Actor Critic II

CISC 7404 - Decision Making

Steven Morad

University of Macau

Quiz

Quiz

- After all students put away their computer/notes/phone I will hand out exams

Quiz

- After all students put away their computer/notes/phone I will hand out exams
- If you have computer/notes/phone out after this point, it counts as cheating

Quiz

- After all students put away their computer/notes/phone I will hand out exams
- If you have computer/notes/phone out after this point, it counts as cheating
- I will hand out exams face down, do not turn them over until I say so

Quiz

- After all students put away their computer/notes/phone I will hand out exams
- If you have computer/notes/phone out after this point, it counts as cheating
- I will hand out exams face down, do not turn them over until I say so
- After turning them over, I will briefly explain each question

Quiz

- After all students put away their computer/notes/phone I will hand out exams
- If you have computer/notes/phone out after this point, it counts as cheating
- I will hand out exams face down, do not turn them over until I say so
- After turning them over, I will briefly explain each question
- After my explanation, you will have 75 minutes to complete the exam

Quiz

- After all students put away their computer/notes/phone I will hand out exams
- If you have computer/notes/phone out after this point, it counts as cheating
- I will hand out exams face down, do not turn them over until I say so
- After turning them over, I will briefly explain each question
- After my explanation, you will have 75 minutes to complete the exam
- After you are done, give me your exam and go relax outside, we resume class at 8:30

Quiz

- After all students put away their computer/notes/phone I will hand out exams
- If you have computer/notes/phone out after this point, it counts as cheating
- I will hand out exams face down, do not turn them over until I say so
- After turning them over, I will briefly explain each question
- After my explanation, you will have 75 minutes to complete the exam
- After you are done, give me your exam and go relax outside, we resume class at 8:30

Quiz

- There may or may not be different versions of the exam

Quiz

- There may or may not be different versions of the exam
- If your exam has the answer for another version, it is cheating

Quiz

- There may or may not be different versions of the exam
- If your exam has the answer for another version, it is cheating
- Instructions are in both english and chinese, english instructions take precedence

Quiz

- There may or may not be different versions of the exam
- If your exam has the answer for another version, it is cheating
- Instructions are in both english and chinese, english instructions take precedence
- Good luck!

Quiz

- 在所有学生收起电脑/笔记/手机后,我会分发试卷。
- 如果在此之后仍有电脑/笔记/手机未收,将视为作弊。
- 试卷会背面朝下发下,在我宣布开始前请勿翻面。
- 试卷翻面后,我会简要说明每道题的注意事项。
- 说明结束后,你们有 75 分钟完成考试。
- 交卷后请到教室外休息,8:30 恢复上课。
- 试卷可能存在不同版本,细节略有差异。
- 若你的试卷上出现其他版本的答案,将被判定为作弊。
- 试卷说明为中英双语,若内容冲突以英文为准。
- 祝各位考试顺利!

Admin

Admin

After today, we finish the foundational course material

Next week, we begin to investigate other parts of decision making

Right now, my plan is:

- Offline RL
- Memory and POMDPs
- Imitation Learning
- Large Language Models

Question: Should we replace a topic with something else?

Admin

- Offline RL
  - RL without exploration
  - How can we learn policies from a fixed dataset?
    - Learn surgery from surgical videos (no need to kill patients)
    - Learn driving from Xiaomi driving dataset (no need to crash cars)
  - Very new topic (2-3 years old)
  - Does not work very well (yet)

Admin

- Memory and POMDPs (my research focus)
  - So far, we always assume MDPs
  - Many interesting problems are not Markov
  - Can we extend RL to work for virtually any problem?
    - Yes, requires long-term memory
    - LSTM, transformer, etc
  - May also have time to introduce world models
    - Dreamer, TD-MPC, etc

Admin

- Imitation learning
  - Sometimes, designing a reward function is hard
  - It is easier to demonstrate desired behavior to agents
  - Agents can copy your behaviors without rewards
  - Closer to supervised learning, easier to train
    - Policies are not better than humans

Admin

- Large Language Models
  - Can train LLMs using unsupervised learning
    - They only learn to predict next word
  - We use RL to teach them to interact with humans
    - Apply policy gradient to language
    - GRPO
    - RL-adjacent methods (DPO)

Admin

Also a possibility to split lecture:

- E.g., 1 hour imitation learning, 1 hour something new

Question: Any topic sound boring?

Question: Any suggestions for other topics?

- Maybe just focus on a specific paper?

Alternative topics:

- Multi-agent RL
- Model-based RL and world-models
- Evolutionary algorithms

Admin

Homework 2 progress

If you did not already start, you might be in trouble

Experiments take a long time, start as soon as possible

Harder and requires more debugging than `FrozenLake` assignment

Admin

Those using Tencent AI Arena (Honor of Kings):

- Backup project should not rely on Honor of Kings
- **Make sure to save your code regularly**
  - Everything stored on Tencent VMs, not sure how safe code is
- There is already a PPO baseline, cannot use PPO
  - Instead, consider DDPG, SAC, TRPO, etc
- Cannot install new python libraries (Tencent security issue)
  - No `jax`, must use `torch`
  - You must learn Tencent's strange callback system
    - Prevents copy/pasting, so `torch` is ok

Review

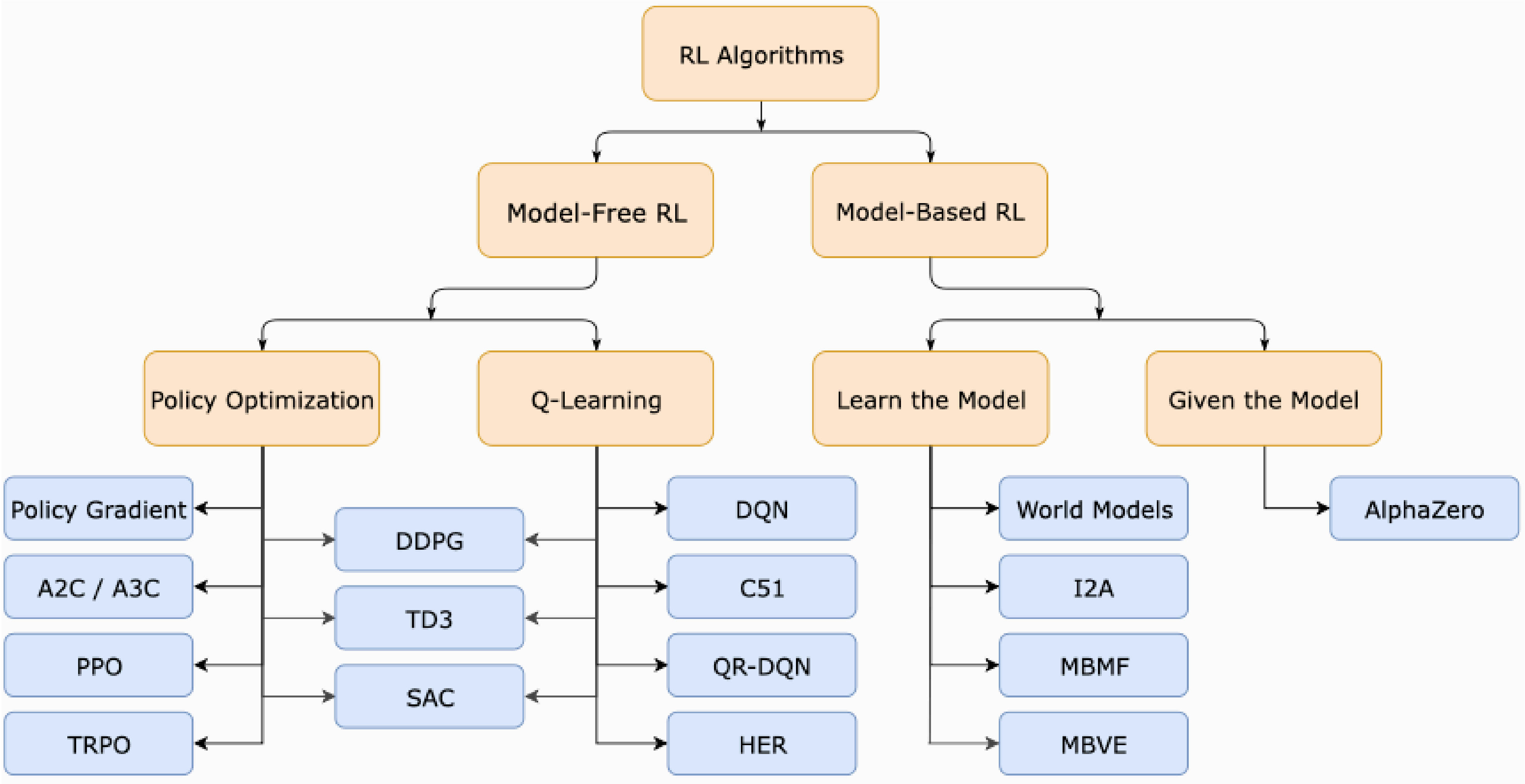
Actor Critic

Actor Critic

Alternative descriptions of actor critic algorithms

https://lilianweng.github.io/posts/2018-04-08-policy-gradient/

Actor Critic



Actor Critic

There are two approaches to actor critic

Actor Critic

There are two approaches to actor critic

1. **Policy gradient based:**

Actor Critic

There are two approaches to actor critic

1. **Policy gradient based:**

PG with V instead of MC

Actor Critic

There are two approaches to actor critic

1. **Policy gradient based:**

PG with V instead of MC

- A2C

Actor Critic

There are two approaches to actor critic

1. **Policy gradient based:**

PG with V instead of MC

- A2C
- TRPO

Actor Critic

There are two approaches to actor critic

1. **Policy gradient based:**

PG with V instead of MC

- A2C
- TRPO
- PPO

Actor Critic

There are two approaches to actor critic

1. **Policy gradient based:**

PG with V instead of MC

- A2C
- TRPO
- PPO
- IMPALA

Actor Critic

There are two approaches to actor critic

1. **Policy gradient based:**

PG with V instead of MC

- A2C
- TRPO
- PPO
- IMPALA
- ACKTR

Actor Critic

There are two approaches to actor critic

1. **Policy gradient based:**

PG with V instead of MC

- A2C
- TRPO
- PPO
- IMPALA
- ACKTR
- ACER

Actor Critic

There are two approaches to actor critic

1. **Policy gradient based:**

PG with V instead of MC

- A2C
- TRPO
- PPO
- IMPALA
- ACKTR
- ACER
- PPG

Actor Critic

There are two approaches to actor critic

1. **Policy gradient based:**

PG with V instead of MC

- A2C
- TRPO
- PPO
- IMPALA
- ACKTR
- ACER
- PPG

2. **Q learning based:**

Actor Critic

There are two approaches to actor critic

1. **Policy gradient based:**

PG with V instead of MC

- A2C
- TRPO
- PPO
- IMPALA
- ACKTR
- ACER
- PPG

2. **Q learning based:**

Learn policy to maximize Q

Actor Critic

There are two approaches to actor critic

1. **Policy gradient based:**

PG with V instead of MC

- A2C
- TRPO
- PPO
- IMPALA
- ACKTR
- ACER
- PPG

2. **Q learning based:**

Learn policy to maximize Q

- DDPG

Actor Critic

There are two approaches to actor critic

1. **Policy gradient based:**

PG with V instead of MC

- A2C
- TRPO
- PPO
- IMPALA
- ACKTR
- ACER
- PPG

2. **Q learning based:**

Learn policy to maximize Q

- DDPG
- TD3

Actor Critic

There are two approaches to actor critic

1. **Policy gradient based:**

PG with V instead of MC

- A2C
- TRPO
- PPO
- IMPALA
- ACKTR
- ACER
- PPG

2. **Q learning based:**

Learn policy to maximize Q

- DDPG
- TD3
- SAC

Actor Critic

There are two approaches to actor critic

1. **Policy gradient based:**

PG with V instead of MC

- A2C
- TRPO
- PPO
- IMPALA
- ACKTR
- ACER
- PPG

2. **Q learning based:**

Learn policy to maximize Q

- DDPG
- TD3
- SAC
- Q-Prop

Actor Critic

There are two approaches to actor critic

1. **Policy gradient based:**

PG with V instead of MC

- A2C
- TRPO
- PPO
- IMPALA
- ACKTR
- ACER
- PPG

2. **Q learning based:**

Learn policy to maximize Q

- DDPG
- TD3
- SAC
- Q-Prop
- QT-Opt

Actor Critic

There are two approaches to actor critic

1. **Policy gradient based:**

PG with V instead of MC

- A2C
- TRPO
- PPO
- IMPALA
- ACKTR
- ACER
- PPG

2. **Q learning based:**

Learn policy to maximize Q

- DDPG
- TD3
- SAC
- Q-Prop
- QT-Opt
- NAF

Actor Critic

There are two approaches to actor critic

1. **Policy gradient based:**

PG with V instead of MC

- A2C
- TRPO
- PPO
- IMPALA
- ACKTR
- ACER
- PPG

2. **Q learning based:**

Learn policy to maximize Q

- DDPG
- TD3
- SAC
- Q-Prop
- QT-Opt
- NAF
- MPO

Actor Critic

Deterministic Policy Gradient

Deterministic Policy Gradient

Question: Why did we introduce policy gradient methods?

