Markov Decision Processes

Deep Reinforcement Learning

University of Cambridge

Overview

1. Lecture (approx. 1 hour)
   1. Review
   2. Finish up MDPs
   3. Agents/policies
   4. Derive and define Q learning
2. Discussion/questions (approx. 30 mins)

Overview

1. Lecture (approx. 1 hour)
   1. **Review**
   2. Finish up MDPs
   3. Agents/policies
   4. Derive and define Q learning
2. Discussion/questions (approx. 30 mins)

Review

Last time we talked about MDPs

Review

Last time we talked about MDPs

Designed state and action spaces for Mario

Review

Last time we talked about MDPs

Designed state and action spaces for Mario

Let's briefly review the MDP

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**

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

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

$\gamma \in [0, 1]$ is a discount factor that we will explain later

State Transition Function

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

State Transition Function

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

$$T\left(\underbrace{\begin{bmatrix} x_1 & y_1 & x_2 & y_2 & \dots \end{bmatrix}}_{\text{state}}, \underbrace{\begin{bmatrix} F_x & F_y & i \end{bmatrix}}_{\text{action}}\right) = \underbrace{\Delta\begin{bmatrix} x_1 & y_1 & x_2 & y_2 & \dots \end{bmatrix}}_{\text{new state}}$$

State Transition Function

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

$$T\left(\underbrace{\begin{bmatrix} x_1 & y_1 & x_2 & y_2 & \dots \end{bmatrix}}_{\text{state}}, \underbrace{\begin{bmatrix} F_x & F_y & i \end{bmatrix}}_{\text{action}}\right) = \underbrace{\Delta\begin{bmatrix} x_1 & y_1 & x_2 & y_2 & \dots \end{bmatrix}}_{\text{new state}}$$

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

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

State Transition Function

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

$$T\left(\underbrace{\begin{bmatrix} x_1 & y_1 & x_2 & y_2 & \dots \end{bmatrix}}_{\text{state}}, \underbrace{\begin{bmatrix} F_x & F_y & i \end{bmatrix}}_{\text{action}}\right) = \underbrace{\Delta\begin{bmatrix} x_1 & y_1 & x_2 & y_2 & \dots \end{bmatrix}}_{\text{new state}}$$

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

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

If conditional independence is violated, the process is **not** Markov

State Transition Function

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

$$T\left(\underbrace{\begin{bmatrix} x_1 & y_1 & x_2 & y_2 & \dots \end{bmatrix}}_{\text{state}}, \underbrace{\begin{bmatrix} F_x & F_y & i \end{bmatrix}}_{\text{action}} \right) = \underbrace{\Delta \begin{bmatrix} x_1 & y_1 & x_2 & y_2 & \dots \end{bmatrix}}_{\text{new state}}$$

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

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

If conditional independence is violated, the process is **not** Markov

$$T(s_t, a_t) \neq T(s_t, a_t \mid s_{t-1}) \Longleftarrow \text{Not Markov!}$$

Super Mario Bros

- **State Space (S)?**

Super Mario Bros

- **State Space (S)**?
  - The position and velocity $(x, y, \dot{x}, \dot{y})$ of Mario and Goombas
  - The score
  - Number of coins collected
  - The time remaining
  - Which question blocks have been opened
  - Which goombas have been squished

Super Mario Bros

- **State Space (S)?**
  - The position and velocity $(x, y, \dot{x}, \dot{y})$ of Mario and Goombas
  - The score
  - Number of coins collected
  - The time remaining
  - Which question blocks have been opened
  - Which goombas have been squished

$$S = \left\{\mathbb{R}^4, \mathbb{R}^4, ..., \mathbb{Z}_+, \mathbb{Z}_+, \mathbb{Z}_+, \{0, 1\}^k\right\}$$

Super Mario Bros

- **State Space (S)?**

Super Mario Bros

- **State Space (S)?**

2x256x240x3 pixels.

