RL?

Humans learn by reinforcement learning too

What is RL?

Humans learn by reinforcement learning too

What is RL?

Humans learn by reinforcement learning too

When the baby cries, they will receive hugs (reward)

What is RL?

Humans learn by reinforcement learning too

When the baby cries, they will receive hugs (reward)

So the baby will learn to cry to get more hugs!

What is RL?

Humans learn by reinforcement learning too

When the baby cries, they will receive hugs (reward)

So the baby will learn to cry to get more hugs!

Note that “good” behavior is subjective!

What is RL?

Humans learn by reinforcement learning too

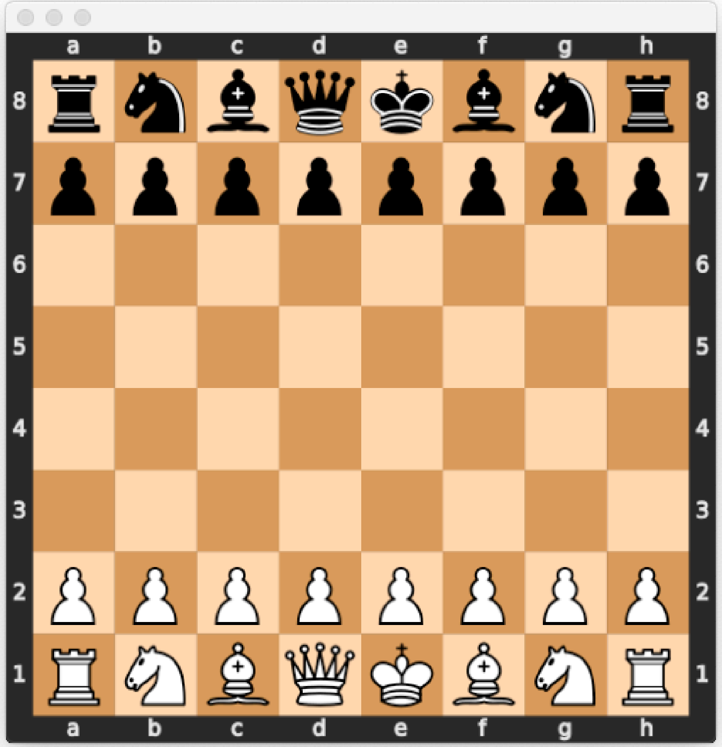
When the baby cries, they will receive hugs (reward)

So the baby will learn to cry to get more hugs!

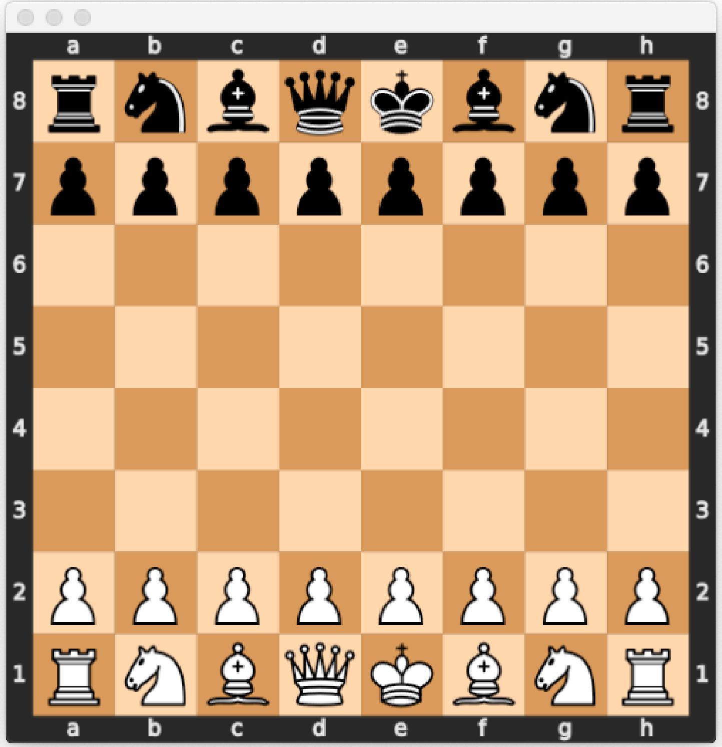
Note that "good" behavior is subjective!

Enough about the agent, let us talk about the environment

What is RL?

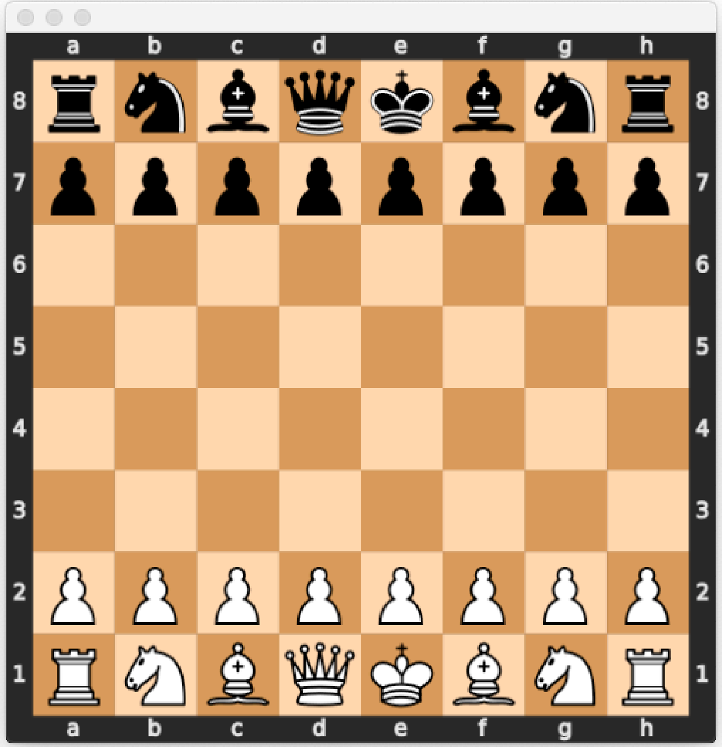


What is RL?



The environment is the world that the agent lives in

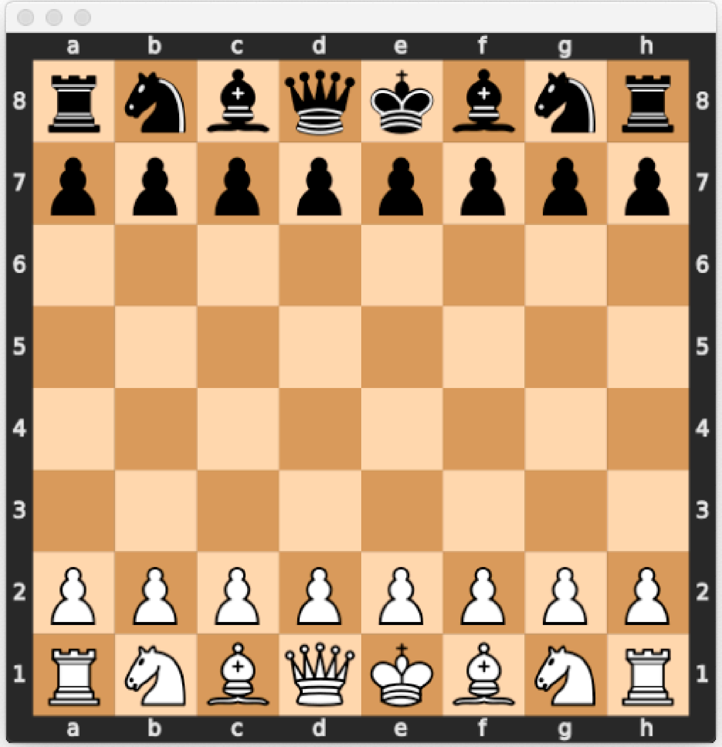
What is RL?



The environment is the world that the agent lives in

The environment is a collection of rules

What is RL?

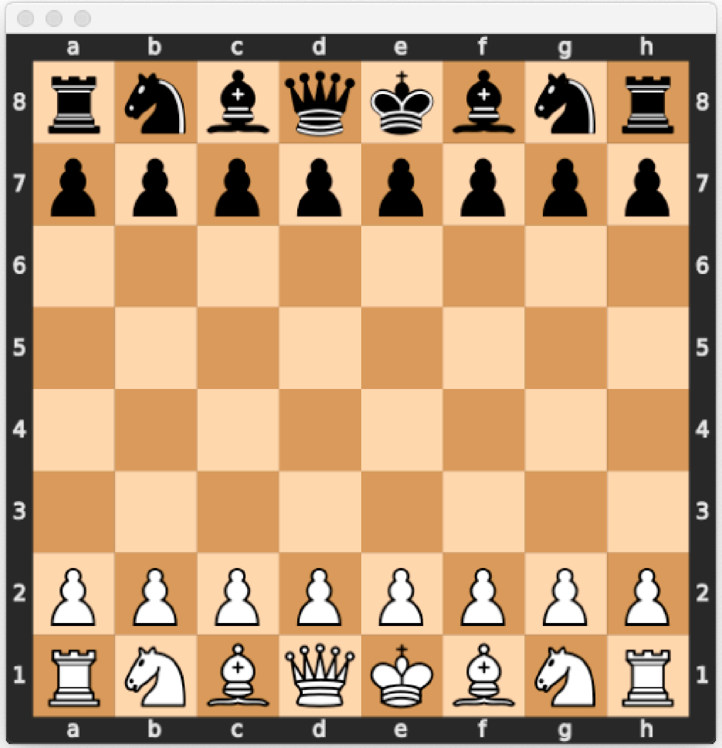


The environment is the world that the agent lives in

The environment is a collection of rules

For example, each piece can only move in certain ways

What is RL?



The environment is the world that the agent lives in

The environment is a collection of rules

For example, each piece can only move in certain ways

If two pieces touch, then one piece dies

What is RL?