Deterministic Policy Gradient

Question: Why did we introduce policy gradient methods?

Deterministic Policy Gradient

Deterministic Policy Gradient

Question: Why did Q learning fail BenBen?

Deterministic Policy Gradient

Question: Why did Q learning fail BenBen?

$$A = [0, 2\pi]^{12}$$

Deterministic Policy Gradient

Question: Why did Q learning fail BenBen?

$$A = [0, 2\pi]^{12}$$

$$\pi(a_t \mid s_t; \theta_\pi) = \begin{cases} 1 \text{ if } a_t = \arg\max_{a_t \in A} Q(s_t, a_t, \theta_\pi) \\ 0 \text{ otherwise} \end{cases}$$

Deterministic Policy Gradient

Question: Why did Q learning fail BenBen?

$$A = [0, 2\pi]^{12}$$

$$\pi(a_t \mid s_t; \theta_\pi) = \begin{cases} 1 \text{ if } a_t = \arg\max_{a_t \in A} Q(s_t, a_t, \theta_\pi) \\ 0 \text{ otherwise} \end{cases}$$

Infinitely many a_t – compute Q for each and take arg max over all

Deterministic Policy Gradient

$$\pi(a_t \mid s_t; \theta_\pi) = \begin{cases} 1 \text{ if } a_t = \arg\max_{a_t \in A} Q(s_t, a_t, \theta_\pi) \\ 0 \text{ otherwise} \end{cases}$$

Deterministic Policy Gradient

$$\pi(a_t \mid s_t; \theta_\pi) = \begin{cases} 1 \text{ if } a_t = \arg\max_{a_t \in A} Q(s_t, a_t, \theta_\pi) \\ 0 \text{ otherwise} \end{cases}$$

The greedy policy cannot work with continuous action spaces

Deterministic Policy Gradient

$$\pi(a_t \mid s_t; \theta_\pi) = \begin{cases} 1 \text{ if } a_t = \arg\max_{a_t \in A} Q(s_t, a_t, \theta_\pi) \\ 0 \text{ otherwise} \end{cases}$$

The greedy policy cannot work with continuous action spaces

But the Q function and greedy policy are different

Deterministic Policy Gradient

$$\pi(a_t \mid s_t; \theta_\pi) = \begin{cases} 1 \text{ if } a_t = \arg\max_{a_t \in A} Q(s_t, a_t, \theta_\pi) \\ 0 \text{ otherwise} \end{cases}$$

The greedy policy cannot work with continuous action spaces

But the Q function and greedy policy are different

Let us quickly review the Q function and value function

Deterministic Policy Gradient

The most general form of Q

Deterministic Policy Gradient

The most general form of Q

$$Q(s_0, a_0, \theta_\pi) = \mathbb{R}[\mathcal{R}(s_{t+1}) \mid s_0, a_0] + \gamma V(s_1, \theta_\pi)$$

Deterministic Policy Gradient

The most general form of Q

$$Q(s_0, a_0, \theta_\pi) = \mathbb{R}[\mathcal{R}(s_{t+1}) \mid s_0, a_0] + \gamma V(s_1, \theta_\pi)$$

The reward for taking a_0 and following θ_π afterward

Deterministic Policy Gradient

The most general form of Q

$$Q(s_0, a_0, \theta_\pi) = \mathbb{R}[\mathcal{R}(s_{t+1}) \mid s_0, a_0] + \gamma V(s_1, \theta_\pi)$$

The reward for taking a_0 and following θ_π afterward

We can replace V with Q if $a \sim \pi(\cdot \mid s_1; \theta_\pi)$

Deterministic Policy Gradient

The most general form of Q

$$Q(s_0, a_0, \theta_\pi) = \mathbb{R}[\mathcal{R}(s_{t+1}) \mid s_0, a_0] + \gamma V(s_1, \theta_\pi)$$

The reward for taking a_0 and following θ_π afterward

We can replace V with Q if $a \sim \pi(\cdot \mid s_1; \theta_\pi)$

$$Q(s_0, a_0, \theta_\pi) = \mathbb{R}[\mathcal{R}(s_{t+1}) \mid s_0, a_0] + \gamma Q(s_1, a, \theta_\pi)$$

Deterministic Policy Gradient

The most general form of Q

$$Q(s_0, a_0, \theta_\pi) = \mathbb{R}[\mathcal{R}(s_{t+1}) \mid s_0, a_0] + \gamma V(s_1, \theta_\pi)$$

The reward for taking a_0 and following θ_π afterward

We can replace V with Q if $a \sim \pi(\cdot \mid s_1; \theta_\pi)$

$$Q(s_0, a_0, \theta_\pi) = \mathbb{R}[\mathcal{R}(s_{t+1}) \mid s_0, a_0] + \gamma Q(s_1, a, \theta_\pi)$$

For the greedy policy, we can reduce Q further

$$Q(s_0, a_0, \theta_\pi) = \mathbb{R}[\mathcal{R}(s_{t+1}) \mid s_0, a_0] + \max_{a \in A} \gamma Q(s_1, a, \theta_\pi)$$

Deterministic Policy Gradient

The most general form of Q

$$Q(s_0, a_0, \theta_\pi) = \mathbb{R}[\mathcal{R}(s_{t+1}) \mid s_0, a_0] + \gamma V(s_1, \theta_\pi)$$

The reward for taking a_0 and following θ_π afterward

We can replace V with Q if $a \sim \pi(\cdot \mid s_1; \theta_\pi)$

$$Q(s_0, a_0, \theta_\pi) = \mathbb{R}[\mathcal{R}(s_{t+1}) \mid s_0, a_0] + \gamma Q(s_1, a, \theta_\pi)$$

For the greedy policy, we can reduce Q further

$$Q(s_0, a_0, \theta_\pi) = \mathbb{R}[\mathcal{R}(s_{t+1}) \mid s_0, a_0] + \max_{a \in A} \gamma Q(s_1, a, \theta_\pi)$$

Deterministic Policy Gradient

$$\cancel{Q(s_0, a_0, \theta_\pi) = \mathbb{R}[\mathcal{R}(s_{t+1}) \mid s_0, a_0] + \max_{a \in A} \gamma Q(s_1, a, \theta_\pi)}$$

Deterministic Policy Gradient

$$\cancel{Q(s_0, a_0, \theta_\pi) = \mathbb{R}[\mathcal{R}(s_{t+1}) \mid s_0, a_0] + \max_{a \in A} \gamma Q(s_1, a, \theta_\pi)}$$

Cannot use the greedy Q function with BenBen

Deterministic Policy Gradient

$$\cancel{Q(s_0, a_0, \theta_\pi) = \mathbb{R}[\mathcal{R}(s_{t+1}) \mid s_0, a_0] + \max_{a \in A} \gamma Q(s_1, a, \theta_\pi)}$$

Cannot use the greedy Q function with BenBen

But we can use Q function with **any** policy θ_π

Deterministic Policy Gradient

$$\cancel{Q(s_0, a_0, \theta_\pi) = \mathbb{R}[\mathcal{R}(s_{t+1}) \mid s_0, a_0] + \max_{a \in A} \gamma Q(s_1, a, \theta_\pi)}$$

Cannot use the greedy Q function with BenBen

But we can use Q function with **any** policy θ_π

- ~~Greedy policy~~

Deterministic Policy Gradient

$$\cancel{Q(s_0, a_0, \theta_\pi) = \mathbb{R}[\mathcal{R}(s_{t+1}) \mid s_0, a_0] + \max_{a \in A} \gamma Q(s_1, a, \theta_\pi)}$$

Cannot use the greedy Q function with BenBen

But we can use Q function with **any** policy θ_π

- ~~Greedy policy~~
- Human policy

Deterministic Policy Gradient

$$\cancel{Q(s_0, a_0, \theta_\pi) = \mathbb{R}[\mathcal{R}(s_{t+1}) \mid s_0, a_0] + \max_{a \in A} \gamma Q(s_1, a, \theta_\pi)}$$

Cannot use the greedy Q function with BenBen

But we can use Q function with **any** policy θ_π

- ~~Greedy policy~~
- Human policy
- Random policy

Deterministic Policy Gradient

$$\cancel{Q(s_0, a_0, \theta_\pi) = \mathbb{R}[\mathcal{R}(s_{t+1}) \mid s_0, a_0] + \max_{a \in A} \gamma Q(s_1, a, \theta_\pi)}$$

Cannot use the greedy Q function with BenBen

But we can use Q function with **any** policy θ_π

- ~~Greedy policy~~
- Human policy
- Random policy

$$Q(s_0, a_0, \theta_\pi) = \mathbb{R}[\mathcal{R}(s_{t+1}) \mid s_0, a_0] + \gamma V(s_1, \theta_\pi)$$

$$Q(s_0, a_0, \theta_\pi) = \mathbb{R}[\mathcal{R}(s_{t+1}) \mid s_0, a_0] + \gamma Q(s_1, a, \theta_\pi); \quad a \sim \pi(\cdot \mid s_1; \theta_\pi)$$

Deterministic Policy Gradient

$$\cancel{Q(s_0, a_0, \theta_\pi) = \mathbb{R}[\mathcal{R}(s_{t+1}) \mid s_0, a_0] + \max_{a \in A} \gamma Q(s_1, a, \theta_\pi)}$$

Cannot use the greedy Q function with BenBen

But we can use Q function with **any** policy θ_π

- ~~Greedy policy~~
- Human policy
- Random policy

$$Q(s_0, a_0, \theta_\pi) = \mathbb{R}[\mathcal{R}(s_{t+1}) \mid s_0, a_0] + \gamma V(s_1, \theta_\pi)$$

$$Q(s_0, a_0, \theta_\pi) = \mathbb{R}[\mathcal{R}(s_{t+1}) \mid s_0, a_0] + \gamma Q(s_1, a, \theta_\pi); \quad a \sim \pi(\cdot \mid s_1; \theta_\pi)$$

Question: Can we learn a continuous policy using Q?

Deterministic Policy Gradient

What if we use Q to learn a policy?

Deterministic Policy Gradient

What if we use Q to learn a policy?

Question: What method can we use for policy learning?

Deterministic Policy Gradient

What if we use Q to learn a policy?

Question: What method can we use for policy learning?

Answer: Policy gradient

Deterministic Policy Gradient

What if we use Q to learn a policy?

Question: What method can we use for policy learning?

Answer: Policy gradient

Perhaps our solution will combine Q with policy gradient

Deterministic Policy Gradient

What if we use Q to learn a policy?

Question: What method can we use for policy learning?

Answer: Policy gradient

Perhaps our solution will combine Q with policy gradient

We will derive the solution like David Silver likely did

Deterministic Policy Gradient

What if we use Q to learn a policy?