$$\text{E.g.} \begin{pmatrix} \begin{pmatrix} 255 \\ 0 \\ 0 \end{pmatrix} & \begin{pmatrix} 170 \\ 10 \\ 50 \end{pmatrix} & \dots \\ \begin{pmatrix} 10 \\ 100 \\ 235 \end{pmatrix} & \begin{pmatrix} 200 \\ 200 \\ 35 \end{pmatrix} & \dots \\ \vdots & & \ddots \end{pmatrix}, \begin{pmatrix} \begin{pmatrix} 255 \\ 0 \\ 0 \end{pmatrix} & \begin{pmatrix} 170 \\ 10 \\ 50 \end{pmatrix} & \dots \\ \begin{pmatrix} 10 \\ 100 \\ 235 \end{pmatrix} & \begin{pmatrix} 200 \\ 200 \\ 35 \end{pmatrix} & \dots \\ \vdots & & \ddots \end{pmatrix}$$

Super Mario Bros

- **State Space (S)?**

2x256x240x3 pixels.

E.g. $$\begin{pmatrix} \begin{pmatrix} 255 \\ 0 \\ 0 \end{pmatrix} & \begin{pmatrix} 170 \\ 10 \\ 50 \end{pmatrix} & \cdots \\ \begin{pmatrix} 10 \\ 100 \\ 235 \end{pmatrix} & \begin{pmatrix} 200 \\ 200 \\ 35 \end{pmatrix} & \cdots \\ \vdots & & \ddots \end{pmatrix}, \begin{pmatrix} \begin{pmatrix} 255 \\ 0 \\ 0 \end{pmatrix} & \begin{pmatrix} 170 \\ 10 \\ 50 \end{pmatrix} & \cdots \\ \begin{pmatrix} 10 \\ 100 \\ 235 \end{pmatrix} & \begin{pmatrix} 200 \\ 200 \\ 35 \end{pmatrix} & \cdots \\ \vdots & & \ddots \end{pmatrix}$$

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

Super Mario Bros

- **Action Space (A)?**

Super Mario Bros

- **Action Space (A)?**
  - Acceleration of Mario $\ddot{x}$

Super Mario Bros

- **Action Space (A)**?
  - Acceleration of Mario $\ddot{x}$
    - But when playing Mario, we cannot explicitly set $\ddot{x}$

Super Mario Bros

- **Action Space (A)?**

Super Mario Bros

- **Action Space (A)**?
  - The Nintendo controller has A, B, up, down, left, right buttons
    - $A = \{A, B, \text{up}, \text{down}, \text{left}, \text{right}\}$

Super Mario Bros

- **Action Space (A)**?
  - The Nintendo controller has A, B, up, down, left, right buttons
    - $A = \{A, B, \text{up}, \text{down}, \text{left}, \text{right}\}$
      - Cannot represent pressing multiple buttons at once

Super Mario Bros

- **Action Space (A)**?
  - The Nintendo controller has A, B, up, down, left, right buttons
    - $A = \{A, B, \text{up}, \text{down}, \text{left}, \text{right}\}$
      - Cannot represent pressing multiple buttons at once
    - $A = \{0, 1\}^6$

Super Mario Bros

- **Action Space (A)?**
  - The Nintendo controller has A, B, up, down, left, right buttons
    - $A = \{A, B, \text{up}, \text{down}, \text{left}, \text{right}\}$
      - Cannot represent pressing multiple buttons at once
    - $A = \{0, 1\}^6$
    - $\left\{ \underbrace{\{0, 1, 2, 3, 4\}}_{\emptyset,\text{direction}}, \underbrace{\{0, 1, 2, 3\}}_{\emptyset,\text{a,b,a+b}} \right\}$

Overview

1. Lecture (approx. 1 hour)
    1. Review
    2. **Finish up MDPs**
    3. Agents/policies
    4. Derive and define Q learning
2. Discussion/questions (approx. 30 mins)

Super Mario Bros

- **Transition Function (T)?**

Super Mario Bros

- **Transition Function (T)**?
  - $T(\text{pixel\_state}, \text{right})$

Super Mario Bros

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

Super Mario Bros

- **Transition Function (T)?**

Super Mario Bros

- **Transition Function (T)?**
  - $T(\text{pos\_vel\_state}, \text{acc. right})$

Super Mario Bros

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

Markov States

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

Markov States

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

Answer: Cannot measure velocity.