For you, your environment is Macau!

What is RL?

For you, your environment is Macau!

There are a set of rules that govern what you can do

What is RL?

For you, your environment is Macau!

There are a set of rules that govern what you can do

- You follow the rules of physics (you cannot fly)

What is RL?

For you, your environment is Macau!

There are a set of rules that govern what you can do

- You follow the rules of physics (you cannot fly)
- You follow the laws (cannot steal)

What is RL?

For you, your environment is Macau!

There are a set of rules that govern what you can do

- You follow the rules of physics (you cannot fly)
- You follow the laws (cannot steal)
- You come to this specific location to attend lecture

What is RL?

For you, your environment is Macau!

There are a set of rules that govern what you can do

- You follow the rules of physics (you cannot fly)
- You follow the laws (cannot steal)
- You come to this specific location to attend lecture
- You get good grades (your parents make rules too)

What is RL?

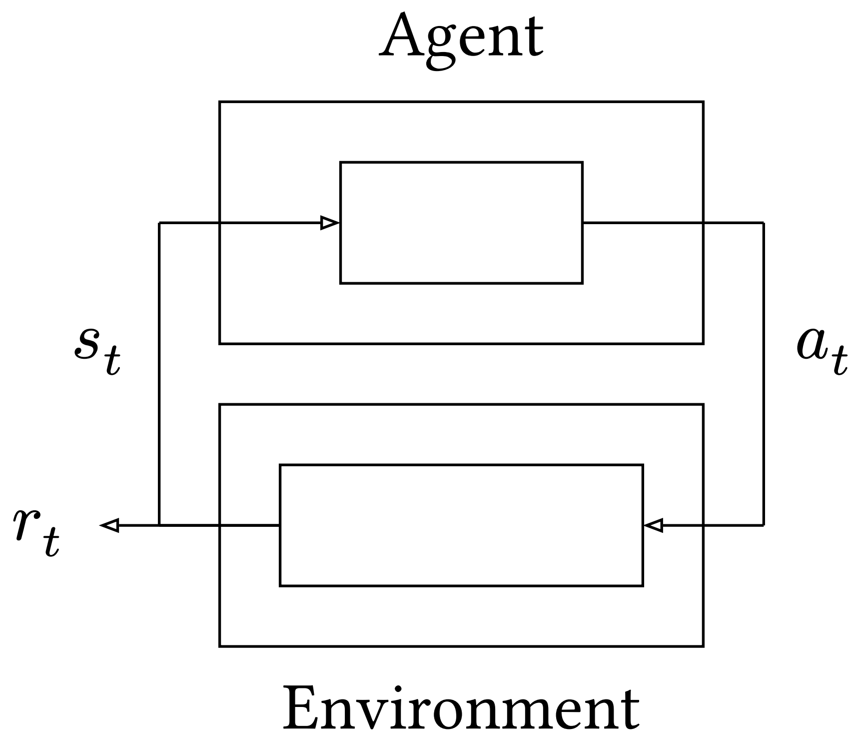
For you, your environment is Macau!

There are a set of rules that govern what you can do

- You follow the rules of physics (you cannot fly)
- You follow the laws (cannot steal)
- You come to this specific location to attend lecture
- You get good grades (your parents make rules too)

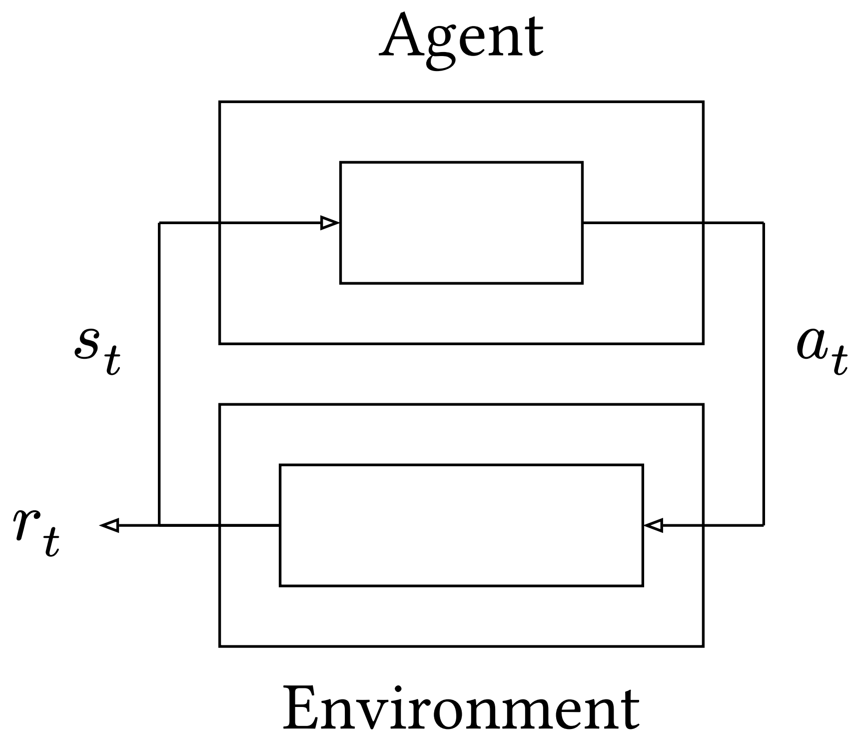
Now that you understand the agent, rewards, and environment, we will get more technical

What is RL?



s_t: state, a_t: action, r_t: reward

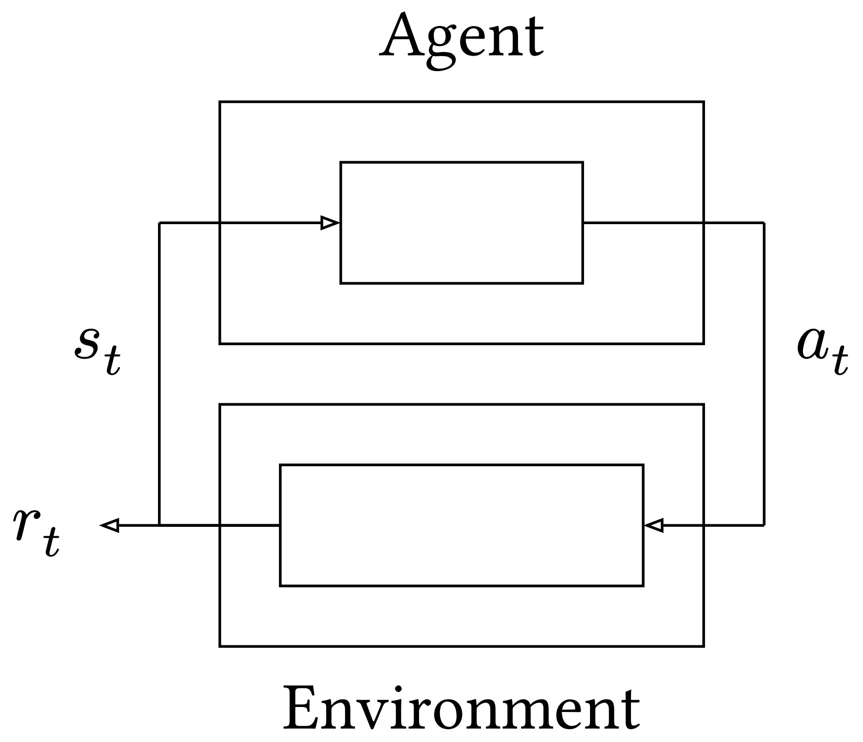
What is RL?



s_t: state, a_t: action, r_t: reward

- The agent takes **actions** in the environment

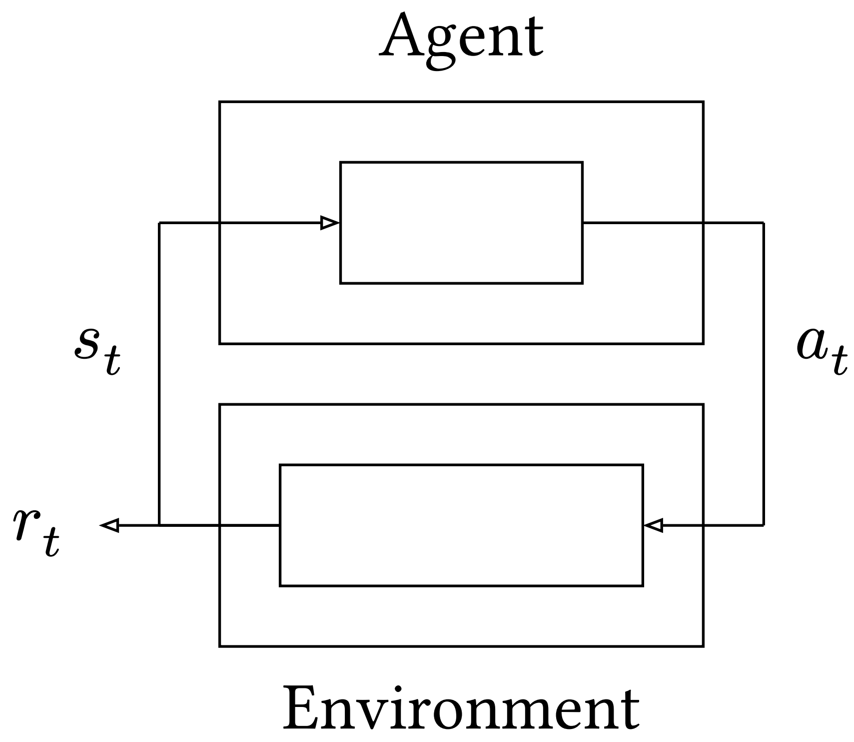
What is RL?



s_t: state, a_t: action, r_t: reward

- The agent takes **actions** in the environment
- Actions change the environment **state**, producing an new state and **reward**