Question: What method can we use for policy learning?

Answer: Policy gradient

Perhaps our solution will combine Q with policy gradient

We will derive the solution like David Silver likely did

The trick is to consider a **deterministic** policy

Deterministic Policy Gradient

What if we use Q to learn a policy?

Question: What method can we use for policy learning?

Answer: Policy gradient

Perhaps our solution will combine Q with policy gradient

We will derive the solution like David Silver likely did

The trick is to consider a **deterministic** policy

$$\mu : S \times \Theta \mapsto A$$

Deterministic Policy Gradient

What if we use Q to learn a policy?

Question: What method can we use for policy learning?

Answer: Policy gradient

Perhaps our solution will combine Q with policy gradient

We will derive the solution like David Silver likely did

The trick is to consider a **deterministic** policy

$$\mu : S \times \Theta \mapsto A \qquad\qquad a = \mu(s, \theta_\mu)$$

Deterministic Policy Gradient

What if we use Q to learn a policy?

Question: What method can we use for policy learning?

Answer: Policy gradient

Perhaps our solution will combine Q with policy gradient

We will derive the solution like David Silver likely did

The trick is to consider a **deterministic** policy

$$\mu : S \times \Theta \mapsto A \qquad a = \mu(s, \theta_\mu)$$

Deterministic Policy Gradient

Recall policy gradient

Deterministic Policy Gradient

Recall policy gradient

We find θ_π using gradient ascent

Deterministic Policy Gradient

Recall policy gradient

We find θ_π using gradient ascent

$$\theta_{i+1} = \theta_i + \alpha \cdot \nabla_{\theta_\pi} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\pi]$$

Deterministic Policy Gradient

Recall policy gradient

We find θ_π using gradient ascent

$$\theta_{i+1} = \theta_i + \alpha \cdot \nabla_{\theta_\pi} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\pi]$$

We need to know $\nabla_{\theta_\pi} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\pi]$

Deterministic Policy Gradient

Recall policy gradient

We find θ_π using gradient ascent

$$\theta_{i+1} = \theta_i + \alpha \cdot \nabla_{\theta_\pi} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\pi]$$

We need to know $\nabla_{\theta_\pi} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\pi]$

We found this for stochastic policy gradient

Deterministic Policy Gradient

Recall policy gradient

We find θ_π using gradient ascent

$$\theta_{i+1} = \theta_i + \alpha \cdot \nabla_{\theta_\pi} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\pi]$$

We need to know $\nabla_{\theta_\pi} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\pi]$

We found this for stochastic policy gradient

How does a deterministic policy change $\nabla_{\theta_\mu} \mathbb{E}\left[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\mu\right]$?

Deterministic Policy Gradient

The expected return with a **stochastic** policy

$$\mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\pi] = \sum_{n=0}^{\infty} \gamma^n \sum_{s_{n+1} \in S} \mathcal{R}(s_{n+1}) \cdot \Pr(s_{n+1} \mid s_0; \theta_\pi)$$

$$\Pr(s_{n+1} \mid s_0; \theta_\pi) = \sum_{s_1, \dots, s_n \in S} \prod_{t=0}^{n} \left(\sum_{a_t \in A} \mathrm{Tr}(s_{t+1} \mid s_t, a_t) \cdot \pi(a_t \mid s_t; \theta_\pi) \right)$$

The state distribution with a **deterministic** policy

$$\Pr(s_{n+1} \mid s_0; \theta_\mu) = \sum_{s_1, \dots, s_n \in S} \prod_{t=0}^{n} \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu))$$

Deterministic Policy Gradient

Let us continue to derive policy gradient with a **deterministic** policy μ

Deterministic Policy Gradient

Let us continue to derive policy gradient with a **deterministic** policy μ

$$\mathbb{E}\left[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\mu\right] = \sum_{n=0}^{\infty} \gamma^n \sum_{s_{n+1} \in S} \mathcal{R}(s_{n+1}) \cdot \Pr\left(s_{n+1} \mid s_0; \theta_\mu\right)$$

$$\Pr\left(s_{n+1} \mid s_0; \theta_\mu\right) = \sum_{s_1, \ldots, s_n \in S} \prod_{t=0}^{n} \mathrm{Tr}\left(s_{t+1} \mid s_t, \mu\left(s_t, \theta_\mu\right)\right)$$

Deterministic Policy Gradient

Let us continue to derive policy gradient with a **deterministic** policy μ

$$\mathbb{E}\left[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\mu\right] = \sum_{n=0}^{\infty} \gamma^n \sum_{s_{n+1} \in S} \mathcal{R}(s_{n+1}) \cdot \Pr(s_{n+1} \mid s_0; \theta_\mu)$$

$$\Pr(s_{n+1} \mid s_0; \theta_\mu) = \sum_{s_1, \dots, s_n \in S} \prod_{t=0}^{n} \operatorname{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu))$$

Plug state distribution into expected return

Deterministic Policy Gradient

Let us continue to derive policy gradient with a **deterministic** policy μ

$$\mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\mu] = \sum_{n=0}^{\infty} \gamma^n \sum_{s_{n+1} \in S} \mathcal{R}(s_{n+1}) \cdot \Pr(s_{n+1} \mid s_0; \theta_\mu)$$

$$\Pr(s_{n+1} \mid s_0; \theta_\mu) = \sum_{s_1, \dots, s_n \in S} \prod_{t=0}^{n} \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu))$$

Plug state distribution into expected return

$$\mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\mu] = \sum_{n=0}^{\infty} \gamma^n \sum_{s_{n+1} \in S} \mathcal{R}(s_{n+1}) \sum_{s_1, \dots, s_n \in S} \prod_{t=0}^{n} \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu))$$

Deterministic Policy Gradient

$$\nabla_{\theta_\mu} \left[\mathbb{E}[\mathcal{G}(\boldsymbol{\tau})] \mid s_0; \theta_\mu \right]$$

$$= \nabla_{\theta_\mu} \left[\sum_{n=0}^{\infty} \gamma^n \sum_{s_{n+1} \in S} \mathcal{R}(s_{n+1}) \sum_{s_1, \dots, s_n \in S} \prod_{t=0}^{n} \mathrm{Tr}\left(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu)\right) \right]$$

Deterministic Policy Gradient

$$\nabla_{\theta_\mu} \left[\mathbb{E}[\mathcal{G}(\boldsymbol{\tau})] \mid s_0; \theta_\mu \right]$$

$$= \nabla_{\theta_\mu} \left[\sum_{n=0}^{\infty} \gamma^n \sum_{s_{n+1} \in S} \mathcal{R}(s_{n+1}) \sum_{s_1, \ldots, s_n \in S} \prod_{t=0}^{n} \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu)) \right]$$

Gradient of sum is sum of gradient, move the gradient inside the sums

Deterministic Policy Gradient

$$\nabla_{\theta_\mu} \left[\mathbb{E}[\mathcal{G}(\boldsymbol{\tau})] \mid s_0; \theta_\mu \right]$$

$$= \nabla_{\theta_\mu} \left[\sum_{n=0}^{\infty} \gamma^n \sum_{s_{n+1} \in S} \mathcal{R}(s_{n+1}) \sum_{s_1, \dots, s_n \in S} \prod_{t=0}^{n} \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu)) \right]$$

Gradient of sum is sum of gradient, move the gradient inside the sums

$$= \sum_{n=0}^{\infty} \gamma^n \sum_{s_{n+1} \in S} \mathcal{R}(s_{n+1}) \sum_{s_1, \dots, s_n \in S} \nabla_{\theta_\mu} \left[\prod_{t=0}^{n} \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu)) \right]$$

Deterministic Policy Gradient

$$= \sum_{n=0}^{\infty} \gamma^n \sum_{s_{n+1} \in S} \mathcal{R}(s_{n+1}) \sum_{s_1, \ldots, s_n \in S} \nabla_{\theta_\mu} \left[\prod_{t=0}^{n} \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu)) \right]$$

Deterministic Policy Gradient

$$= \sum_{n=0}^{\infty} \gamma^n \sum_{s_{n+1} \in S} \mathcal{R}(s_{n+1}) \sum_{s_1, \dots, s_n \in S} \nabla_{\theta_\mu} \left[\prod_{t=0}^{n} \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu)) \right]$$

Recall the log trick

Deterministic Policy Gradient

$$= \sum_{n=0}^{\infty} \gamma^n \sum_{s_{n+1} \in S} \mathcal{R}(s_{n+1}) \sum_{s_1, \ldots, s_n \in S} \nabla_{\theta_\mu} \left[\prod_{t=0}^{n} \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu)) \right]$$

Recall the log trick $\nabla_x f(x) = f(x) \nabla_x \log f(x)$

Deterministic Policy Gradient

$$= \sum_{n=0}^{\infty} \gamma^n \sum_{s_{n+1} \in S} \mathcal{R}(s_{n+1}) \sum_{s_1, \ldots, s_n \in S} \nabla_{\theta_\mu} \left[\prod_{t=0}^{n} \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu)) \right]$$

Recall the log trick $\nabla_x f(x) = f(x) \nabla_x \log f(x)$

Apply log trick to product terms

Deterministic Policy Gradient

$$= \sum_{n=0}^{\infty} \gamma^n \sum_{s_{n+1} \in S} \mathcal{R}(s_{n+1}) \sum_{s_1, \ldots, s_n \in S} \nabla_{\theta_\mu} \left[\prod_{t=0}^{n} \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu)) \right]$$

Recall the log trick $\nabla_x f(x) = f(x) \nabla_x \log f(x)$

Apply log trick to product terms

$$= \sum_{n=0}^{\infty} \gamma^n \sum_{s_{n+1} \in S} \mathcal{R}(s_{n+1}) \sum_{s_1, \ldots, s_n \in S}$$

$$\prod_{t=0}^{n} \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu)) \nabla_{\theta_\mu} \left[\log \left(\prod_{t=0}^{n} \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu)) \right) \right]$$

Deterministic Policy Gradient

$$= \sum_{n=0}^{\infty} \gamma^n \sum_{s_{n+1} \in S} \mathcal{R}(s_{n+1}) \sum_{s_1, \ldots, s_n \in S} \prod_{t=0}^{n} \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu))$$

$$\nabla_{\theta_\mu} \left[\log \left(\prod_{t=0}^{n} \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu)) \right) \right]$$

Deterministic Policy Gradient

$$= \sum_{n=0}^{\infty} \gamma^n \sum_{s_{n+1} \in S} \mathcal{R}(s_{n+1}) \sum_{s_1, \ldots, s_n \in S} \prod_{t=0}^{n} \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu))$$

$$\nabla_{\theta_\mu} \left[\log \left(\prod_{t=0}^{n} \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu)) \right) \right]$$

Log of products is sum of logs

Deterministic Policy Gradient

$$= \sum_{n=0}^{\infty} \gamma^n \sum_{s_{n+1} \in S} \mathcal{R}(s_{n+1}) \sum_{s_1, \ldots, s_n \in S} \prod_{t=0}^{n} \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu))$$

$$\nabla_{\theta_\mu} \left[\log \left(\prod_{t=0}^{n} \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu)) \right) \right]$$

Log of products is sum of logs

$$= \sum_{n=0}^{\infty} \gamma^n \sum_{s_{n+1} \in S} \mathcal{R}(s_{n+1}) \sum_{s_1, \ldots, s_n \in S} \prod_{t=0}^{n} \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu))$$

$$\nabla_{\theta_\mu} \left[\sum_{t=0}^{n} \log \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu)) \right]$$

Deterministic Policy Gradient

$$= \sum_{n=0}^{\infty} \gamma^n \sum_{s_{n+1} \in S} \mathcal{R}(s_{n+1}) \sum_{s_1, \ldots, s_n \in S} \prod_{t=0}^{n} \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu))$$

$$\nabla_{\theta_\mu} \left[\sum_{t=0}^{n} \log \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu)) \right]$$