Markov States

Question: Why do we need velocity in the state?

Markov States

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

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

Markov States

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

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

Not conditionally independent!

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

Super Mario Bros

- **Reward (R)?**

Super Mario Bros

- **Reward (R)**?
  - 1 for beating the level and 0 otherwise

Super Mario Bros

- **Reward (R)**?
  - 1 for beating the level and 0 otherwise
  - Total score

Super Mario Bros

- **Reward (R)?**
  - 1 for beating the level and 0 otherwise
  - Total score
  - 1 for beating the level + $0.01 \cdot$ score

Markov Decision Processes

- $S \checkmark$
- $A \checkmark$
- $T \checkmark$
- $R \checkmark$
- $\gamma ?$

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

Not all rewards are created equal

Markov Decision Processes

Not all rewards are created equal

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

Markov Decision Processes

Not all rewards are created equal

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

Usually, humans and animals prefer rewards sooner.

Markov Decision Processes

Not all rewards are created equal

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

Not all rewards are created equal

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)**

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

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

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 $\gamma = 0.9$

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

MDP Wrap Up

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

MDP Wrap Up

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

Environment: City of Cambridge

MDP Wrap Up

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

MDP Wrap Up

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

MDP Wrap Up

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

MDP Wrap Up

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

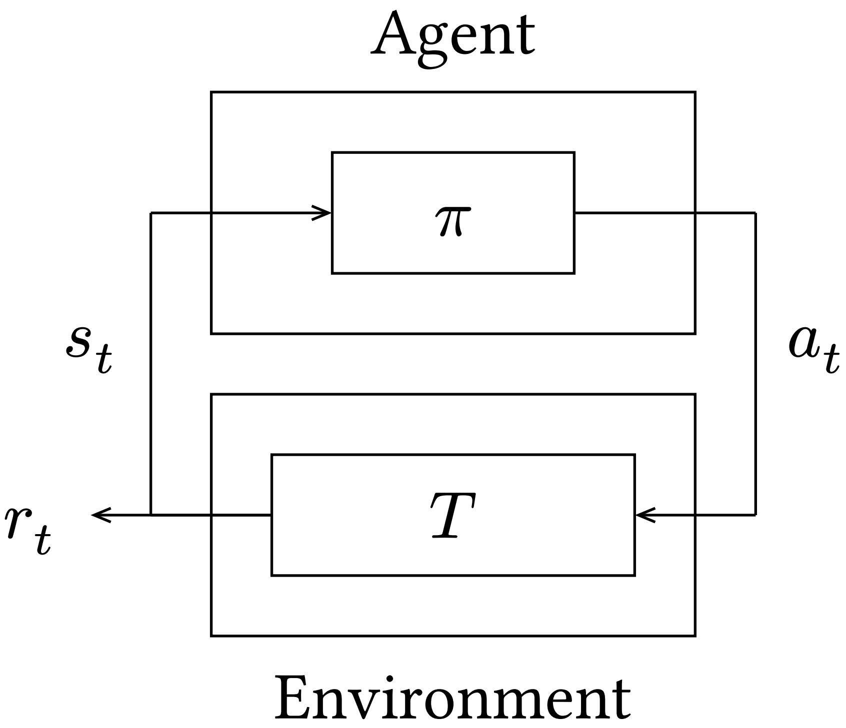
This happens internally when I decide to go to the pub after work