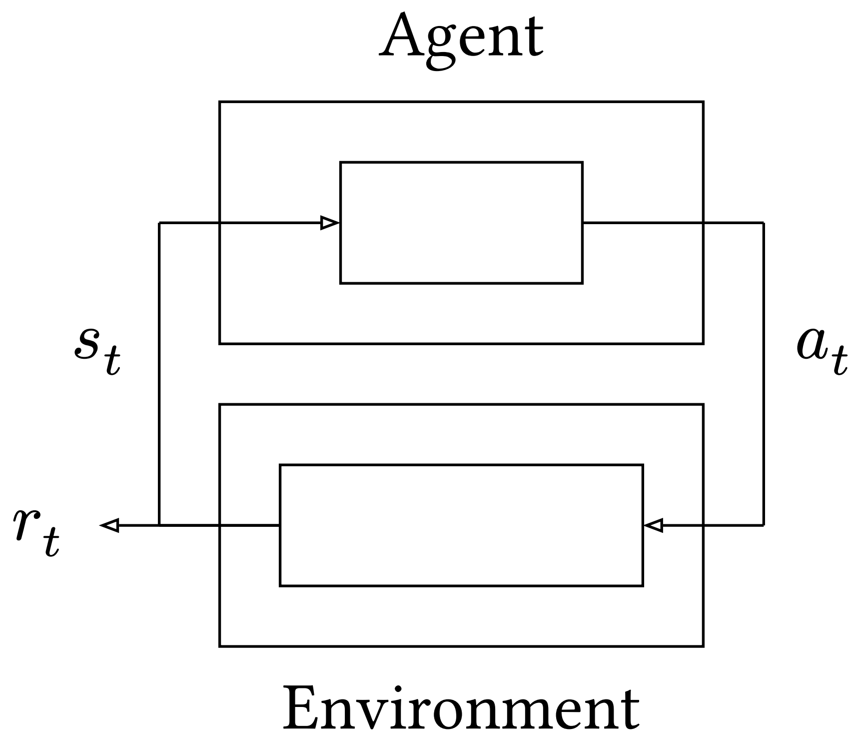
What is RL?



s_t: state, a_t: action, r_t: reward

- The agent takes **actions** in the environment
- Actions change the environment **state**, producing an new state and **reward**
- The cycle continues for $t = 0, 1, \ldots$

What is RL?



s_t: state, a_t: action, r_t: reward

- The agent takes **actions** in the environment
- Actions change the environment **state**, producing an new state and **reward**
- The cycle continues for $t = 0, 1, ...$
- Goal is to maximize the **cumulative reward**

What is RL?

By definition, RL solves **Markov Decision Processes (MDPs)**.

What is RL?

By definition, RL solves **Markov Decision Processes (MDPs)**.

To solve a problem using RL, we first must convert the problem into an MDP. We call this converted problem the **environment**.

What is RL?

By definition, RL solves **Markov Decision Processes (MDPs)**.

To solve a problem using RL, we first must convert the problem into an MDP. We call this converted problem the **environment**.

How you structure your problem is **critical** – more important than which algorithms you use, how much compute you have, etc.

Markov Decision Processes

Markov Decision Processes

Definition: An MDP is a 5-tuple (S, A, T, R, γ)

Markov Decision Processes

Definition: An MDP is a 5-tuple (S, A, T, R, γ)

S is the set of states known as the **state space**.

Markov Decision Processes

Definition: An MDP is a 5-tuple (S, A, T, R, γ)

S is the set of states known as the **state space**.

A is the set of actions known as the **action space**

Markov Decision Processes

Definition: An MDP is a 5-tuple (S, A, T, R, γ)

S is the set of states known as the **state space**.

A is the set of actions known as the **action space**

$T : S \times A \to \Delta S$ The **state transition function**.

Markov Decision Processes

Definition: An MDP is a 5-tuple (S, A, T, R, γ)

S is the set of states known as the **state space**.

A is the set of actions known as the **action space**

$T : S \times A \rightarrow \Delta S$ The **state transition function**.

$R : S \rightarrow \mathbb{R}$ is the **reward function**.

Markov Decision Processes

Definition: An MDP is a 5-tuple (S, A, T, R, γ)

S is the set of states known as the **state space**.

A is the set of actions known as the **action space**

$T : S \times A \rightarrow \Delta S$ The **state transition function**.

$R : S \rightarrow \mathbb{R}$ is the **reward function**.

$\gamma \in [0, 1]$ is the **discount factor**

Markov Decision Processes

Definition: An MDP is a 5-tuple (S, A, T, R, γ)

S is the set of states known as the **state space**.

A is the set of actions known as the **action space**

$T : S \times A \to \Delta S$ The **state transition function**.

$R : S \to \mathbb{R}$ is the **reward function**.

$\gamma \in [0, 1]$ is the **discount factor**

Let us briefly explain these terms.

Markov Decision Processes

S is the set of states known as the **state space**.

Markov Decision Processes

S is the set of states known as the **state space**.

Recall that the environment is always changing

Markov Decision Processes

S is the set of states known as the **state space**.

Recall that the environment is always changing

We need a way to describe what state the environment is in

Markov Decision Processes

S is the set of states known as the **state space**.

Recall that the environment is always changing

We need a way to describe what state the environment is in

If the environment is a table, the state space might describe the positions of all objects on the table

$$\boldsymbol{s} = \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ \vdots \end{bmatrix}$$

Markov Decision Processes

A is the set of actions known as the **action space**

Markov Decision Processes

A is the set of actions known as the **action space**

What capabilities does the agent have?

Markov Decision Processes

A is the set of actions known as the **action space**

What capabilities does the agent have?

For the table example, I can apply a force to a specific object on the table

$$\boldsymbol{a} = \begin{bmatrix} F_x \\ F_y \\ i \end{bmatrix}$$

Markov Decision Processes

$T : S \times A \to \Delta S$ The **state transition function**.

Markov Decision Processes

$T : S \times A \to \Delta S$ The **state transition function**.

Sometimes called “transition dynamics”, “state transition matrix”, etc

Markov Decision Processes

$T : S \times A \to \Delta S$ The **state transition function**.

Sometimes called “transition dynamics”, “state transition matrix”, etc

“Rules” of the environment, determine the (stochastic) evolution of the environment

Markov Decision Processes

$T : S \times A \to \Delta S$ The **state transition function**.

Sometimes called "transition dynamics", "state transition matrix", etc

"Rules" of the environment, determine the (stochastic) evolution of the environment

$$T\left(\underbrace{\begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ \vdots \end{bmatrix}}_{\text{state}}, \underbrace{\begin{bmatrix} F_x \\ F_y \\ i \end{bmatrix}}_{\text{action}}\right) = \underbrace{\Delta \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ \vdots \end{bmatrix}}_{\text{next state dist.}}$$

Markov Decision Processes

$T : S \times A \to \Delta S$ The **state transition function**.

Sometimes called "transition dynamics", "state transition matrix", etc

"Rules" of the environment, determine the (stochastic) evolution of the environment

$$T\left(\underbrace{\begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ \vdots \end{bmatrix}}_{\text{state}}, \underbrace{\begin{bmatrix} F_x \\ F_y \\ i \end{bmatrix}}_{\text{action}}\right) = \underbrace{\Delta \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ \vdots \end{bmatrix}}_{\text{next state dist.}}$$

This is a **Markov** decision process because transition dynamics are **conditionally independent** of past states and actions

$$T(s_t, a_t \mid s_{t-1}, a_{t-1}, ..., s_0, a_0) = T(s_t, a_t)$$

Markov Decision Processes

$R : S \to \mathbb{R}$ is the **reward function**.

Markov Decision Processes

$R : S \to \mathbb{R}$ is the **reward function**.

Produces reward based on the state

Markov Decision Processes

$R : S \to \mathbb{R}$ is the **reward function**.

Produces reward based on the state

Reward function determines agent behavior

Markov Decision Processes

$R : S \to \mathbb{R}$ is the **reward function**.

Produces reward based on the state

Reward function determines agent behavior

+100 for pushing objects onto the floor, or +100 for pushing objects to the centre

Markov Decision Processes

Definition: An MDP is a 5-tuple (S, A, T, R, γ)

Markov Decision Processes

Definition: An MDP is a 5-tuple (S, A, T, R, γ)

S is the set of states known as the **state space**.

Markov Decision Processes

Definition: An MDP is a 5-tuple (S, A, T, R, γ)

S is the set of states known as the **state space**.

A is the set of actions known as the **action space**

Markov Decision Processes

Definition: An MDP is a 5-tuple (S, A, T, R, γ)

S is the set of states known as the **state space**.

A is the set of actions known as the **action space**

$T : S \times A \to \Delta S$ The **state transition function**.

Markov Decision Processes

Definition: An MDP is a 5-tuple (S, A, T, R, γ)

S is the set of states known as the **state space**.

A is the set of actions known as the **action space**

$T : S \times A \to \Delta S$ The **state transition function**.

$R : S \to \mathbb{R}$ is the **reward function**.

Markov Decision Processes

Definition: An MDP is a 5-tuple (S, A, T, R, γ)

S is the set of states known as the **state space**.

A is the set of actions known as the **action space**

$T : S \times A \to \Delta S$ The **state transition function**.

$R : S \to \mathbb{R}$ is the **reward function**.

$\gamma \in [0, 1]$ is the **discount factor**

Markov Decision Processes

Super Mario Bros. is a video game about Mario, an Italian plumber

Markov Decision Processes

Super Mario Bros. is a video game about Mario, an Italian plumber

Mario can move and jump

Markov Decision Processes

Super Mario Bros. is a video game about Mario, an Italian plumber

Mario can move and jump

Touching a goomba kills Mario

Markov Decision Processes

Super Mario Bros. is a video game about Mario, an Italian plumber

Mario can move and jump

Touching a goomba kills Mario

Mario can squish Goombas

Markov Decision Processes

Super Mario Bros. is a video game about Mario, an Italian plumber

Mario can move and jump

Touching a goomba kills Mario

Mario can squish Goombas

? blocks give you mushrooms

Markov Decision Processes

Super Mario Bros. is a video game about Mario, an Italian plumber

Mario can move and jump

Touching a goomba kills Mario

Mario can squish Goombas

? blocks give you mushrooms

You collect coins and have a time limit and score

Markov Decision Processes

Super Mario Bros. is a video game about Mario, an Italian plumber

Mario can move and jump

Touching a goomba kills Mario

Mario can squish Goombas

? blocks give you mushrooms

You collect coins and have a time limit and score

Task: Define Super Mario MDP

Markov Decision Processes

State Space (S)?