Deterministic Policy Gradient

$$= \sum_{n=0}^{\infty} \gamma^n \sum_{s_{n+1} \in S} \mathcal{R}(s_{n+1}) \sum_{s_1, \ldots, s_n \in S} \prod_{t=0}^{n} \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu))$$

$$\nabla_{\theta_\mu} \left[\sum_{t=0}^{n} \log \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu)) \right]$$

Move the gradient inside sum

Deterministic Policy Gradient

$$= \sum_{n=0}^{\infty} \gamma^n \sum_{s_{n+1} \in S} \mathcal{R}(s_{n+1}) \sum_{s_1, \ldots, s_n \in S} \prod_{t=0}^{n} \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu))$$

$$\nabla_{\theta_\mu} \left[\sum_{t=0}^{n} \log \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu)) \right]$$

Move the gradient inside sum

$$= \sum_{n=0}^{\infty} \gamma^n \sum_{s_{n+1} \in S} \mathcal{R}(s_{n+1}) \sum_{s_1, \ldots, s_n \in S} \prod_{t=0}^{n} \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu))$$

$$\sum_{t=0}^{n} \nabla_{\theta_\mu} \log \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu))$$

Deterministic Policy Gradient

$$= \sum_{n=0}^{\infty} \gamma^n \sum_{s_{n+1} \in S} \mathcal{R}(s_{n+1}) \sum_{s_1, \ldots, s_n \in S} \prod_{t=0}^{n} \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu))$$

$$\sum_{t=0}^{n} \nabla_{\theta_\mu} \log \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu))$$

Deterministic Policy Gradient

$$= \sum_{n=0}^{\infty} \gamma^n \sum_{s_{n+1} \in S} \mathcal{R}(s_{n+1}) \sum_{s_1, \ldots, s_n \in S} \prod_{t=0}^{n} \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu))$$

$$\sum_{t=0}^{n} \nabla_{\theta_\mu} \log \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu))$$

Use the chain rule $\qquad \nabla_x f(g(x)) = \nabla_g [f(g(x))] \nabla_x g(x)$

Deterministic Policy Gradient

$$= \sum_{n=0}^{\infty} \gamma^n \sum_{s_{n+1} \in S} \mathcal{R}(s_{n+1}) \sum_{s_1, \dots, s_n \in S} \prod_{t=0}^{n} \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu))$$

$$\sum_{t=0}^{n} \nabla_{\theta_\mu} \log \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu))$$

Use the chain rule $\nabla_x f(g(x)) = \nabla_g [f(g(x))] \nabla_x g(x)$

$$= \sum_{n=0}^{\infty} \gamma^n \sum_{s_{n+1} \in S} \mathcal{R}(s_{n+1}) \cdot \sum_{s_1, \dots, s_n \in S} \prod_{t=0}^{n} \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu))$$

$$\sum_{t=0}^{n} \nabla_\mu [\log \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu))] \cdot \nabla_{\theta_\mu} \mu(s_t, \theta_\mu)$$

Deterministic Policy Gradient

$$= \sum_{n=0}^{\infty} \gamma^n \sum_{s_{n+1} \in S} \mathcal{R}(s_{n+1}) \cdot \sum_{s_1, \ldots, s_n \in S} \prod_{t=0}^{n} \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu))$$

$$\sum_{t=0}^{n} \nabla_\mu \left[\log \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu))\right] \cdot \nabla_{\theta_\mu} \mu(s_t, \theta_\mu)$$

Deterministic Policy Gradient

$$= \sum_{n=0}^{\infty} \gamma^n \sum_{s_{n+1} \in S} \mathcal{R}(s_{n+1}) \cdot \sum_{s_1, \dots, s_n \in S} \prod_{t=0}^{n} \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu))$$

$$\sum_{t=0}^{n} \nabla_\mu [\log \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu))] \cdot \nabla_{\theta_\mu} \mu(s_t, \theta_\mu)$$

We know must know ∇ Tr to find the deterministic policy gradient

Deterministic Policy Gradient

$$= \sum_{n=0}^{\infty} \gamma^n \sum_{s_{n+1} \in S} \mathcal{R}(s_{n+1}) \cdot \sum_{s_1, \dots, s_n \in S} \prod_{t=0}^{n} \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu))$$

$$\sum_{t=0}^{n} \nabla_\mu \left[\log \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu))\right] \cdot \nabla_{\theta_\mu} \mu(s_t, \theta_\mu)$$

We know must know ∇ Tr to find the deterministic policy gradient

In many cases, we do not know ∇ Tr

Deterministic Policy Gradient

$$= \sum_{n=0}^{\infty} \gamma^n \sum_{s_{n+1} \in S} \mathcal{R}(s_{n+1}) \cdot \sum_{s_1, \ldots, s_n \in S} \prod_{t=0}^{n} \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu))$$

$$\sum_{t=0}^{n} \nabla_\mu \left[\log \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu))\right] \cdot \nabla_{\theta_\mu} \mu(s_t, \theta_\mu)$$

We know must know ∇ Tr to find the deterministic policy gradient

In many cases, we do not know ∇ Tr

Stochastic policy gradient is very special – we do not need ∇ Tr

Deterministic Policy Gradient

$$= \sum_{n=0}^{\infty} \gamma^n \sum_{s_{n+1} \in S} \mathcal{R}(s_{n+1}) \cdot \sum_{s_1, \ldots, s_n \in S} \prod_{t=0}^{n} \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu))$$

$$\sum_{t=0}^{n} \nabla_\mu [\log \mathrm{Tr}(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu))] \cdot \nabla_{\theta_\mu} \mu(s_t, \theta_\mu)$$

We know must know ∇ Tr to find the deterministic policy gradient

In many cases, we do not know ∇ Tr

Stochastic policy gradient is very special – we do not need ∇ Tr

Let me explain what I mean

Deterministic Policy Gradient

With deterministic policy, μ inside Tr means chain rule

Deterministic Policy Gradient

With deterministic policy, μ inside Tr means chain rule

$$\sum_{t=0}^{n} \nabla_{\theta_\mu} \log \mathrm{Tr}\left(s_{t+1} \mid s_t, \mu\left(s_t, \theta_\mu\right)\right)$$

Deterministic Policy Gradient

With deterministic policy, μ inside Tr means chain rule

$$\sum_{t=0}^{n} \nabla_{\theta_\mu} \log \mathrm{Tr}\big(s_{t+1} \mid s_t, \mu(s_t, \theta_\mu)\big)$$

With stochastic policy, we multiply Tr by π (product rule)

Deterministic Policy Gradient

With deterministic policy, μ inside Tr means chain rule

$$\sum_{t=0}^{n} \nabla_{\theta_\mu} \log \mathrm{Tr}\left(s_{t+1} \mid s_t, \mu\left(s_t, \theta_\mu\right)\right)$$

With stochastic policy, we multiply Tr by π (product rule)

$$\sum_{a_0, \ldots, a_n \in A} \sum_{t=0}^{n} \nabla_{\theta_\pi} \left[\mathrm{Tr}\left(s_{t+1} \mid s_t, a_t\right) \cdot \log \pi\left(a_t \mid s_t; \theta_\pi\right)\right]$$

Deterministic Policy Gradient

With deterministic policy, μ inside Tr means chain rule

$$\sum_{t=0}^{n} \nabla_{\theta_\mu} \log \mathrm{Tr}\left(s_{t+1} \mid s_t, \mu\left(s_t, \theta_\mu\right)\right)$$

With stochastic policy, we multiply Tr by π (product rule)

$$\sum_{a_0, \ldots, a_n \in A} \sum_{t=0}^{n} \nabla_{\theta_\pi} \left[\mathrm{Tr}\left(s_{t+1} \mid s_t, a_t\right) \cdot \log \pi\left(a_t \mid s_t; \theta_\pi\right)\right]$$

Tr comes out of the sum and disappears into the expected return

Deterministic Policy Gradient

With deterministic policy, μ inside Tr means chain rule

$$\sum_{t=0}^{n} \nabla_{\theta_\mu} \log \mathrm{Tr}\left(s_{t+1} \mid s_t, \mu\left(s_t, \theta_\mu\right)\right)$$

With stochastic policy, we multiply Tr by π (product rule)

$$\sum_{a_0, \ldots, a_n \in A} \sum_{t=0}^{n} \nabla_{\theta_\pi} \left[\mathrm{Tr}\left(s_{t+1} \mid s_t, a_t\right) \cdot \log \pi\left(a_t \mid s_t; \theta_\pi\right)\right]$$

Tr comes out of the sum and disappears into the expected return

This is one reason why we always consider stochastic π

Deep Deterministic Policy Gradient

Deep Deterministic Policy Gradient

$$\nabla_{\theta_\mu} \mathbb{E}\left[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\mu\right]$$

Deep Deterministic Policy Gradient

$$\nabla_{\theta_\mu} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\mu] \qquad \theta_{\pi, i+1} = \theta_{\pi, i} + \alpha \cdot \nabla_{\theta_\mu} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\mu]$$

Deep Deterministic Policy Gradient

$$\nabla_{\theta_\mu} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\mu] \qquad \theta_{\pi,i+1} = \theta_{\pi,i} + \alpha \cdot \nabla_{\theta_\mu} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\mu]$$

We failed – a deterministic policy gradient does not seem to work

Deep Deterministic Policy Gradient

$$\nabla_{\theta_\mu} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\mu] \qquad \theta_{\pi,i+1} = \theta_{\pi,i} + \alpha \cdot \nabla_{\theta_\mu} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\mu]$$

We failed – a deterministic policy gradient does not seem to work

Can we try and optimize something else?

Deep Deterministic Policy Gradient

$$\nabla_{\theta_\mu} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\mu] \qquad \theta_{\pi, i+1} = \theta_{\pi, i} + \alpha \cdot \nabla_{\theta_\mu} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\mu]$$

We failed – a deterministic policy gradient does not seem to work

Can we try and optimize something else?

Question: Could we replace $\nabla_{\theta_\mu} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\mu]$ with something else?

Deep Deterministic Policy Gradient

$$\nabla_{\theta_\mu} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\mu] \qquad \theta_{\pi, i+1} = \theta_{\pi, i} + \alpha \cdot \nabla_{\theta_\mu} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\mu]$$

We failed – a deterministic policy gradient does not seem to work

Can we try and optimize something else?

Question: Could we replace $\nabla_{\theta_\mu} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\mu]$ with something else?

$$\nabla_{\theta_\mu} V(s_0, \theta_\mu)$$

$$\nabla_{\theta_\mu} Q(s_0, a_0, \theta_\mu)$$

Deep Deterministic Policy Gradient

$$\nabla_{\theta_\mu} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\mu] \qquad \theta_{\pi, i+1} = \theta_{\pi, i} + \alpha \cdot \nabla_{\theta_\mu} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\mu]$$

We failed – a deterministic policy gradient does not seem to work

Can we try and optimize something else?

Question: Could we replace $\nabla_{\theta_\mu} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\mu]$ with something else?

$$\nabla_{\theta_\mu} V(s_0, \theta_\mu)$$

$$\nabla_{\theta_\mu} Q(s_0, a_0, \theta_\mu)$$

In English: Find the policy parameters θ_μ that maximize the value

Deep Deterministic Policy Gradient

$$\nabla_{\theta_\mu} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\mu] \qquad \theta_{\pi, i+1} = \theta_{\pi, i} + \alpha \cdot \nabla_{\theta_\mu} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\mu]$$

We failed – a deterministic policy gradient does not seem to work

Can we try and optimize something else?