Agents and Policies

Deep Reinforcement Learning

University of Cambridge

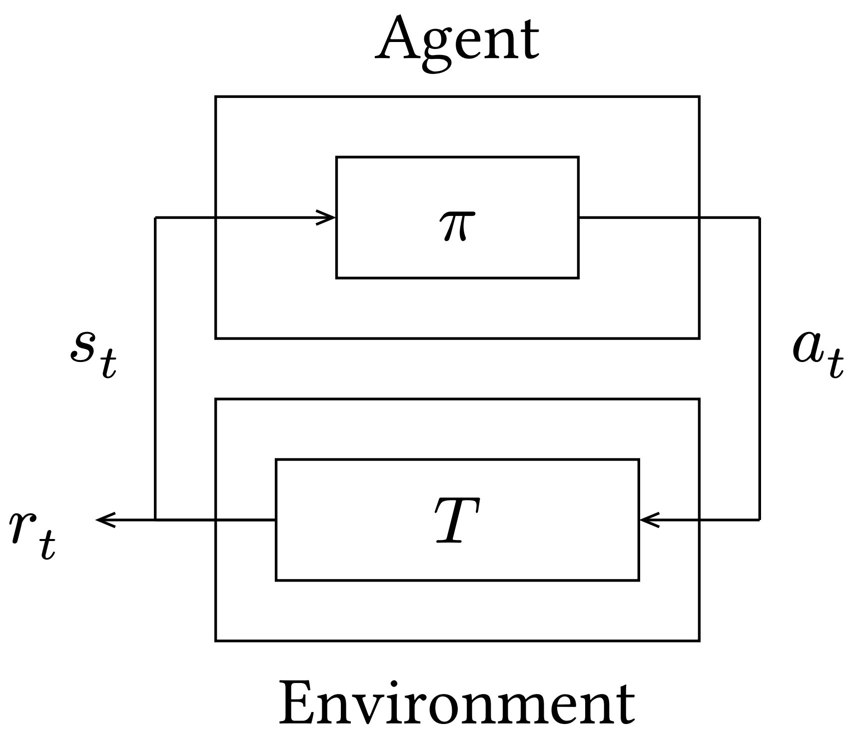
Reinforcement Learning



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

- We have defined the environment

Reinforcement Learning



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

Policies

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

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.

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.

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

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

Policies

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

Policies

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

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

Policies

The policy that maximizes the 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 transitions and policy influencing it are stochastic. The optimal policy must take this uncertainty into account.

Policies

The policy that maximizes the 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 transitions and policy influencing it are stochastic. The optimal policy must take this uncertainty into account.

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

Policies

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

The **expectation** turns that distribution into a single number. This tells us what reward to expect “on average”

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

Policies

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

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

Policies

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

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.

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

Policies

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

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 optimal with respect to all the uncertainty present!

Algorithms

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

Algorithms

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

Algorithms

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)

Algorithms

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)

Algorithms

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)

Algorithms

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)

Algorithms

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)

Algorithms

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)

Algorithms

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)

Algorithms

DQN: Q learning using a deep neural network

Algorithms

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

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

Algorithms

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

Algorithms

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

Algorithms

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

Algorithms

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

Algorithms

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

1. Lecture (approx. 1 hour)
    1. Review
    2. Finish up MDPs
    3. Agents/policies
    4. **Derive and define Q learning**
2. Discussion/questions (approx. 30 mins)

Q Learning

Q learning is state of the art in 2024 (see REDQ, DroQ, TQC)

Q Learning

Q learning is state of the art in 2024 (see REDQ, DroQ, TQC)

Today, we are doing a deep dive on Q learning

Q Learning

Q learning is state of the art in 2024 (see REDQ, DroQ, TQC)

Today, we are doing a deep dive on Q learning

A theoretical understanding of Q learning is necessary, because as discussed, many algorithms add tricks to Q learning

Q Learning

Q learning is state of the art in 2024 (see REDQ, DroQ, TQC)

Today, we are doing a deep dive on Q learning

A theoretical understanding of Q learning is necessary, because as discussed, many algorithms add tricks to Q learning

The Plan:

Q Learning

Q learning is state of the art in 2024 (see REDQ, DroQ, TQC)

Today, we are doing a deep dive on Q learning

A theoretical understanding of Q learning is necessary, because as discussed, many algorithms add tricks to Q learning

The Plan:

1. Derive the value function V

Q Learning

Q learning is state of the art in 2024 (see REDQ, DroQ, TQC)