Markov Decision Processes

State Space (S)?

- Mario position/velocity $(x, y, \dot{x}, \dot{y})$

Markov Decision Processes

State Space (S)?

- Mario position/velocity $(x, y, \dot{x}, \dot{y})$
- Score

Markov Decision Processes

State Space (S)?

- Mario position/velocity $(x, y, \dot{x}, \dot{y})$
- Score
- Number of coins collected

Markov Decision Processes

State Space (S)?

- Mario position/velocity $(x, y, \dot{x}, \dot{y})$
- Score
- Number of coins collected
- The time remaining

Markov Decision Processes

State Space (S)?

- Mario position/velocity $(x, y, \dot{x}, \dot{y})$
- Score
- Number of coins collected
- The time remaining
- Which question blocks we opened

Markov Decision Processes

State Space (S)?

- Mario position/velocity $(x, y, \dot{x}, \dot{y})$
- Score
- Number of coins collected
- The time remaining
- Which question blocks we opened
- Goomba position/velocity and squished/not squished

Markov Decision Processes

State Space (S)?

- Mario position/velocity $(x, y, \dot{x}, \dot{y})$
- Score
- Number of coins collected
- The time remaining
- Which question blocks we opened
- Goomba position/velocity and squished/not squished

$$S = \{\mathbb{R}^4, \mathbb{Z}_+, \mathbb{Z}_+, \mathbb{Z}_+, \{0, 1\}^m, \mathbb{R}^{4 \times k}, \{0, 1\}^k\}$$

Markov Decision Processes

State Space (S)?

Markov Decision Processes

State Space (S)? $[0, 1]^{2 \times 256 \times 240 \times 3}$

Markov Decision Processes

State Space (S)? $[0, 1]^{2 \times 256 \times 240 \times 3}$

$$
\begin{bmatrix} \begin{bmatrix} 255 \\ 0 \\ 0 \end{bmatrix} & \begin{bmatrix} 170 \\ 10 \\ 50 \end{bmatrix} & \dots \\ \begin{bmatrix} 10 \\ 100 \\ 235 \end{bmatrix} & \begin{bmatrix} 200 \\ 200 \\ 35 \end{bmatrix} & \dots \\ \vdots & & \ddots \end{bmatrix}, \begin{bmatrix} \begin{bmatrix} 255 \\ 0 \\ 0 \end{bmatrix} & \begin{bmatrix} 170 \\ 10 \\ 50 \end{bmatrix} & \dots \\ \begin{bmatrix} 10 \\ 100 \\ 235 \end{bmatrix} & \begin{bmatrix} 200 \\ 200 \\ 35 \end{bmatrix} & \dots \\ \vdots & & \ddots \end{bmatrix}
$$

Markov Decision Processes

State Space (S)? $[0, 1]^{2 \times 256 \times 240 \times 3}$

$$\begin{bmatrix} \begin{bmatrix} 255 \\ 0 \\ 0 \end{bmatrix} & \begin{bmatrix} 170 \\ 10 \\ 50 \end{bmatrix} & \dots \\ \begin{bmatrix} 10 \\ 100 \\ 235 \end{bmatrix} & \begin{bmatrix} 200 \\ 200 \\ 35 \end{bmatrix} & \dots \\ \vdots & & \ddots \end{bmatrix}, \begin{bmatrix} \begin{bmatrix} 255 \\ 0 \\ 0 \end{bmatrix} & \begin{bmatrix} 170 \\ 10 \\ 50 \end{bmatrix} & \dots \\ \begin{bmatrix} 10 \\ 100 \\ 235 \end{bmatrix} & \begin{bmatrix} 200 \\ 200 \\ 35 \end{bmatrix} & \dots \\ \vdots & & \ddots \end{bmatrix}$$

Two images necessary to compute velocities!

Markov Decision Processes

State Space (S)? $[0, 1]^{2 \times 256 \times 240 \times 3}$

$$\begin{bmatrix} \begin{bmatrix} 255 \\ 0 \\ 0 \end{bmatrix} & \begin{bmatrix} 170 \\ 10 \\ 50 \end{bmatrix} & \dots \\ \begin{bmatrix} 10 \\ 100 \\ 235 \end{bmatrix} & \begin{bmatrix} 200 \\ 200 \\ 35 \end{bmatrix} & \dots \\ \vdots & & \ddots \end{bmatrix}, \begin{bmatrix} \begin{bmatrix} 255 \\ 0 \\ 0 \end{bmatrix} & \begin{bmatrix} 170 \\ 10 \\ 50 \end{bmatrix} & \dots \\ \begin{bmatrix} 10 \\ 100 \\ 235 \end{bmatrix} & \begin{bmatrix} 200 \\ 200 \\ 35 \end{bmatrix} & \dots \\ \vdots & & \ddots \end{bmatrix}$$

Two images necessary to compute velocities!

$$S = \mathbb{Z}_{<255}^{2 \times 256 \times 240 \times 3}$$

Markov Decision Processes

Action Space (A)?

Markov Decision Processes

Action Space (A)?

- Acceleration of Mario $\ddot{x}$

Markov Decision Processes

Action Space (A)?

- Acceleration of Mario $\ddot{x}$
  - But when playing Mario, we cannot explicitly set $\ddot{x}$

Markov Decision Processes

Action Space (A)?

Markov Decision Processes

Action Space (A)?

- The Nintendo controller has $A, B, \uparrow, \downarrow, \leftarrow, \rightarrow$ buttons

Markov Decision Processes

Action Space (A)?

- The Nintendo controller has $A, B, \uparrow, \downarrow, \leftarrow, \rightarrow$ buttons
  - $A = \{A, B, \uparrow, \downarrow, \leftarrow, \rightarrow\}$

Markov Decision Processes

Action Space (A)?

- The Nintendo controller has $A, B, \uparrow, \downarrow, \leftarrow, \rightarrow$ buttons
  - $A = \{A, B, \uparrow, \downarrow, \leftarrow, \rightarrow\}$
    - Cannot represent multiple buttons at once

Markov Decision Processes

Action Space (A)?

- The Nintendo controller has $A, B, \uparrow, \downarrow, \leftarrow, \rightarrow$ buttons
  - $A = \{A, B, \uparrow, \downarrow, \leftarrow, \rightarrow\}$
    - Cannot represent multiple buttons at once
  - $A = \{0, 1\}^6$

Markov Decision Processes

Action Space (A)?

- The Nintendo controller has $A, B, \uparrow, \downarrow, \leftarrow, \rightarrow$ buttons
  - $A = \{A, B, \uparrow, \downarrow, \leftarrow, \rightarrow\}$
    - Cannot represent multiple buttons at once
  - $A = \{0, 1\}^6$
  - $$\left\{ \underbrace{\{0, 1, 2, 3, 4\}}_{\emptyset,\text{direction}} \times \underbrace{\{0, 1, 2, 3\}}_{\emptyset,\text{a,b,a+b}} \right\}$$

Markov Decision Processes

Transition Function (T)?

Markov Decision Processes

Transition Function (T)?

- $T(\text{pixel\_state}, \text{right})$

Markov Decision Processes

Transition Function (T)?

- $T(\text{pixel\_state}, \text{right})$
  - Move the Mario pixels right, unless a wall
  - Difficult to write down
  - Deterministic

Markov Decision Processes

Transition Function (T)?

Markov Decision Processes

Transition Function (T)?

- $T(\text{pos\_vel\_state}, \text{right})$

Markov Decision Processes

Transition Function (T)?

- $T(\text{pos\_vel\_state}, \text{right})$
  - Changes Mario's $(x, y, \dot{x}, \dot{y})$ in game memory
  - Human understandable, easier to implement for game developers

Markov Decision Processes

Question: In Mario, a single image frame is not a Markov state. How come?

Markov Decision Processes

Question: In Mario, a single image frame is not a Markov state. How come?

Answer: Cannot measure velocity.

Markov Decision Processes

Question: Why do we need velocity in the state?

Markov Decision Processes

Question: Why do we need velocity in the state?

Answer: If we don't have it, Markov property is violated

$T(s_t, a_t)$: Mario is going up, down, left or right

$T(s_t, a_t \mid s_{t-1})$: Mario is going right with velocity 1 m/s

Markov Decision Processes

Question: Why do we need velocity in the state?

Answer: If we don't have it, Markov property is violated

$T(s_t, a_t)$: Mario is going up, down, left or right

$T(s_t, a_t \mid s_{t-1})$: Mario is going right with velocity 1 m/s

Not conditionally independent!
$T(s_t, a_t \mid s_{t-1}, a_{t-1}, \ldots, s_0, a_0) \neq T(s_t, a_t)$

Markov Decision Processes

Reward (R)?

Markov Decision Processes

Reward (R)?

- 1 for beating the level and 0 otherwise

Markov Decision Processes

Reward (R)?

- 1 for beating the level and 0 otherwise
- Total score

Markov Decision Processes

Reward (R)?

- 1 for beating the level and 0 otherwise
- Total score
- 1 for beating the level + $0.01 \cdot$ score

Markov Decision Processes

- S✓
- A✓
- T✓
- R✓
- γ?