Question: Could we replace $\nabla_{\theta_\mu} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\mu]$ with something else?

$$\nabla_{\theta_\mu} V(s_0, \theta_\mu)$$

$$\nabla_{\theta_\mu} Q(s_0, a_0, \theta_\mu)$$

In English: Find the policy parameters θ_μ that maximize the value

Let us figure out the gradient of Q

Deep Deterministic Policy Gradient

$$Q(s_0, a_0, \theta_\mu) = \mathbb{E}[\mathcal{R}(s_1) \mid s_0, a_0] + \gamma Q(s_1, a, \theta_\mu); \quad a = \mu(s_1, \theta_\mu)$$

Deep Deterministic Policy Gradient

$$Q(s_0, a_0, \theta_\mu) = \mathbb{E}[\mathcal{R}(s_1) \mid s_0, a_0] + \gamma Q(s_1, a, \theta_\mu); \quad a = \mu(s_1, \theta_\mu)$$

Plug in μ for a

Deep Deterministic Policy Gradient

$$Q(s_0, a_0, \theta_\mu) = \mathbb{E}[\mathcal{R}(s_1) \mid s_0, a_0] + \gamma Q(s_1, a, \theta_\mu); \quad a = \mu(s_1, \theta_\mu)$$

Plug in μ for a

$$Q(s_0, a_0, \theta_\mu) = \mathbb{E}[\mathcal{R}(s_1) \mid s_0, a_0] + \gamma Q(s_1, \mu(s_1, \theta_\mu), \theta_\mu)$$

Deep Deterministic Policy Gradient

$$Q(s_0, a_0, \theta_\mu) = \mathbb{E}[\mathcal{R}(s_1) \mid s_0, a_0] + \gamma Q(s_1, a, \theta_\mu); \quad a = \mu(s_1, \theta_\mu)$$

Plug in μ for a

$$Q(s_0, a_0, \theta_\mu) = \mathbb{E}[\mathcal{R}(s_1) \mid s_0, a_0] + \gamma Q(s_1, \mu(s_1, \theta_\mu), \theta_\mu)$$

Take the gradient of both sides

Deep Deterministic Policy Gradient

$$Q(s_0, a_0, \theta_\mu) = \mathbb{E}[\mathcal{R}(s_1) \mid s_0, a_0] + \gamma Q(s_1, a, \theta_\mu); \quad a = \mu(s_1, \theta_\mu)$$

Plug in μ for a

$$Q(s_0, a_0, \theta_\mu) = \mathbb{E}[\mathcal{R}(s_1) \mid s_0, a_0] + \gamma Q(s_1, \mu(s_1, \theta_\mu), \theta_\mu)$$

Take the gradient of both sides

$$\nabla_{\theta_\mu} Q(s_0, a_0, \theta_\mu) = \nabla_{\theta_\mu} \left[\mathbb{E}[\mathcal{R}(s_1) \mid s_0, a_0] + \gamma Q(s_1, \mu(s_1, \theta_\mu), \theta_\mu)\right]$$

Deep Deterministic Policy Gradient

$$\nabla_{\theta_\mu} Q(s_0, a_0, \theta_\mu) = \nabla_{\theta_\mu} \left[\mathbb{E}[\mathcal{R}(s_1) \mid s_0, a_0] + \gamma Q(s_1, \mu(s_1, \theta_\mu), \theta_\mu)\right]$$

Deep Deterministic Policy Gradient

$$\nabla_{\theta_\mu} Q(s_0, a_0, \theta_\mu) = \nabla_{\theta_\mu} \left[\mathbb{E}[\mathcal{R}(s_1) \mid s_0, a_0] + \gamma Q(s_1, \mu(s_1, \theta_\mu), \theta_\mu)\right]$$

Initial reward only depends on action, not θ_μ – gradient is zero

Deep Deterministic Policy Gradient

$$\nabla_{\theta_\mu} Q(s_0, a_0, \theta_\mu) = \nabla_{\theta_\mu} \left[\mathbb{E}[\mathcal{R}(s_1) \mid s_0, a_0] + \gamma Q(s_1, \mu(s_1, \theta_\mu), \theta_\mu)\right]$$

Initial reward only depends on action, not θ_μ – gradient is zero

$$\nabla_{\theta_\mu} Q(s_0, a_0, \theta_\mu) = \nabla_{\theta_\mu} \left[\gamma Q(s_1, \mu(s_1, \theta_\mu), \theta_\mu)\right]$$

Deep Deterministic Policy Gradient

$$\nabla_{\theta_\mu} Q(s_0, a_0, \theta_\mu) = \nabla_{\theta_\mu} [\mathbb{E}[\mathcal{R}(s_1) \mid s_0, a_0] + \gamma Q(s_1, \mu(s_1, \theta_\mu), \theta_\mu)]$$

Initial reward only depends on action, not θ_μ – gradient is zero

$$\nabla_{\theta_\mu} Q(s_0, a_0, \theta_\mu) = \nabla_{\theta_\mu} [\gamma Q(s_1, \mu(s_1, \theta_\mu), \theta_\mu)]$$

Pull out γ

Deep Deterministic Policy Gradient

$$\nabla_{\theta_\mu} Q(s_0, a_0, \theta_\mu) = \nabla_{\theta_\mu} [\mathbb{E}[\mathcal{R}(s_1) \mid s_0, a_0] + \gamma Q(s_1, \mu(s_1, \theta_\mu), \theta_\mu)]$$

Initial reward only depends on action, not θ_μ – gradient is zero

$$\nabla_{\theta_\mu} Q(s_0, a_0, \theta_\mu) = \nabla_{\theta_\mu} [\gamma Q(s_1, \mu(s_1, \theta_\mu), \theta_\mu)]$$

Pull out γ

$$\nabla_{\theta_\mu} Q(s_0, a_0, \theta_\mu) = \gamma \nabla_{\theta_\mu} Q(s_1, \mu(s_1, \theta_\mu), \theta_\mu)$$

Deep Deterministic Policy Gradient

$$\nabla_{\theta_\mu} Q(s_0, a_0, \theta_\mu) = \nabla_{\theta_\mu} [\mathbb{E}[\mathcal{R}(s_1) \mid s_0, a_0] + \gamma Q(s_1, \mu(s_1, \theta_\mu), \theta_\mu)]$$

Initial reward only depends on action, not θ_μ – gradient is zero

$$\nabla_{\theta_\mu} Q(s_0, a_0, \theta_\mu) = \nabla_{\theta_\mu} [\gamma Q(s_1, \mu(s_1, \theta_\mu), \theta_\mu)]$$

Pull out γ

$$\nabla_{\theta_\mu} Q(s_0, a_0, \theta_\mu) = \gamma \nabla_{\theta_\mu} Q(s_1, \mu(s_1, \theta_\mu), \theta_\mu)$$

Use chain rule

Deep Deterministic Policy Gradient

$$\nabla_{\theta_\mu} Q(s_0, a_0, \theta_\mu) = \nabla_{\theta_\mu} [\mathbb{E}[\mathcal{R}(s_1) \mid s_0, a_0] + \gamma Q(s_1, \mu(s_1, \theta_\mu), \theta_\mu)]$$

Initial reward only depends on action, not θ_μ – gradient is zero

$$\nabla_{\theta_\mu} Q(s_0, a_0, \theta_\mu) = \nabla_{\theta_\mu} [\gamma Q(s_1, \mu(s_1, \theta_\mu), \theta_\mu)]$$

Pull out γ

$$\nabla_{\theta_\mu} Q(s_0, a_0, \theta_\mu) = \gamma \nabla_{\theta_\mu} Q(s_1, \mu(s_1, \theta_\mu), \theta_\mu)$$

Use chain rule

$$\nabla_{\theta_\mu} Q(s_0, a_0, \theta_\mu) = \gamma \nabla_{\mu} Q(s_1, \mu(s_1, \theta_\mu), \theta_\mu) \cdot \nabla_{\theta_\mu} \mu(s_1, \theta_\mu)$$

Deep Deterministic Policy Gradient

$$\nabla_{\theta_\mu} Q(s_0, a_0, \theta_\mu) = \gamma \; \nabla_\mu Q(s_1, \mu(s_1, \theta_\mu), \theta_\mu) \cdot \nabla_{\theta_\mu} \mu(s_1, \theta_\mu)$$

Let us inspect these terms more closely

Deep Deterministic Policy Gradient

How θ_μ changes a

$$\nabla_{\theta_\mu} Q(s_0, a_0, \theta_\mu) = \gamma \, \nabla_\mu Q(s_1, \mu(s_1, \theta_\mu), \theta_\mu) \cdot \boxed{\nabla_{\theta_\mu} \mu(s_1, \theta_\mu)}$$

Let us inspect these terms more closely

Deep Deterministic Policy Gradient

How θ_μ changes a

$$\nabla_{\theta_\mu} Q(s_0, a_0, \theta_\mu) = \gamma \; \nabla_\mu Q(s_1, \mu(s_1, \theta_\mu), \theta_\mu) \cdot \nabla_{\theta_\mu} \mu(s_1, \theta_\mu)$$

How a changes Q

Let us inspect these terms more closely

The second term is easy to compute, gradient of deterministic policy

Deep Deterministic Policy Gradient

How θ_μ changes a

$$\nabla_{\theta_\mu} Q(s_0, a_0, \theta_\mu) = \gamma \underbrace{\nabla_\mu Q(s_1, \mu(s_1, \theta_\mu), \theta_\mu)}_{\text{How } a \text{ changes } Q} \cdot \nabla_{\theta_\mu} \mu(s_1, \theta_\mu)$$

Let us inspect these terms more closely

The second term is easy to compute, gradient of deterministic policy

Analytical solution for the first term is difficult (recursive definition)

Deep Deterministic Policy Gradient

$$\nabla_{\theta_\mu} Q(s_0, a_0, \theta_\mu) = \gamma \underbrace{\nabla_\mu Q(s_1, \mu(s_1, \theta_\mu), \theta_\mu)}_{\text{How } a \text{ changes } Q} \cdot \overbrace{\nabla_{\theta_\mu} \mu(s_1, \theta_\mu)}^{\text{How } \theta_\mu \text{ changes } a}$$

Let us inspect these terms more closely

The second term is easy to compute, gradient of deterministic policy

Analytical solution for the first term is difficult (recursive definition)

But what if Q is a neural network?

Deep Deterministic Policy Gradient

How θ_μ changes a

$$\nabla_{\theta_\mu} Q(s_0, a_0, \theta_\mu) = \gamma \nabla_\mu Q(s_1, \mu(s_1, \theta_\mu), \theta_\mu) \cdot \nabla_{\theta_\mu} \mu(s_1, \theta_\mu)$$

How a changes Q

Deep Deterministic Policy Gradient

How θ_μ changes a

$$\nabla_{\theta_\mu} Q(s_0, a_0, \theta_\mu) = \gamma \, \nabla_\mu Q(s_1, \mu(s_1, \theta_\mu), \theta_\mu) \cdot \nabla_{\theta_\mu} \mu(s_1, \theta_\mu)$$

How a changes Q

Writing the code makes it look easy

Deep Deterministic Policy Gradient

How θ_μ changes a

$$\nabla_{\theta_\mu} Q(s_0, a_0, \theta_\mu) = \gamma \nabla_\mu Q(s_1, \mu(s_1, \theta_\mu), \theta_\mu) \cdot \nabla_{\theta_\mu} \mu(s_1, \theta_\mu)$$

How a changes Q

Writing the code makes it look easy

```
def Q(s, a, Q_nn, mu_nn):
    a = mu_nn(s)
    return Q_nn(s, a)

# Optimize policy to maximize Q
# Make sure to differentiate w.r.t mu parameters!
J = grad(Q, argnums=3)(states, actions, Q_nn, mu_nn)
mu_nn = optimizer.update(mu_nn, J)
```

Deep Deterministic Policy Gradient

This assumes we know the Q function for θ_μ

Deep Deterministic Policy Gradient

This assumes we know the Q function for θ_μ

We can learn Q for any policy (before we focused on greedy)

Deep Deterministic Policy Gradient

This assumes we know the Q function for θ_μ

We can learn Q for any policy (before we focused on greedy)

$$Q(s_0, a_0, \theta_\mu, \theta_Q) = \mathbb{E}[\mathcal{R}(s_1) \mid s_0, a_0] + \gamma Q(s_1, \mu(s_1, \theta_\mu), \theta_\mu, \theta_Q)$$