Today, we are doing a deep dive on Q learning

A theoretical understanding of Q learning is necessary, because as discussed, many algorithms add tricks to Q learning

The Plan:

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

Q Learning

Q learning is state of the art in 2024 (see REDQ, DroQ, TQC)

Today, we are doing a deep dive on Q learning

A theoretical understanding of Q learning is necessary, because as discussed, many algorithms add tricks to Q learning

The Plan:

1. Derive the value function V
2. Derive Q function from V
3. Figure out a behavior policy using Q

Q Learning

Q learning is state of the art in 2024 (see REDQ, DroQ, TQC)

Today, we are doing a deep dive on Q learning

A theoretical understanding of Q learning is necessary, because as discussed, many algorithms add tricks to Q learning

The Plan:

1. Derive the value function V
2. Derive Q function from V
3. Figure out a behavior policy using Q
4. Learn to train Q

Step 1: The Value Function

Recall the discounted return of a specific policy π

Step 1: The Value Function

Recall the discounted return of a specific policy π

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

Step 1: The Value Function

Recall the discounted return of 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 $a_t \sim \pi(s_t)$

Step 1: The Value Function

Recall the discounted return of 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 $s_{t+1} \sim T(s_t, a_t)$

Step 1: The Value Function

Recall the discounted return of 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 $r_t = R(s_{t+1})$

Step 1: The Value Function

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

Step 1: The Value Function

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

Rather than start the return from a given timestep, what if we defined it from a given state?

Step 1: The Value Function

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

Rather than start the return from a given timestep, what if we defined it from a given state?

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

Step 1: The Value Function

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

Rather than start the return from a given timestep, what if we defined it from a given state?

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

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

Step 1: The Value Function

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

Rather than start the return from a given timestep, what if we defined it from a given state?

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

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

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

Step 2: Deriving Q

The Plan:

1. Derive the value function V
2. **Derive Q function from V**
3. Figure out a behavior policy using Q
4. Learn to train Q

Step 2: The Q Function

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

Step 2: The Q Function

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

Let's go one step further. What if the value function were conditioned on the first action?

Step 2: The Q Function

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

Let's go one step further. What if the value function were conditioned on the first action?

First, let's factor out the upcoming reward

Step 2: The Q Function

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

Let's go one step further. What if the value function were conditioned on the first action?

First, let's factor out the upcoming reward

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

Step 2: The Q Function

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

Now we can rewrite V_π as a function of the action, independent of π

Step 2: The Q Function

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

Now we can rewrite V_π as a function of the action, independent of π

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

Step 2: The Q Function

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

Step 2: The Q Function

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

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

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

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

Step 2: The Q Function

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

Step 2: The Q Function

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

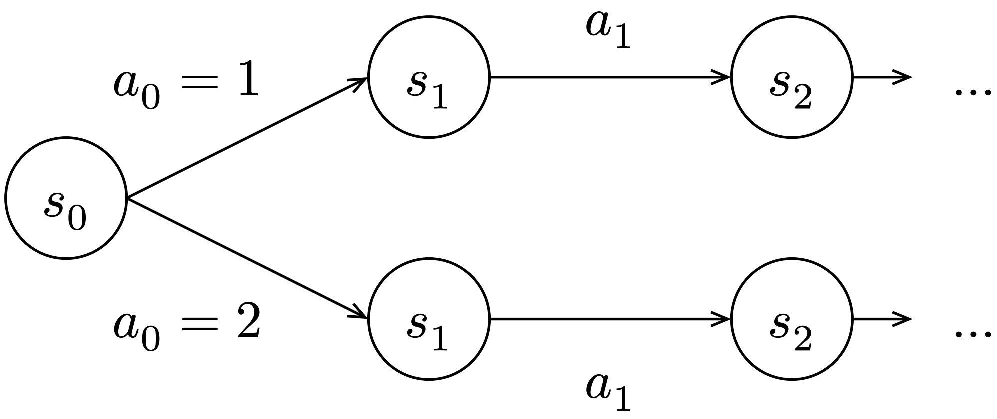
The Q function might appear simple but it is very powerful

Step 2: The Q Function

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

The Q function might appear simple but it is very powerful

a_0 affects your next state s_1, which affects the future



Step 2: The Q Function

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

Example: You have MPhil offers from Cambridge and Oxford

Step 2: The Q Function

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

Example: You have MPhil offers from Cambridge and Oxford

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

Step 2: The Q Function

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

Example: You have MPhil 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).