Markov Decision Processes

Agent goal in RL is to maximize the **cumulative** reward

Markov Decision Processes

Agent goal in RL is to maximize the **cumulative** reward

The cumulative reward is called the **return (G)**

$$G = \sum_{t=0}^{\infty} R(s_{t+1}) = \sum_{t=0}^{\infty} r_t$$

Markov Decision Processes

Agent goal in RL is to maximize the **cumulative** reward

The cumulative reward is called the **return (G)**

$$G = \sum_{t=0}^{\infty} R(s_{t+1}) = \sum_{t=0}^{\infty} r_t$$

Note that we care about all future rewards, not just the current reward!

Markov Decision Processes

Do humans maximize the return?

Markov Decision Processes

Do humans maximize the return?

Experiment: one cookie now, or two cookies in a year?

Markov Decision Processes

Do humans maximize the return?

Experiment: one cookie now, or two cookies in a year?

Usually, humans and animals prefer rewards sooner.

Markov Decision Processes

Do humans maximize the return?

Experiment: one cookie now, or two cookies in a year?

Usually, humans and animals prefer rewards sooner.

The **discount factor** γ injects this preference into RL. Results in the **discounted return (G)**

Markov Decision Processes

Do humans maximize the return?

Experiment: one cookie now, or two cookies in a year?

Usually, humans and animals prefer rewards sooner.

The **discount factor** γ injects this preference into RL. Results in the **discounted return (G)**

$$G = \sum_{t=0}^{\infty} \gamma^t r_t = r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots$$

$$0 \leq \gamma \leq 1$$

Markov Decision Processes

$$G = \sum_{t=0}^{\infty} \gamma^t r_t = r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots$$

$$0 \leq \gamma \leq 1$$

Markov Decision Processes

$$G = \sum_{t=0}^{\infty} \gamma^t r_t = r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots$$

$$0 \leq \gamma \leq 1$$

With a reward of 1 at each timestep and $\gamma = 0.9$

$$G = \sum_{t=0}^{\infty} \gamma^t r_t = 1 + 0.9 + 0.81 + \ldots = \frac{1}{1 - \gamma} = 10$$

Markov Decision Processes

$$G = \sum_{t=0}^{\infty} \gamma^t r_t = r_0 + \gamma r_1 + \gamma^2 r_2 + \dots$$

$$0 \leq \gamma \leq 1$$

With a reward of 1 at each timestep and $\gamma = 0.9$

$$G = \sum_{t=0}^{\infty} \gamma^t r_t = 1 + 0.9 + 0.81 + \dots = \frac{1}{1 - \gamma} = 10$$

We almost always choose to maximize the <u>discounted</u> return

Markov Decision Processes

Exercise: Reinforcement learning also describes human and animal behaviors. How can you describe your behavior using reinforcement learning?

Markov Decision Processes

Exercise: Reinforcement learning also describes human and animal behaviors. How can you describe your behavior using reinforcement learning?

Environment: City of Cambridge

Markov Decision Processes

Exercise: Reinforcement learning also describes human and animal behaviors. How can you describe your behavior using reinforcement learning?

Environment: City of Cambridge

State: My position, motivation, weather, and current activity

Markov Decision Processes

Exercise: Reinforcement learning also describes human and animal behaviors. How can you describe your behavior using reinforcement learning?

Environment: City of Cambridge

State: My position, motivation, weather, and current activity

Action Space: Either go to the Cambridge Blue or go home

Markov Decision Processes

Exercise: Reinforcement learning also describes human and animal behaviors. How can you describe your behavior using reinforcement learning?

Environment: City of Cambridge

State: My position, motivation, weather, and current activity

Action Space: Either go to the Cambridge Blue or go home

Reward Function: $100 \cdot \text{drink}_{\text{beer}} - \frac{1}{m} w_{\text{rain}}$

Markov Decision Processes

Exercise: Reinforcement learning also describes human and animal behaviors. How can you describe your behavior using reinforcement learning?

Environment: City of Cambridge

State: My position, motivation, weather, and current activity

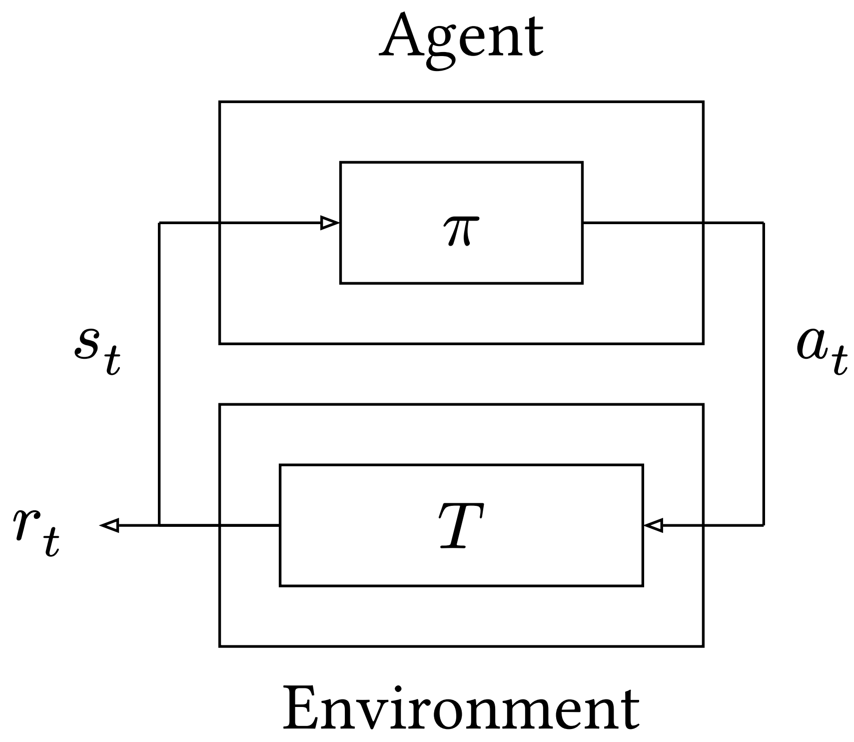
Action Space: Either go to the Cambridge Blue or go home

Reward Function: $100 \cdot \text{drink}_{\text{beer}} - \frac{1}{m} w_{\text{rain}}$

This happens internally when I decide to go to the pub after work

Agents and Policies

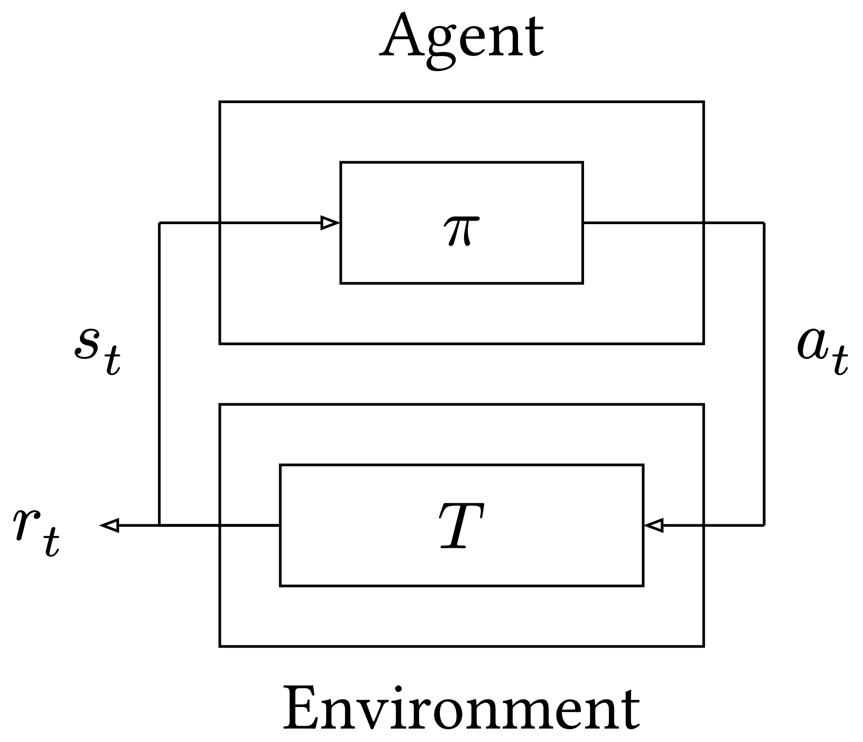
Agents and Policies



s_t: state, a_t: action,
r_t: reward, π: policy,
T: transition fn

- We have defined the environment

Agents and Policies



s_t: state, a_t: action,
r_t: reward, π: policy,
T: transition fn

- We have defined the environment
- Now let us define the agent

Agents and Policies

The agent acts following a **policy** π.

Agents and Policies

The agent acts following a **policy** π.

$\pi : S \rightarrow \Delta A$ is a mapping from states to actions (or action probabilities), determining agent behavior in the MDP.

Agents and Policies

The agent acts following a **policy** π.

$\pi : S \to \Delta A$ is a mapping from states to actions (or action probabilities), determining agent behavior in the MDP.

$$a_t \sim \pi(s_t)$$

$$\pi(a_t \mid s_t)$$

Agents and Policies

The policy that maximizes the discounted return (G) is called an **optimal policy (π_*)**

Agents and Policies

The policy that maximizes the discounted return (G) is called an **optimal policy (π_*)**

$$\pi_* = \max_{\pi} \sum_{t=0}^{\infty} \gamma^t r_t$$

Agents and Policies

The policy that maximizes the discounted return (G) is called an **optimal policy (π_*)**

$$\pi_* = \max_\pi \sum_{t=0}^{\infty} \gamma^t r_t$$

Even though the reward function is deterministic, the state transition function and policy influencing it are stochastic. The optimal policy must take this uncertainty into account.