Deep Deterministic Policy Gradient

This assumes we know the Q function for θ_μ

We can learn Q for any policy (before we focused on greedy)

$$Q(s_0, a_0, \theta_\mu, \theta_Q) = \mathbb{E}[\mathcal{R}(s_1) \mid s_0, a_0] + \gamma Q(s_1, \mu(s_1, \theta_\mu), \theta_\mu, \theta_Q)$$

```
def Q(s, a, Q_nn, mu_nn):
    a = mu_nn(s)
    return Q_nn(s, a)
# Before, we learned policy params to maximize Q
# Now, we learn params of Q following policy (argnums=2)
J = grad(Q, argnums=2)(states, actions, Q_nn, mu_nn)
Q_nn = optimizer.update(Q_nn, J)
```

Deep Deterministic Policy Gradient

Definition: Deep Deterministic Policy Gradient (DDPG) jointly learns a Q function for deterministic policy μ, and the policy parameters θ_μ

Deep Deterministic Policy Gradient

Definition: Deep Deterministic Policy Gradient (DDPG) jointly learns a Q function for deterministic policy μ, and the policy parameters θ_μ

Step 1: Learn a Q function for μ

Deep Deterministic Policy Gradient

Definition: Deep Deterministic Policy Gradient (DDPG) jointly learns a Q function for deterministic policy μ, and the policy parameters θ_μ

Step 1: Learn a Q function for μ

$$\theta_{Q,i+1} = \arg\min_{\theta_{Q,i}}$$

$$\left(Q\left(s_0, a_0, \theta_{\mu,i}, \theta_{Q,i}\right) - \left(\hat{\mathbb{E}}[\mathcal{R}(s_1) \mid s_0, a_0] + \gamma Q\left(s_1, \mu\left(s_1, \theta_{\mu,i}\right), \theta_{\mu,i}, \theta_{Q,i}\right)\right)\right)$$

Deep Deterministic Policy Gradient

Definition: Deep Deterministic Policy Gradient (DDPG) jointly learns a Q function for deterministic policy μ, and the policy parameters θ_μ

Step 1: Learn a Q function for μ

$$\theta_{Q,i+1} = \arg\min_{\theta_{Q,i}}$$

$$\left(Q\left(s_0, a_0, \theta_{\mu,i}, \theta_{Q,i}\right) - \left(\hat{\mathbb{E}}[\mathcal{R}(s_1) \mid s_0, a_0] + \gamma Q\left(s_1, \mu\left(s_1, \theta_{\mu,i}\right), \theta_{\mu,i}, \theta_{Q,i}\right)\right)\right)$$

Step 2: Learn θ_μ that maximizes Q

Deep Deterministic Policy Gradient

Definition: Deep Deterministic Policy Gradient (DDPG) jointly learns a Q function for deterministic policy μ, and the policy parameters θ_μ

Step 1: Learn a Q function for μ

$$\theta_{Q,i+1} = \arg\min_{\theta_{Q,i}}$$

$$\left(Q\left(s_0, a_0, \theta_{\mu,i}, \theta_{Q,i}\right) - \left(\hat{\mathbb{E}}[\mathcal{R}(s_1) \mid s_0, a_0] + \gamma Q\left(s_1, \mu\left(s_1, \theta_{\mu,i}\right), \theta_{\mu,i}, \theta_{Q,i}\right)\right)\right)$$

Step 2: Learn θ_μ that maximizes Q

$$\theta_{\mu,i+1} = \theta_{\mu,i} + \alpha \cdot Q\left(s_0, \mu\left(s_0, \theta_\mu\right), \theta_{\mu,i}, \theta_{Q,i+1}\right)$$

Deep Deterministic Policy Gradient

Definition: Deep Deterministic Policy Gradient (DDPG) jointly learns a Q function for deterministic policy μ, and the policy parameters θ_μ

Step 1: Learn a Q function for μ

$$\theta_{Q,i+1} = \arg\min_{\theta_{Q,i}}$$

$$\left(Q(s_0, a_0, \theta_{\mu,i}, \theta_{Q,i}) - \left(\hat{\mathbb{E}}[\mathcal{R}(s_1) \mid s_0, a_0] + \gamma Q(s_1, \mu(s_1, \theta_{\mu,i}), \theta_{\mu,i}, \theta_{Q,i})\right)\right)$$

Step 2: Learn θ_μ that maximizes Q

$$\theta_{\mu,i+1} = \theta_{\mu,i} + \alpha \cdot Q(s_0, \mu(s_0, \theta_\mu), \theta_{\mu,i}, \theta_{Q,i+1})$$

Repeat until convergence, $\theta_{\mu,i+1} = \theta_{\mu,i}, \quad \theta_{Q,i+1} = \theta_{Q,i}$

Deep Deterministic Policy Gradient

$$\theta_{Q,i+1} = \arg\min_{\theta_{Q,i}}$$

$$\Big(Q\big(s_0, a_0, \theta_{\mu,i}, \theta_{Q,i}\big) - \Big(\hat{\mathbb{E}}[\mathcal{R}(s_1) \mid s_0, a_0] + \gamma Q\big(s_1, \mu(s_1, \theta_{\mu,i}), \theta_{\mu,i}, \theta_{Q,i}\big)\Big)\Big)$$

$$\theta_{\mu,i+1} = \theta_{\mu,i} + \alpha \cdot Q\big(s_0, \mu(s_0, \theta_\mu), \theta_{\mu,i}, \theta_{Q,i+1}\big)$$

Deep Deterministic Policy Gradient

$$\theta_{Q,i+1} = \underset{\theta_{Q,i}}{\arg\min}$$

$$\left(Q\left(s_0, a_0, \theta_{\mu,i}, \theta_{Q,i}\right) - \left(\hat{\mathbb{E}}[\mathcal{R}(s_1) \mid s_0, a_0] + \gamma Q\left(s_1, \mu\left(s_1, \theta_{\mu,i}\right), \theta_{\mu,i}, \theta_{Q,i}\right)\right)\right)$$

$$\theta_{\mu,i+1} = \theta_{\mu,i} + \alpha \cdot Q\left(s_0, \mu\left(s_0, \theta_{\mu}\right), \theta_{\mu,i}, \theta_{Q,i+1}\right)$$

Question: Is DDPG on-policy or off-policy? Why?

Deep Deterministic Policy Gradient

$$\theta_{Q,i+1} = \arg\min_{\theta_{Q,i}}$$

$$\Big(Q\big(s_0, a_0, \theta_{\mu,i}, \theta_{Q,i}\big) - \Big(\hat{\mathbb{E}}[\mathcal{R}(s_1) \mid s_0, a_0] + \gamma Q\big(s_1, \mu\big(s_1, \theta_{\mu,i}\big), \theta_{\mu,i}, \theta_{Q,i}\big)\Big)\Big)$$

$$\theta_{\mu,i+1} = \theta_{\mu,i} + \alpha \cdot Q\big(s_0, \mu\big(s_0, \theta_\mu\big), \theta_{\mu,i}, \theta_{Q,i+1}\big)$$

Question: Is DDPG on-policy or off-policy? Why?

Answer: Depends on how we train Q:

Deep Deterministic Policy Gradient

$$\theta_{Q,i+1} = \arg\min_{\theta_{Q,i}}$$

$$\Big(Q\big(s_0, a_0, \theta_{\mu,i}, \theta_{Q,i}\big) - \Big(\hat{\mathbb{E}}[\mathcal{R}(s_1) \mid s_0, a_0] + \gamma Q\big(s_1, \mu\big(s_1, \theta_{\mu,i}\big), \theta_{\mu,i}, \theta_{Q,i}\big)\Big)\Big)$$

$$\theta_{\mu,i+1} = \theta_{\mu,i} + \alpha \cdot Q\big(s_0, \mu\big(s_0, \theta_\mu\big), \theta_{\mu,i}, \theta_{Q,i+1}\big)$$

Question: Is DDPG on-policy or off-policy? Why?

Answer: Depends on how we train Q:

- TD Q is off-policy

Deep Deterministic Policy Gradient

$$\theta_{Q,i+1} = \arg\min_{\theta_{Q,i}}$$

$$\Big(Q\big(s_0, a_0, \theta_{\mu,i}, \theta_{Q,i}\big) - \Big(\hat{\mathbb{E}}[\mathcal{R}(s_1) \mid s_0, a_0] + \gamma Q\big(s_1, \mu\big(s_1, \theta_{\mu,i}\big), \theta_{\mu,i}, \theta_{Q,i}\big)\Big)\Big)$$

$$\theta_{\mu,i+1} = \theta_{\mu,i} + \alpha \cdot Q\big(s_0, \mu\big(s_0, \theta_\mu\big), \theta_{\mu,i}, \theta_{Q,i+1}\big)$$

Question: Is DDPG on-policy or off-policy? Why?

Answer: Depends on how we train Q:
- TD Q is off-policy
- MC Q is on-policy

Deep Deterministic Policy Gradient

$$\theta_{Q,i+1} = \arg \min_{\theta_{Q,i}}$$

$$\left(Q\left(s_0, a_0, \theta_{\mu,i}, \theta_{Q,i}\right) - \left(\hat{\mathbb{E}}[\mathcal{R}(s_1) \mid s_0, a_0] + \gamma Q\left(s_1, \mu\left(s_1, \theta_{\mu,i}\right), \theta_{\mu,i}, \theta_{Q,i}\right)\right)\right)$$

$$\theta_{\mu,i+1} = \theta_{\mu,i} + \alpha \cdot Q\left(s_0, \mu\left(s_0, \theta_\mu\right), \theta_{\mu,i}, \theta_{Q,i+1}\right)$$

Question: Is DDPG on-policy or off-policy? Why?

Answer: Depends on how we train Q:

- TD Q is off-policy
- MC Q is on-policy

Almost **all** good off-policy actor-critic algorithms are based on DDPG

Deep Deterministic Policy Gradient

Summary:

Deep Deterministic Policy Gradient

Summary:

- Wanted to extend Q learning to continuous action spaces

Deep Deterministic Policy Gradient

Summary:

- Wanted to extend Q learning to continuous action spaces
  - Simple greedy policy does not work!

Deep Deterministic Policy Gradient

Summary:

- Wanted to extend Q learning to continuous action spaces
  - Simple greedy policy does not work!
- Introduced deterministic policy $a = \mu(s, \theta_\mu)$

Deep Deterministic Policy Gradient

Summary:

- Wanted to extend Q learning to continuous action spaces
  - Simple greedy policy does not work!
- Introduced deterministic policy $a = \mu(s, \theta_\mu)$
- Failed to find $\nabla_{\theta_\mu} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\mu]$

Deep Deterministic Policy Gradient

Summary:

- Wanted to extend Q learning to continuous action spaces
  - Simple greedy policy does not work!
- Introduced deterministic policy $a = \mu(s, \theta_\mu)$
- Failed to find $\nabla_{\theta_\mu} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\mu]$
  - Must know Tr and $\nabla \mathrm{Tr}$

Deep Deterministic Policy Gradient

Summary:

- Wanted to extend Q learning to continuous action spaces
  - Simple greedy policy does not work!
- Introduced deterministic policy $a = \mu(s, \theta_\mu)$
- Failed to find $\nabla_{\theta_\mu} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\mu]$
  - Must know Tr and $\nabla \mathrm{Tr}$
- Trick: Gradient ascent using $\nabla_{\theta_\mu} Q$ instead of $\nabla_{\theta_\mu} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\mu]$

Deep Deterministic Policy Gradient

Summary:

- Wanted to extend Q learning to continuous action spaces
  - Simple greedy policy does not work!
- Introduced deterministic policy $a = \mu(s, \theta_\mu)$
- Failed to find $\nabla_{\theta_\mu} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\mu]$
  - Must know Tr and ∇ Tr
- Trick: Gradient ascent using $\nabla_{\theta_\mu} Q$ instead of $\nabla_{\theta_\mu} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\mu]$
- Iterative optimization of θ_Q and θ_μ

Deep Deterministic Policy Gradient

Summary:

- Wanted to extend Q learning to continuous action spaces
  - Simple greedy policy does not work!
- Introduced deterministic policy $a = \mu(s, \theta_\mu)$
- Failed to find $\nabla_{\theta_\mu} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\mu]$
  - Must know Tr and ∇ Tr
- Trick: Gradient ascent using $\nabla_{\theta_\mu} Q$ instead of $\nabla_{\theta_\mu} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\mu]$
- Iterative optimization of θ_Q and θ_μ

Another way to think of DDPG:

Deep Deterministic Policy Gradient

Summary:

- Wanted to extend Q learning to continuous action spaces
  - Simple greedy policy does not work!
- Introduced deterministic policy $a = \mu(s, \theta_\mu)$
- Failed to find $\nabla_{\theta_\mu} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\mu]$
  - Must know Tr and ∇ Tr
- Trick: Gradient ascent using $\nabla_{\theta_\mu} Q$ instead of $\nabla_{\theta_\mu} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\mu]$
- Iterative optimization of θ_Q and θ_μ

Another way to think of DDPG:

- μ neural network approximation of $\arg\max_{a \in A} Q(s, a)$

Deep Deterministic Policy Gradient

Summary:

- Wanted to extend Q learning to continuous action spaces
  - Simple greedy policy does not work!
- Introduced deterministic policy $a = \mu(s, \theta_\mu)$
- Failed to find $\nabla_{\theta_\mu} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\mu]$
  - Must know Tr and ∇ Tr
- Trick: Gradient ascent using $\nabla_{\theta_\mu} Q$ instead of $\nabla_{\theta_\mu} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\mu]$
- Iterative optimization of θ_Q and θ_μ

Another way to think of DDPG:

- μ neural network approximation of $\arg\max_{a \in A} Q(s, a)$
- Policy learning is learning arg max over infinite action space

Deep Deterministic Policy Gradient

One small problem: deterministic policy means no exploration!

Deep Deterministic Policy Gradient

One small problem: deterministic policy means no exploration!

We must visit other states/actions to find optimal θ_Q, θ_μ

Deep Deterministic Policy Gradient

One small problem: deterministic policy means no exploration!

We must visit other states/actions to find optimal θ_Q, θ_μ

With Q learning, we had epsilon greedy policy

Deep Deterministic Policy Gradient

One small problem: deterministic policy means no exploration!

We must visit other states/actions to find optimal θ_Q, θ_μ

With Q learning, we had epsilon greedy policy

$$\pi(a \mid s; \theta_\pi) = \begin{cases} 1 - \varepsilon : a = \arg\max_{a \in A} Q(s, a, \theta_\pi) \\ \varepsilon : \text{uniform}(A) \end{cases}$$

Deep Deterministic Policy Gradient

One small problem: deterministic policy means no exploration!

We must visit other states/actions to find optimal θ_Q, θ_μ

With Q learning, we had epsilon greedy policy

$$\pi(a \mid s; \theta_\pi) = \begin{cases} 1 - \varepsilon : a = \arg\max_{a \in A} Q(s, a, \theta_\pi) \\ \varepsilon : \text{uniform}(A) \end{cases}$$

Question: What about DDPG exploration? HINT: Continuous actions

Deep Deterministic Policy Gradient

One small problem: deterministic policy means no exploration!

We must visit other states/actions to find optimal θ_Q, θ_μ

With Q learning, we had epsilon greedy policy

$$\pi(a \mid s; \theta_\pi) = \begin{cases} 1 - \varepsilon : a = \arg\max_{a \in A} Q(s, a, \theta_\pi) \\ \varepsilon : \text{uniform}(A) \end{cases}$$

Question: What about DDPG exploration? HINT: Continuous actions

Add some random noise to the action $a = \mu(s, \mu_\pi) + \eta$

Deep Deterministic Policy Gradient

One small problem: deterministic policy means no exploration!

We must visit other states/actions to find optimal θ_Q, θ_μ

With Q learning, we had epsilon greedy policy

$$\pi(a \mid s; \theta_\pi) = \begin{cases} 1 - \varepsilon : a = \arg\max_{a \in A} Q(s, a, \theta_\pi) \\ \varepsilon : \text{uniform}(A) \end{cases}$$

Question: What about DDPG exploration? HINT: Continuous actions

Add some random noise to the action $a = \mu(s, \mu_\pi) + \eta$

$$\pi(a \mid s; \mu_\pi) = \text{Normal}(\mu(s, \mu_\pi), \sigma)$$

Coding

Coding

Like policy gradient, the math and code is different

Coding

Like policy gradient, the math and code is different

- Construct μ and Q networks

Coding

Like policy gradient, the math and code is different

- Construct μ and Q networks
- Sample actions

Coding

Like policy gradient, the math and code is different

- Construct μ and Q networks
- Sample actions
  - Be careful that random actions are in action space!

Coding

Like policy gradient, the math and code is different

- Construct μ and Q networks
- Sample actions
  - Be careful that random actions are in action space!
  - $A = [0, 2\pi]$, then $a = 2.1\pi$ not ok

Coding

Like policy gradient, the math and code is different

- Construct μ and Q networks
- Sample actions
  - Be careful that random actions are in action space!
  - $A = [0, 2\pi]$, then $a = 2.1\pi$ not ok
- Train networks

Coding

First, construct deterministic policy

Coding

First, construct deterministic policy

```
mu = Sequential([
    Linear(state_size, hidden_size),
    Lambda(leaky_relu),
    Linear(hidden_size, hidden_size),
    Lambda(leaky_relu),
    Linear(hidden_size, action_dims),
])
```

Coding

First, construct deterministic policy

```
mu = Sequential([
    Linear(state_size, hidden_size),
    Lambda(leaky_relu),
    Linear(hidden_size, hidden_size),
    Lambda(leaky_relu),
    Linear(hidden_size, action_dims),
])
```

Here, `action_dims` correspond to the number of continuous dimensions

Coding

First, construct deterministic policy

```
mu = Sequential([
    Linear(state_size, hidden_size),
    Lambda(leaky_relu),
    Linear(hidden_size, hidden_size),
    Lambda(leaky_relu),
    Linear(hidden_size, action_dims),
])
```

Here, `action_dims` correspond to the number of continuous dimensions

BenBen: $A = [0, 2\pi]^{12}$, so `action_dims=12`

Coding

Now, we need to make sure actions do not leave action space!

Coding

Now, we need to make sure actions do not leave action space!

- BenBen: $A \in [0, 2\pi]^{12}$, `lower=[0, 0, ...]`, `upper=[2pi, 2pi, ...]`

tanh keeps actions in $[-1, 1]$, scale tanh to action space limits

Coding

Now, we need to make sure actions do not leave action space!

- BenBen: $A \in [0, 2\pi]^{12}$, `lower=[0, 0, ...]`, `upper=[2pi, 2pi, ...]`

tanh keeps actions in $[-1, 1]$, scale tanh to action space limits

- Can also use `clip(action, lower, upper)`, but this zeros gradients

Coding

Coding

Now, we need to make sure actions do not leave action space!

- BenBen: $A \in [0, 2\pi]^{12}$, `lower=[0, 0, ...]`, `upper=[2pi, 2pi, ...]`

tanh keeps actions in $[-1, 1]$, scale tanh to action space limits

- Can also use `clip(action, lower, upper)`, but this zeros gradients
- Agent can get stuck taking limiting actions (e.g., 0 or 2π)

Coding

Now, we need to make sure actions do not leave action space!

- BenBen: $A \in [0, 2\pi]^{12}$, `lower=[0, 0, ...]`, `upper=[2pi, 2pi, ...]`

tanh keeps actions in $[-1, 1]$, scale tanh to action space limits

- Can also use `clip(action, lower, upper)`, but this zeros gradients
- Agent can get stuck taking limiting actions (e.g., 0 or 2π)

```
def bound_action(action, lower, upper):
    return 0.5 * (upper + lower) + 0.5 * (upper - lower)
        * tanh(action)
```

Coding

Now, we need to make sure actions do not leave action space!

- BenBen: $A \in [0, 2\pi]^{12}$, `lower=[0, 0, ...]`, `upper=[2pi, 2pi, ...]`

tanh keeps actions in $[-1, 1]$, scale tanh to action space limits

- Can also use `clip(action, lower, upper)`, but this zeros gradients
- Agent can get stuck taking limiting actions (e.g., 0 or 2π)

```
def bound_action(action, lower, upper):
    return 0.5 * (upper + lower) + 0.5 * (upper - lower)
        * tanh(action)

def sample_action(mu, state, A_bounds, std):
    action = mu(state)
    noisy_action = action + normal(0, std) # Explore
    return bound_action(noisy_action, *A_bounds)
```

Coding

Now construct Q

Coding

Now construct Q

```
Q = Sequential([
    # Different from DQN network
    # Input action and state together
    Lambda(lambda s, a: concatenate(s, a)),
    Linear(state_size + action_dims, hidden_size),
    Lambda(leaky_relu),
    Linear(hidden_size, hidden_size),
    Lambda(leaky_relu),
    Linear(hidden_size, 1), # Single value for Q(s, a)
])
```

Coding

```
while not terminated:
    # Exploration: make sure actions within action space!
    action = sample_action(mu, state, bounds, std)
    transition = env.step(action)
    replay_buffer.append(transition)
    data = replay_buffer.sample()
    # Theta_pi params are in mu neural network
    # Argnums tells us differentiation variable
    J_Q = grad(Q_loss, argnums=0)(theta_Q, theta_T, mu, data)
    J_mu = grad(mu_loss, argnums=0)(mu, theta_Q, data)
    theta_Q, mu = apply_updates(J_Q, J_mu, ...)
    if step % 200 == 0: # Target network necessary
        theta_T = theta_Q
```

Coding

```
def Q_loss(theta_Q, theta_T, theta_pi, data):
    Qnet = combine(Q, theta_Q)
    Tnet = combine(Q, theta_T) # Target network
    # Predict Q values for action we took
    prediction = vmap(Qnet)(data.state, data.action)
    # Now compute labels
    next_action = vmap(mu)(data.next_state)
    # NOTE: No argmax! Mu approximates argmax
    next_Q = vmap(Tnet)(data.next_state, next_action)
    label = reward + gamma * data.done * next_Q
    return (prediction - label) ** 2
```

Coding

```
def mu_loss(mu, theta_Q, data):
    # Find the action that maximizes the Q function
    Qnet = combine(Q, theta_Q)
    # Instead of multiply, chain rule -- plug action into Q
    action = vmap(mu)(data.state)
    q_value = vmap(Qnet)(data.state, action)
    # Update the policy parameters to maximize the Q value
    # Gradient ascent but we min loss, use negative
    return -q_value
```

Max Entropy RL

Max Entropy RL

Many algorithms add improvements to DDPG

Max Entropy RL

Many algorithms add improvements to DDPG

The most popular algorithm based on DDPG is **Soft Actor Critic** (SAC)

Max Entropy RL

Many algorithms add improvements to DDPG

The most popular algorithm based on DDPG is **Soft Actor Critic** (SAC)

SAC is arguably the "best" model-free algorithm

Max Entropy RL

Many algorithms add improvements to DDPG

The most popular algorithm based on DDPG is **Soft Actor Critic** (SAC)

SAC is arguably the "best" model-free algorithm

The motivation for SAC comes from **max-entropy RL**

Max Entropy RL

Many algorithms add improvements to DDPG

The most popular algorithm based on DDPG is **Soft Actor Critic** (SAC)

SAC is arguably the “best” model-free algorithm

The motivation for SAC comes from **max-entropy RL**

We will very briefly cover max-entropy RL

Max Entropy RL

Many algorithms add improvements to DDPG

The most popular algorithm based on DDPG is **Soft Actor Critic** (SAC)