Step 2: The Q Function

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

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

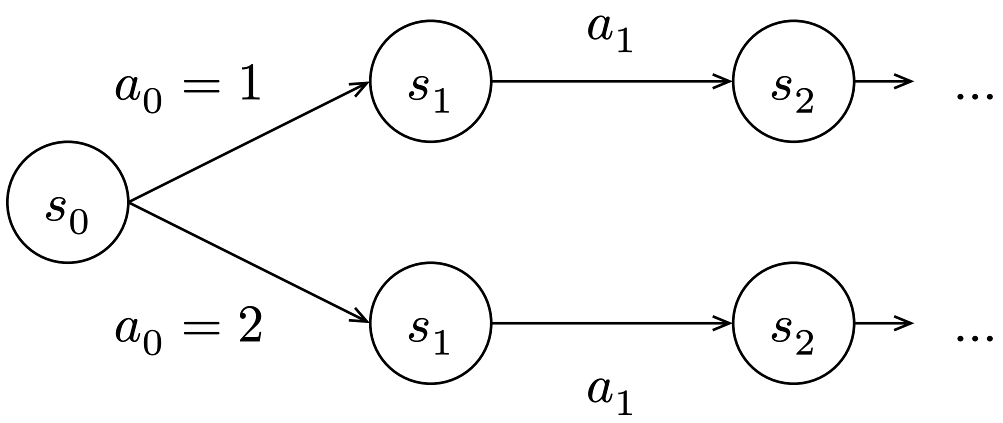
$$Q(s_0, \text{Oxford}) = f(\text{friends} + \text{experiences} + \text{income}) = 900$$

Step 2: The Q Function

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

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

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



Step 3: Find the Policy

The Plan:

1. Derive the value function V
2. Derive Q function from V
3. **Figure out a behavior policy using Q**
4. Learn to train Q

Step 3: Find the Policy

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

We know the Q function for a specific policy π, but how does this help us learn a policy?

Step 3: Find the Policy

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

We know the Q function for a specific policy π, but how does this help us learn a policy?

What would the Q function for the optimal policy π_* look like? Just add $*$ to π

Step 3: Find the Policy

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

We know the Q function for a specific policy π, but how does this help us learn a policy?

What would the Q function for the optimal policy π_* look like? Just add $*$ to π

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

Step 3: Find the Policy

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

Richard Bellman proved that a greedy policy is optimal (see the Bellman Equation)

$$\pi_*(s) = \operatorname*{argmax}_{a \in A} Q_*(s, a)$$

Step 3: Find the Policy

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

Richard Bellman proved that a greedy policy is optimal (see the Bellman Equation)

$$\pi_*(s) = \underset{a \in A}{\mathrm{argmax}}\, Q_*(s, a)$$

In English: Just take things one step at a time. Compute Q value for all possible actions and pick the action with the biggest Q value. Repeat at each timestep.