Agents and Policies

The policy that maximizes the discounted return (G) is called an **optimal policy (π_*)**

$$\pi_* = \max_{\pi} \sum_{t=0}^{\infty} \gamma^t r_t$$

Even though the reward function is deterministic, the state transition function and policy influencing it are stochastic. The optimal policy must take this uncertainty into account.

$$r_t \sim \left[\underbrace{R(s_{t+1})}_{\text{reward fn}} \overbrace{T(s_{t+1} \mid s_t, a_t)}^{\text{state trans. probs}} \underbrace{\pi(a_t \mid s_t)}_{\text{action probs}} \right]$$

Agents and Policies

$$r_t \sim \left[\underbrace{R(s_{t+1})}_{\text{reward fn}} \overbrace{T(s_{t+1} \mid s_t, a_t)}^{\text{state trans. probs}} \underbrace{\pi(a_t \mid s_t)}_{\text{action probs}} \right]$$

(Many) RL researchers are afraid of distributions

Agents and Policies

$$r_t \sim \left[\underbrace{R(s_{t+1})}_{\text{reward fn}} \overbrace{T(s_{t+1} \mid s_t, a_t)}^{\text{state trans. probs}} \underbrace{\pi(a_t \mid s_t)}_{\text{action probs}} \right]$$

(Many) RL researchers are afraid of distributions

$$\mathbb{E}[r_t] = \int_S \int_A \underbrace{R(s_{t+1})}_{\text{reward fn}} \overbrace{T(s_{t+1} \mid s_t, a_t)}^{\text{state trans. probs}} \underbrace{\pi(a_t \mid s_t)}_{\text{action probs}} da_t ds_{t+1}$$

Agents and Policies

$$\pi_* = \max_{\pi} \sum_{t=0}^{\infty} \gamma^t r_t; \quad \mathbb{E}[r_t] = \int_S \int_A R(s_{t+1}) T(s_{t+1} \mid s_t, a_t) \pi(a_t \mid s_t) da_t ds_{t+1}$$

In English: We need to consider the action distribution combined with our state transition distribution when computing the reward/return

Agents and Policies

$$\pi_* = \max_\pi \sum_{t=0}^{\infty} \gamma^t r_t; \quad \mathbb{E}[r_t] = \int_S \int_A R(s_{t+1}) T(s_{t+1} \mid s_t, a_t) \pi(a_t \mid s_t) da_t ds_{t+1}$$

In English: We need to consider the action distribution combined with our state transition distribution when computing the reward/return

We write the return as the **expectation** given our policy actions.

$$\pi_* = \max_\pi \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t)\right]$$

Agents and Policies

$$\pi_* = \max_\pi \sum_{t=0}^{\infty} \gamma^t r_t; \quad \mathbb{E}[r_t] = \int_S \int_A R(s_{t+1}) T(s_{t+1} \mid s_t, a_t) \pi(a_t \mid s_t) da_t ds_{t+1}$$

In English: We need to consider the action distribution combined with our state transition distribution when computing the reward/return

We write the return as the **expectation** given our policy actions.

$$\pi_* = \max_\pi \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t)\right]$$

Now, our policy is truly optimal

Q Learning

Q Learning

We use **algorithms** to search for the optimal policy π_*

Q Learning

We use **algorithms** to search for the optimal policy π_*

Virtually all algorithms are based on either **Q Learning (QL)**, **Policy Gradient (PG)**, or both

Q Learning

We use **algorithms** to search for the optimal policy π_*

Virtually all algorithms are based on either **Q Learning (QL)**, **Policy Gradient (PG)**, or both

Popular algorithms:

- Deep Q Networks (DQN)

Q Learning

We use **algorithms** to search for the optimal policy π_*

Virtually all algorithms are based on either **Q Learning (QL)**, **Policy Gradient (PG)**, or both

Popular algorithms:

- Deep Q Networks (DQN)
- Proximal Policy Optimization (PPO)

Q Learning

We use **algorithms** to search for the optimal policy π_*

Virtually all algorithms are based on either **Q Learning (QL)**, **Policy Gradient (PG)**, or both

Popular algorithms:
- Deep Q Networks (DQN)
- Proximal Policy Optimization (PPO)
- Deep Deterministic Policy Gradient (DDPG)

Q Learning

We use **algorithms** to search for the optimal policy π_*

Virtually all algorithms are based on either **Q Learning (QL)**, **Policy Gradient (PG)**, or both

Popular algorithms:

- Deep Q Networks (DQN)
- Proximal Policy Optimization (PPO)
- Deep Deterministic Policy Gradient (DDPG)
- Twin Deep Deterministic Policy Gradient (TD3)

Q Learning

We use **algorithms** to search for the optimal policy π_*

Virtually all algorithms are based on either **Q Learning (QL)**, **Policy Gradient (PG)**, or both

Popular algorithms:

- Deep Q Networks (DQN)
- Proximal Policy Optimization (PPO)
- Deep Deterministic Policy Gradient (DDPG)
- Twin Deep Deterministic Policy Gradient (TD3)
- Soft Actor Critic (SAC)

Q Learning

We use **algorithms** to search for the optimal policy π_*

Virtually all algorithms are based on either **Q Learning (QL)**, **Policy Gradient (PG)**, or both

Popular algorithms:

- Deep Q Networks (DQN)
- Proximal Policy Optimization (PPO)
- Deep Deterministic Policy Gradient (DDPG)
- Twin Deep Deterministic Policy Gradient (TD3)
- Soft Actor Critic (SAC)
- Advantage Weighted Regression (AWR)

Q Learning

We use **algorithms** to search for the optimal policy π_*

Virtually all algorithms are based on either **Q Learning (QL)**, **Policy Gradient (PG)**, or both

Popular algorithms:
- Deep Q Networks (DQN)
- Proximal Policy Optimization (PPO)
- Deep Deterministic Policy Gradient (DDPG)
- Twin Deep Deterministic Policy Gradient (TD3)
- Soft Actor Critic (SAC)
- Advantage Weighted Regression (AWR)
- Asynchronous Actor Critic (A2C)

Q Learning

DQN: Q learning using a deep neural network

Q Learning

DQN: Q learning using a deep neural network

PPO: Policy gradient with update clipping and **Q/V** function

Q Learning

DQN: Q learning using a deep neural network

PPO: Policy gradient with update clipping and **Q/V** function

DDPG: Q learning with continuous actions via learned argmax

Q Learning

DQN: Q learning using a deep neural network

PPO: Policy gradient with update clipping and **Q/V** function

DDPG: Q learning with continuous actions via learned argmax

TD3: DDPG with action noise and a double **Q** trick

Q Learning

DQN: **Q** learning using a deep neural network

PPO: Policy gradient with update clipping and **Q/V** function

DDPG: **Q** learning with continuous actions via learned argmax

TD3: DDPG with action noise and a double **Q** trick

SAC: TD3 with entropy bonuses

Q Learning

DQN: **Q** learning using a deep neural network

PPO: Policy gradient with update clipping and **Q/V** function

DDPG: **Q** learning with continuous actions via learned argmax

TD3: DDPG with action noise and a double **Q** trick

SAC: TD3 with entropy bonuses

AWR: Offline policy gradient with **Q/V** function

Q Learning

DQN: **Q** learning using a deep neural network

PPO: Policy gradient with update clipping and **Q/V** function

DDPG: **Q** learning with continuous actions via learned argmax

TD3: DDPG with action noise and a double **Q** trick

SAC: TD3 with entropy bonuses

AWR: Offline policy gradient with **Q/V** function

A2C: Policy gradient with **Q/V** function

Q Learning

A theoretical understanding of Q learning is necessary, because algorithms build on top of Q learning

Q Learning

A theoretical understanding of Q learning is necessary, because algorithms build on top of Q learning

We will derive and define Q learning

Q Learning

A theoretical understanding of Q learning is necessary, because algorithms build on top of Q learning

We will derive and define Q learning

The Plan:

Q Learning

A theoretical understanding of Q learning is necessary, because algorithms build on top of Q learning

We will derive and define Q learning

The Plan:

1. Derive the value function V
2.
3.
4.

Q Learning

A theoretical understanding of Q learning is necessary, because algorithms build on top of Q learning

We will derive and define Q learning

The Plan:

1. Derive the value function V
2. Derive Q function from V
3.
4.

Q Learning

A theoretical understanding of Q learning is necessary, because algorithms build on top of Q learning

We will derive and define Q learning

The Plan:

1. Derive the value function V
2. Derive Q function from V
3. Derive an optimal policy from Q
4.

Q Learning

A theoretical understanding of Q learning is necessary, because algorithms build on top of Q learning

We will derive and define Q learning

The Plan:

1. Derive the value function V
2. Derive Q function from V
3. Derive an optimal policy from Q
4. Learn to train Q

Q Learning

Recall the discounted return for a specific policy π

$$G_\pi = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t)\right]$$

Q Learning

Recall the discounted return for a specific policy π

$$G_\pi = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t)\right]$$

In English: At each timestep, we take an action $a_t \sim \pi(s_t)$

Q Learning

Recall the discounted return for a specific policy π

$$G_\pi = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t)\right]$$

In English: At each timestep, we take an action $a_t \sim \pi(s_t)$

follow the state transition function $s_{t+1} \sim T(s_t, a_t)$

Q Learning

Recall the discounted return for a specific policy π

$$G_\pi = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t)\right]$$

In English: At each timestep, we take an action $a_t \sim \pi(s_t)$

follow the state transition function $s_{t+1} \sim T(s_t, a_t)$

and get a reward $r_t = R(s_{t+1})$

Q Learning

$$G_\pi = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t)\right]$$

Q Learning

$$G_\pi = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t)\right]$$

Question: Where does s_0 come from?

Q Learning

$$G_\pi = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t)\right]$$

Question: Where does s_0 come from?

Answer: Some higher power

Q Learning

$$G_\pi = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t)\right]$$

Question: Where does s_0 come from?

Answer: Some higher power

That is not a good answer

Q Learning

$$G_\pi = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t)\right]$$

What if we defined the return starting from a specific state?

Q Learning

$$G_\pi = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t)\right]$$

What if we defined the return starting from a specific state?

$$V_\pi(s_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t)\right]$$

Q Learning

$$G_\pi = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t)\right]$$

What if we defined the return starting from a specific state?

$$V_\pi(s_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t)\right]$$

Measures the value of a state (how good is it to be in this state?), for a given policy π

We call this the **Value Function (V_π)** $\quad V_\pi : S \to \mathbb{R}$

Q Learning

The Plan:

1. Derive the value function V
2. **Derive Q function from V**
3. Derive an optimal policy from Q
4. Learn to train Q

Q Learning

$$V_\pi(s_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t)\right]$$

Q Learning

$$V_\pi(s_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t)\right]$$

Let's go one step further. What if we parameterize the value function with an initial action?

Q Learning

$$V_\pi(s_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t)\right]$$

Let's go one step further. What if we parameterize the value function with an initial action?

Pull the first term out of the sum

Q Learning

$$V_\pi (s_0) = \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t) \right]$$

Let's go one step further. What if we parameterize the value function with an initial action?

Pull the first term out of the sum

$$V_\pi (s_0) = \mathbb{E}[r_0 \mid a_0 \sim \pi(s_0)] + \mathbb{E} \left[\sum_{t=1}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t) \right]$$

Q Learning

$$V_\pi(s_0) = \mathbb{E}[r_0 \mid a_0 \sim \pi(s_0)] + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t)\right]$$

Now, rewrite V_π as a function of an action a_0

Q Learning

$$V_\pi(s_0) = \mathbb{E}[r_0 \mid a_0 \sim \pi(s_0)] + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t)\right]$$

Now, rewrite V_π as a function of an action a_0

$$V_\pi(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t)\right]$$

Q Learning

$$V_\pi(s_0) = \mathbb{E}[r_0 \mid a_0 \sim \pi(s_0)] + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t)\right]$$

Now, rewrite V_π as a function of an action a_0

$$V_\pi(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t)\right]$$

When V depends on a specific action, we call it the **Q function**:

$$Q_\pi(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t)\right]$$

Q Learning

$$Q_\pi(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t)\right]$$

Q Learning

$$Q_\pi(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t)\right]$$

The Q function might appear simple but it is very powerful

Q Learning

$$Q_\pi(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t)\right]$$

The Q function might appear simple but it is very powerful

a_0 affects your next state s_1, which affects the future

Q Learning

$$Q_\pi(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t)\right]$$

Example: You have summer school offers from Cambridge and Oxford

Q Learning

$$Q_\pi(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t)\right]$$

Example: You have summer school offers from Cambridge and Oxford

$$a_0 = \{\text{Oxford}, \text{Cambridge}\}$$

Q Learning

$$Q_\pi(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t)\right]$$

Example: You have summer school offers from Cambridge and Oxford

$$a_0 = \{\text{Oxford}, \text{Cambridge}\}$$

Q function gives you a number denoting how much better your life will be for attending Cambridge (based on your behavior π). Takes into account reward (based on income, friend group, experiences, etc).

Q Learning

$$Q_\pi(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t)\right]$$

$$Q(s_0, \text{Cambridge}) = f(\text{friends} + \text{experiences} + \text{income}) = 1200$$

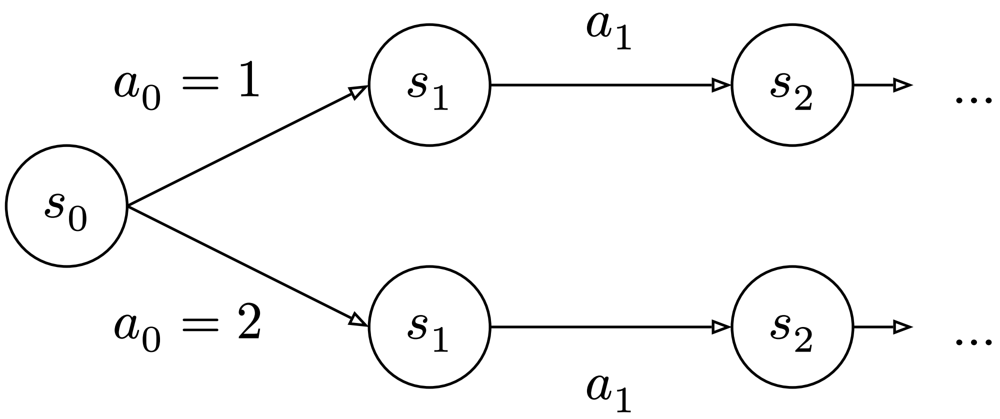
$$Q(s_0, \text{Oxford}) = f(\text{friends} + \text{experiences} + \text{income}) = 900$$

Q Learning

$$Q_\pi(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t)\right]$$

$$Q(s_0, \text{Cambridge}) = f(\text{friends} + \text{experiences} + \text{income}) = 1200$$

$$Q(s_0, \text{Oxford}) = f(\text{friends} + \text{experiences} + \text{income}) = 900$$



Q Learning

The Plan:

1. Derive the value function V
2. Derive Q function from V
3. **Derive an optimal policy from Q**
4. Learn to train Q

Q Learning

$$Q_\pi(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t)\right]$$

$$Q(s_0, \text{Cambridge}) = f(\text{friends} + \text{experiences} + \text{income}) = 1200$$

$$Q(s_0, \text{Oxford}) = f(\text{friends} + \text{experiences} + \text{income}) = 900$$

Q Learning

$$Q_\pi(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t)\right]$$

$$Q(s_0, \text{Cambridge}) = f(\text{friends} + \text{experiences} + \text{income}) = 1200$$

$$Q(s_0, \text{Oxford}) = f(\text{friends} + \text{experiences} + \text{income}) = 900$$

Question: Given the Q function, what would your policy be?

Q Learning

$$Q_\pi(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t)\right]$$

$$\boldsymbol{Q(s_0, \text{Cambridge}) = f(\text{friends} + \text{experiences} + \text{income}) = 1200}$$

$$Q(s_0, \text{Oxford}) = f(\text{friends} + \text{experiences} + \text{income}) = 900$$

Question: Given the Q function, what would your policy be?

Q Learning

Write this more formally

$$\pi(s) = \underset{a \in A}{\operatorname{argmax}} Q(s, a)$$

Q Learning

Write this more formally

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

Make it a degenerate distribution

$$\pi(s) = \mathrm{Deg}\left[\arg\max_{a \in A} Q(s, a)\right]$$

Q Learning

Write this more formally

$$\pi(s) = \underset{a \in A}{\operatorname{argmax}} Q(s, a)$$

Make it a degenerate distribution

$$\pi(s) = \operatorname{Deg}\left[\underset{a \in A}{\operatorname{argmax}} Q(s, a)\right]$$

We call this the **greedy policy**

Q Learning

$$\pi(s) = \mathrm{Deg}\left[\underset{a \in A}{\operatorname{argmax}} Q(s, a)\right]$$

Q Learning

$$\pi(s) = \mathrm{Deg}\left[\underset{a \in A}{\operatorname{argmax}} Q(s, a)\right]$$

The greedy policy is optimal (see Bellman Optimality Equation)

Q Learning

$$\pi(s) = \mathrm{Deg}\left[\underset{a \in A}{\operatorname{argmax}} Q(s, a)\right]$$

The greedy policy is optimal (see Bellman Optimality Equation)

$$\pi_*(s) = \mathrm{Deg}\left[\underset{a \in A}{\operatorname{argmax}} Q_*(s, a)\right]$$

Q Learning

$$\pi(s) = \mathrm{Deg}\left[\underset{a \in A}{\operatorname{argmax}} Q(s, a)\right]$$

The greedy policy is optimal (see Bellman Optimality Equation)

$$\pi_*(s) = \mathrm{Deg}\left[\underset{a \in A}{\operatorname{argmax}} Q_*(s, a)\right]$$

$$Q_*(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid a_t \sim \pi_*(s_t)\right]$$

Q Learning

$$\pi_*(s) = \mathrm{Deg}\left[\underset{a \in A}{\operatorname{argmax}} \, Q_*(s, a)\right]$$

Q Learning

$$\pi_*(s) = \mathrm{Deg}\left[\underset{a \in A}{\operatorname{argmax}} Q_*(s, a)\right]$$

$$\boldsymbol{Q_*(s_0, \textbf{Cambridge}) = f(\textbf{friends} + \textbf{experiences} + \textbf{income}) = 1200}$$

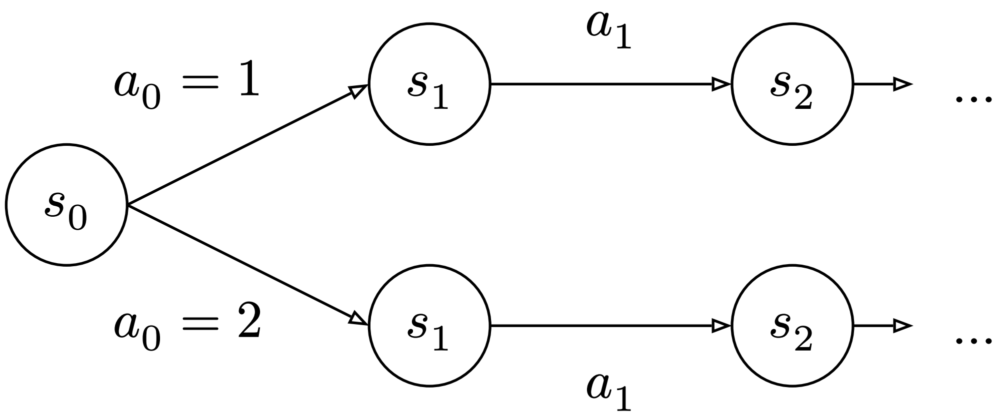
$$Q_*(s_0, \text{Oxford}) = f(\text{friends} + \text{experiences} + \text{income}) = 900$$

Q Learning

$$\pi_*(s) = \mathrm{Deg}\left[\underset{a \in A}{\operatorname{argmax}}\, Q_*(s, a)\right]$$

$$\boldsymbol{Q_*(s_0, \textbf{Cambridge}) = f(\textbf{friends} + \textbf{experiences} + \textbf{income}) = 1200}$$

$$Q_*(s_0, \mathrm{Oxford}) = f(\mathrm{friends} + \mathrm{experiences} + \mathrm{income}) = 900$$



Q Learning

The Plan:

1. Derive the value function V
2. Derive Q function from V
3. Derive an optimal policy from Q
4. **Learn to train Q**

Q Learning

$$Q_*(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid a_t \sim \pi_*(s_t)\right]$$

Q Learning

$$Q_*(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\underbrace{\sum_{t=1}^{\infty} \gamma^t r_t}_{\text{Very annoying}} \mid a_t \sim \pi_*(s_t) \right]$$

Q Learning

$$Q_*(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\underbrace{\sum_{t=1}^{\infty} \gamma^t r_t}_{\text{Very annoying}} \mid a_t \sim \pi_*(s_t)\right]$$

We need infinitely many rewards to approximate Q_*

Q Learning

$$Q_*(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\underbrace{\sum_{t=1}^{\infty} \gamma^t r_t}_{\text{Very annoying}} \mid a_t \sim \pi_*(s_t)\right]$$

We need infinitely many rewards to approximate Q_*

Can we get rid of the infinite sum?