Step 3: Find the Policy

$$\pi_*(s) = \underset{a \in A}{\operatorname{argmax}} \, Q_*(s, a)$$

Step 3: Find the Policy

$$\pi_*(s) = \underset{a \in A}{\operatorname{argmax}} \, Q_*(s, a)$$

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

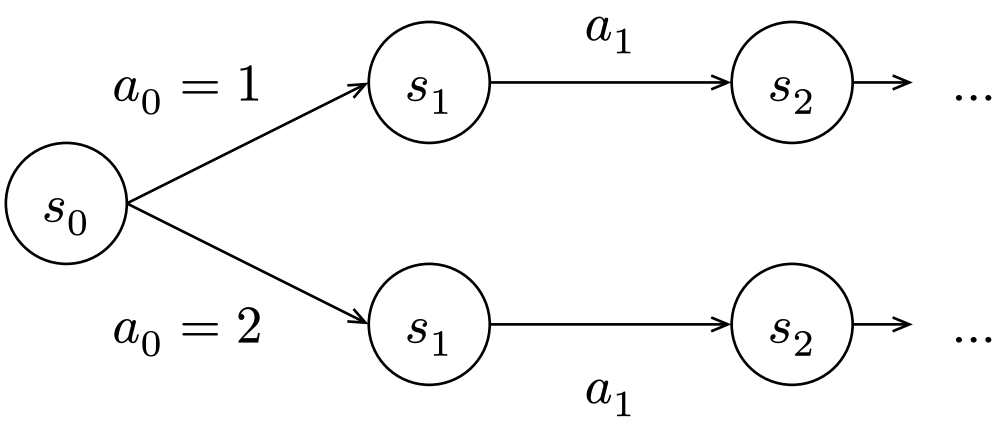
$$Q_*(s_0, \text{Oxford}) = f(\text{friends} + \text{experiences} + \text{income}) = 900$$

Step 3: Find the Policy

$$\pi_*(s) = \underset{a \in A}{\operatorname{argmax}}\, Q_*(s, a)$$

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

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



Step 3: Find the Policy

$$\pi_*(s) = \underset{a \in A}{\operatorname{argmax}} \, Q_*(s, a)$$

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

Step 3: Find the Policy

$$\pi_*(s) = \underset{a \in A}{\operatorname{argmax}} Q_*(s, a)$$

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

We can rewrite Q_* using our new π_*

$$Q_*(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid a_t = \underset{\{a \in A\}}{\operatorname{argmax}} Q_*(s_t, a)\right]$$

Step 3: Find the Policy

$$\pi_*(s) = \underset{a \in A}{\operatorname{argmax}} \, Q_*(s, a)$$

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

We can rewrite Q_* using our new π_*

$$Q_*(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid a_t = \underset{\{a \in A\}}{\operatorname{argmax}} \, Q_*(s_t, a)\right]$$

Sidenote: OpenAI's leaked AI breakthrough named Q_* is likely related to this!

Step 4: Train Q

The Plan:

1. Derive the value function V
2. Derive Q function from V
3. Figure out a behavior policy using Q
4. **Learn to train Q**

Step 4: Train Q

$$Q_*(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid a_t = \underset{\{a \in A\}}{\operatorname{argmax}} Q_*(s_t, a)\right]$$

Step 4: Train Q

$$Q_*(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\underbrace{\sum_{t=1}^{\infty} \gamma^t r_t}_{\text{Very annoying}} \mid a_t = \underset{\{a \in A\}}{\operatorname{argmax}} Q(s_t, a) \right]$$

Step 4: Train Q

$$Q_*(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\underbrace{\sum_{t=1}^{\infty} \gamma^t r_t}_{\text{Very annoying}} \mid a_t = \underset{\{a \in A\}}{\operatorname{argmax}} Q(s_t, a) \right]$$

After infinite time, we will have one datapoint for training. Can we get rid of the infinite sum?

Step 4: Train Q

$$Q_*(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid a_t = \underset{\{a \in A\}}{\operatorname{argmax}} Q_*(s_t, a)\right]$$

Factoring out the first element worked before, let's try it again

Step 4: Train Q

$$Q_*(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid a_t = \underset{\{a \in A\}}{\text{argmax}}\, Q_*(s_t, a)\right]$$

Factoring out the first element worked before, let's try it again

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

$$\gamma \cdot \mathbb{E}\left[r_1 \mid a_1 = \underset{\{a \in A\}}{\text{argmax}}\, Q_*(s_0, a)\right] + \mathbb{E}\left[\sum_{t=2}^{\infty} \gamma^t r_t \mid a_t = \underset{\{a \in A\}}{\text{argmax}}\, Q_*(s_t, a)\right]$$