Q Learning

$$Q_*(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid a_t \sim \pi_*(s_t)\right]$$

Q Learning

$$Q_*(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid a_t \sim \pi_*(s_t)\right]$$

Shift the sum so it starts at $t = 0$

$$Q_*(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^{t+1} r_{t+1} \mid a_{t+1} \sim \pi_*(s_{t+1})\right]$$

Q Learning

$$Q_*(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid a_t \sim \pi_*(s_t)\right]$$

Shift the sum so it starts at $t = 0$

$$Q_*(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^{t+1} r_{t+1} \mid a_{t+1} \sim \pi_*(s_{t+1})\right]$$

Factor out γ

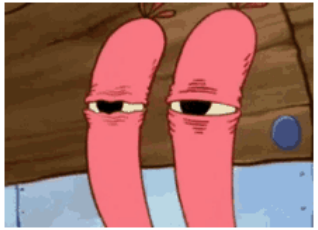
$$Q_*(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \gamma \cdot \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \mid a_{t+1} \sim \pi_*(s_{t+1})\right]$$

Q Learning

$$Q_*(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \gamma \cdot \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \mid a_{t+1} \sim \pi_*(s_{t+1})\right]$$

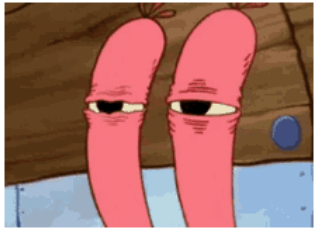
Q Learning

$$Q_*(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \gamma \cdot \underbrace{\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \mid a_{t+1} \sim \pi_*(s_{t+1})\right]}$$



Q Learning

$$Q_*(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \gamma \cdot \underbrace{\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \mid a_{t+1} \sim \pi_*(s_{t+1})\right]}$$



This is the value function for the policy π_* starting at s_1

$$Q_*(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \gamma \cdot V_*(s_1)$$

Q Learning

$$Q_*(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \gamma \cdot \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \mid a_{t+1} \sim \pi_*(s_{t+1})\right]$$

This is the value function for the policy π_* starting at s_1

$$Q_*(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \gamma \cdot V_*(s_1)$$

We can rewrite V as Q, removing the dependence on V

$$Q_*(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \gamma \cdot Q_*(s_1, \pi_*(s_1))$$

Q Learning

$$Q_*(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \gamma \cdot \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \mid a_{t+1} \sim \pi_*(s_{t+1})\right]$$

This is the value function for the policy π_* starting at s_1

$$Q_*(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \gamma \cdot V_*(s_1)$$

We can rewrite V as Q, removing the dependence on V

$$Q_*(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \gamma \cdot Q_*(s_1, \pi_*(s_1))$$

The policy π_* takes the argmax over Q, which reduces to

$$Q_*(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \gamma \cdot \max_{\{a \in A\}} Q_*(s_1, a)$$

Q Learning

$$Q_*(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \gamma \cdot \max_{\{a \in A\}} Q_*(s_1, a)$$

Q Learning

$$Q_*(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \gamma \cdot \max_{\{a \in A\}} Q_*(s_1, a)$$

Literature often drops the expectation and time subscripts

$$Q_*(s, a) = r + \gamma \cdot \max_{\{a' \in A\}} Q_*(s', a')$$

Q Learning

$$Q_*(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \gamma \cdot \max_{\{a \in A\}} Q_*(s_1, a)$$

Literature often drops the expectation and time subscripts

$$Q_*(s, a) = r + \gamma \cdot \max_{\{a' \in A\}} Q_*(s', a')$$

With the infinite sum gone, this is much easier to compute!

Q Learning

$$Q_*(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \gamma \cdot \max_{\{a \in A\}} Q_*(s_1, a)$$

Literature often drops the expectation and time subscripts

$$Q_*(s, a) = r + \gamma \cdot \max_{\{a' \in A\}} Q_*(s', a')$$

With the infinite sum gone, this is much easier to compute!

All we need is:

$$(s, a, r, \gamma, s')$$

Q Learning

We can parameterize Q with parameters θ and try to approximate Q_*

$$Q(s, a, \theta) = r + \gamma \cdot \max_{\{a' \in A\}} Q(s', a', \theta)$$

Q Learning

We can parameterize Q with parameters θ and try to approximate Q_*

$$Q(s, a, \theta) = r + \gamma \cdot \max_{\{a' \in A\}} Q(s', a', \theta)$$

Let us turn this into an optimization objective

Q Learning

We can parameterize Q with parameters θ and try to approximate Q_*

$$Q(s, a, \theta) = r + \gamma \cdot \max_{\{a' \in A\}} Q(s', a', \theta)$$

Let us turn this into an optimization objective

$$Q(s, a, \theta) - \left(r + \gamma \cdot \max_{\{a' \in A\}} Q(s', a', \theta) \right) = 0$$

Q Learning

We can parameterize Q with parameters θ and try to approximate Q_*

$$Q(s, a, \theta) = r + \gamma \cdot \max_{\{a' \in A\}} Q(s', a', \theta)$$

Let us turn this into an optimization objective

$$Q(s, a, \theta) - \left(r + \gamma \cdot \max_{\{a' \in A\}} Q(s', a', \theta) \right) = 0$$

$$\min_{\theta} \left(Q(s, a, \theta) - \left(r + \gamma \cdot \max_{\{a' \in A\}} Q(s', a', \theta) \right) \right)^2$$

Q Learning

We defined the Q function

$$Q(s, a, \theta) = r + \gamma \cdot \max_{\{a' \in A\}} Q(s', a', \theta)$$

Q Learning

We defined the Q function

$$Q(s, a, \theta) = r + \gamma \cdot \max_{\{a' \in A\}} Q(s', a', \theta)$$

We defined the optimal policy given the Q function

$$\pi(s) = \mathrm{Deg}\left[\operatorname{argmax}_{a \in A} Q(s, a, \theta)\right]$$

Q Learning

We defined the Q function

$$Q(s, a, \theta) = r + \gamma \cdot \max_{\{a' \in A\}} Q(s', a', \theta)$$

We defined the optimal policy given the Q function

$$\pi(s) = \mathrm{Deg}\left[\operatorname*{argmax}_{a \in A} Q(s, a, \theta)\right]$$

We defined the Q function training objective

$$\min_{\theta} \left(Q(s, a, \theta) - \left(r + \gamma \cdot \max_{\{a' \in A\}} Q(s', a', \theta) \right) \right)^2$$

Q Learning

Question: How do we model the Q function?

Q Learning

Question: How do we model the Q function?

Answer: Today, we use deep neural networks

Q Learning

Question: How do we model the Q function?

Answer: Today, we use deep neural networks

The Q learning approach we covered today is sufficient to learn superhuman policies on many video games

Q Learning

Question: How do we model the Q function?

Answer: Today, we use deep neural networks

The Q learning approach we covered today is sufficient to learn superhuman policies on many video games

https://www.youtube.com/watch?v=O2QaSh4tNVw	SMB
https://youtu.be/VIwGxOdXGfw?si=A-CVLI6vEJHOxrvx&t=478	MK

Q Learning

1. Reinforcement Learning, an Introduction (2018, Sutton and Barto)
   - Available for free online (legal)
   - All the RL theory you will ever need
2.
3.
4.

Q Learning

1. Reinforcement Learning, an Introduction (2018, Sutton and Barto)
   - Available for free online (legal)
   - All the RL theory you will ever need

2. David Silver's slides for his RL course at UCL
   - Builds good intuition

3.

4.

Q Learning

1. Reinforcement Learning, an Introduction (2018, Sutton and Barto)
   - Available for free online (legal)
   - All the RL theory you will ever need
2. David Silver's slides for his RL course at UCL
   - Builds good intuition
3. OpenAI Spinning Up
   - Mixes theory with implementation
4.

Q Learning

1. Reinforcement Learning, an Introduction (2018, Sutton and Barto)
    - Available for free online (legal)
    - All the RL theory you will ever need
2. David Silver's slides for his RL course at UCL
    - Builds good intuition
3. OpenAI Spinning Up
    - Mixes theory with implementation
4. CleanRL
    - Verified, single-file implementations of many RL algorithms

Q Learning

1. Reinforcement Learning, an Introduction (2018, Sutton and Barto)
   - Available for free online (legal)
   - All the RL theory you will ever need

2. David Silver's slides for his RL course at UCL
   - Builds good intuition

3. OpenAI Spinning Up
   - Mixes theory with implementation

4. CleanRL
   - Verified, single-file implementations of many RL algorithms

Questions? sm2558@cam.ac.uk