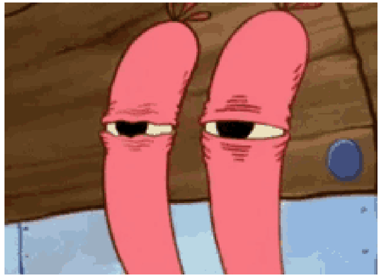
Step 4: Train Q

$$Q_*(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid a_t = \underset{\{a \in A\}}{\operatorname{argmax}} Q_*(s_t, a)\right]$$

Factoring out the first element worked before, let's try it again

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

$$\underbrace{\gamma \cdot \mathbb{E}\left[r_1 \mid a_1 = \underset{\{a \in A\}}{\operatorname{argmax}} Q(s_1, a)\right] + \mathbb{E}\left[\sum_{t=2}^{\infty} \gamma^t r_t \mid a_t = \underset{\{a \in A\}}{\operatorname{argmax}} Q_*(s_t, a)\right]}$$



Step 4: Train Q

$$Q_*(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid a_t = \underset{\{a \in A\}}{\operatorname{argmax}} Q_*(s_t, a)\right]$$

Factoring out the first element worked before, let's try it again

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

$$\gamma \cdot \mathbb{E}\left[r_1 \mid a_1 = \underset{\{a \in A\}}{\operatorname{argmax}} Q(s_1, a)\right] + \mathbb{E}\left[\sum_{t=2}^{\infty} \gamma^t r_t \mid a_t = \underset{\{a \in A\}}{\operatorname{argmax}} Q_*(s_t, a)\right]$$

It is the Q function starting at s_1, a recursive formulation!

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

Step 4: Train Q

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

Step 4: Train Q

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

Often written more simply as

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

Step 4: Train Q

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

Often written more simply as

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

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

Summary

We defined the Q function

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

Summary

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

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

Summary

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

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

We defined the Q function training objective

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

Next Time

Homework: Read *Human-level control through deep reinforcement learning*, Mnih et. al

Next Time

Homework: Read *Human-level control through deep reinforcement learning*, Mnih et. al

They go from our Q learning definition to an agent that can beat humans at most Atari games

Next Time

Homework: Read *Human-level control through deep reinforcement learning*, Mnih et. al

They go from our Q learning definition to an agent that can beat humans at most Atari games

This is what your miniproject is based on

Next Time

Homework: Read *Human-level control through deep reinforcement learning*, Mnih et. al

They go from our Q learning definition to an agent that can beat humans at most Atari games

This is what your miniproject is based on

Next Time: We will focus on a practical implementation of Deep Q Learning

Discussion Topics

1. Go around and introduce everyone
    - Name and 2-3 sentences why they chose this module

Discussion Topics

1. Go around and introduce everyone
   - Name and 2-3 sentences why they chose this module
2. What “works” in RL?
   - Why aren’t there robots delivering me lunch and folding my laundry?
3. ML/RL was supposed to free us from menial labor and give us time to pursue our passions
   - Now thousands of people provide thumbs up/thumbs down for ChatGPT completions 40 hours a week, while GPT-based models produce art, music, creative writing, etc

Discussion Topics

1. Go around and introduce everyone
   - Name and 2-3 sentences why they chose this module
2. What "works" in RL?
   - Why aren't there robots delivering me lunch and folding my laundry?
3. ML/RL was supposed to free us from menial labor and give us time to pursue our passions
   - Now thousands of people provide thumbs up/thumbs down for ChatGPT completions 40 hours a week, while GPT-based models produce art, music, creative writing, etc
4. What are you most excited to learn about next?

Discussion Topics

1. Go around and introduce everyone
   - Name and 2-3 sentences why they chose this module
2. What "works" in RL?
   - Why aren't there robots delivering me lunch and folding my laundry?
3. ML/RL was supposed to free us from menial labor and give us time to pursue our passions
   - Now thousands of people provide thumbs up/thumbs down for ChatGPT completions 40 hours a week, while GPT-based models produce art, music, creative writing, etc
4. What are you most excited to learn about next?