SAC is arguably the “best” model-free algorithm

The motivation for SAC comes from **max-entropy RL**

We will very briefly cover max-entropy RL

First, let us introduce entropy

Max Entropy RL

Entropy measures the uncertainty of a distribution

Max Entropy RL

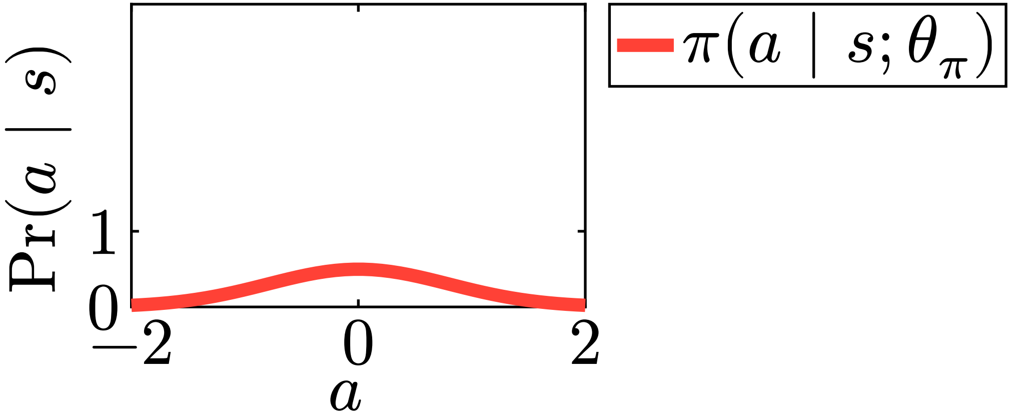
Entropy measures the uncertainty of a distribution

$$H(\pi(a \mid s; \theta_\pi))$$

Max Entropy RL

Entropy measures the uncertainty of a distribution

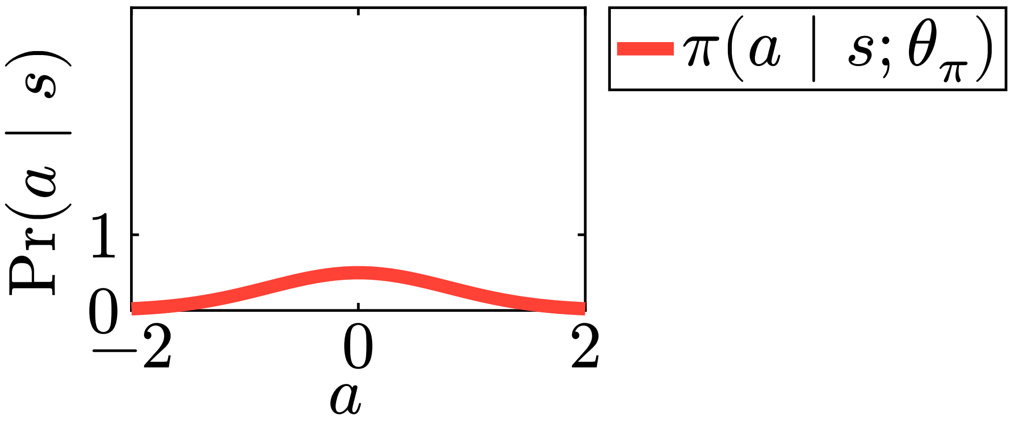
$$H(\pi(a \mid s; \theta_\pi))$$

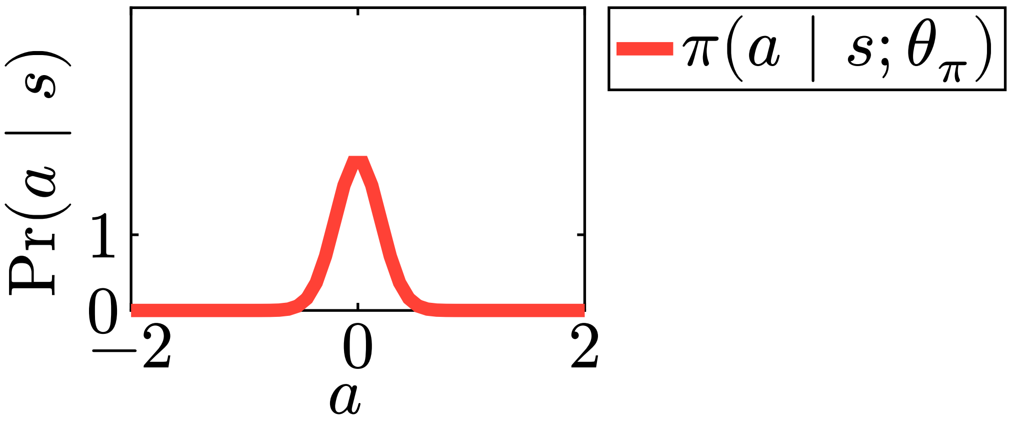


Max Entropy RL

Entropy measures the uncertainty of a distribution

$$H(\pi(a \mid s; \theta_\pi))$$

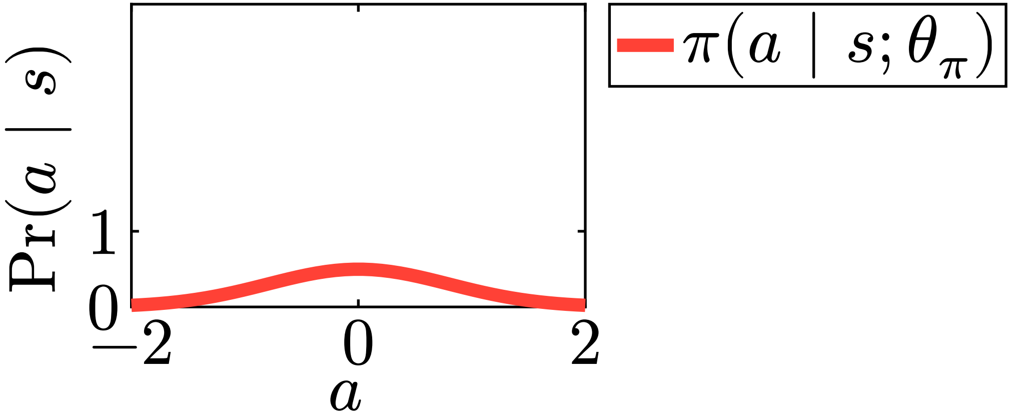


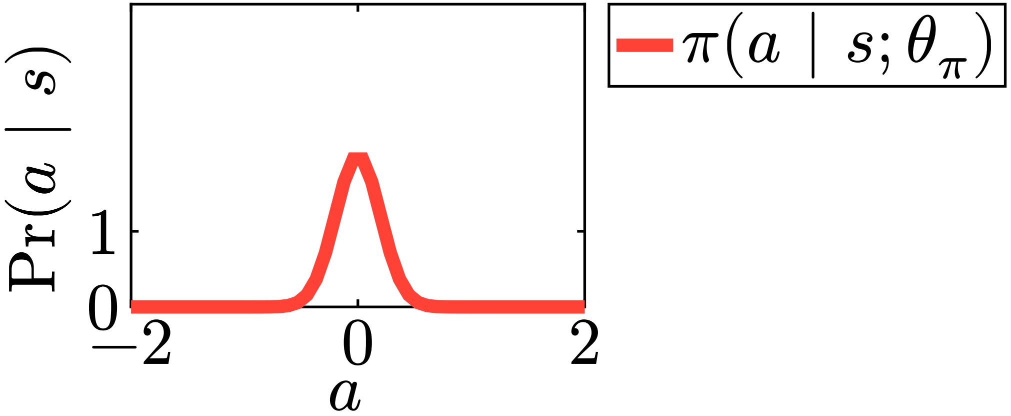


Max Entropy RL

Entropy measures the uncertainty of a distribution

$$H(\pi(a \mid s; \theta_\pi))$$



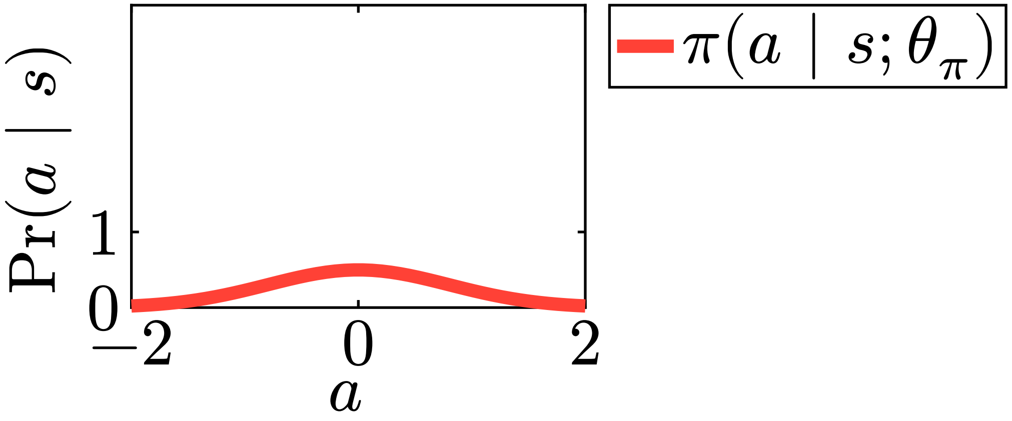


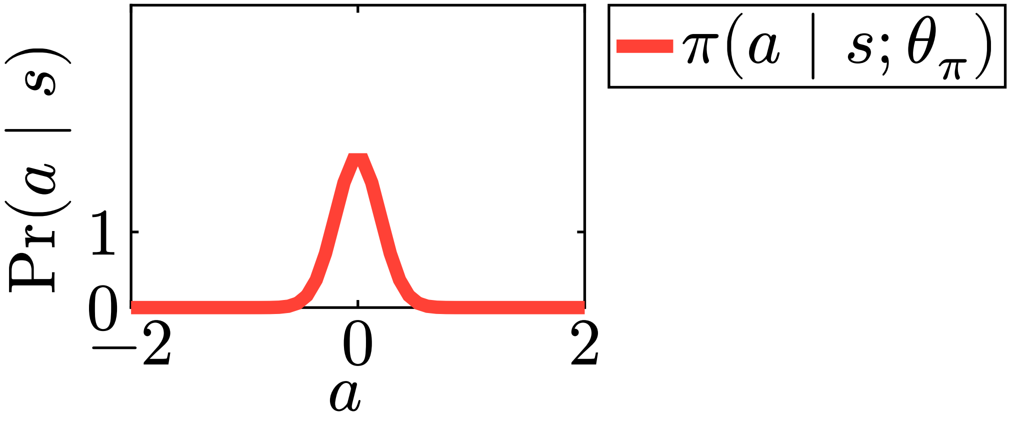
Question: Which policy has higher entropy?

Max Entropy RL

Entropy measures the uncertainty of a distribution

$$H(\pi(a \mid s; \theta_\pi))$$





Question: Which policy has higher entropy?

Left policy, more uncertain/random

Max Entropy RL

In maximum entropy RL, we change the objective

Max Entropy RL

In maximum entropy RL, we change the objective

$$\arg\max_{\theta_\pi} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\pi] = \arg\max_{\theta_\pi} \sum_{t=0}^{\infty} \gamma^t \mathbb{E}[\mathcal{R}(s_{t+1}) \mid s_0; \theta_\pi]$$

Max Entropy RL

In maximum entropy RL, we change the objective

$$\arg\max_{\theta_\pi} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\pi] = \arg\max_{\theta_\pi} \sum_{t=0}^{\infty} \gamma^t \mathbb{E}[\mathcal{R}(s_{t+1}) \mid s_0; \theta_\pi]$$

We consider the entropy in the return

Max Entropy RL

In maximum entropy RL, we change the objective

$$\arg\max_{\theta_\pi} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\pi] = \arg\max_{\theta_\pi} \sum_{t=0}^{\infty} \gamma^t \mathbb{E}[\mathcal{R}(s_{t+1}) \mid s_0; \theta_\pi]$$

We consider the entropy in the return

$$\arg\max_{\theta_\pi} \mathbb{E}[\mathcal{H}(\boldsymbol{\tau}) \mid s_0; \theta_\pi] =$$

$$\arg\max_{\theta_\pi} \sum_{t=0}^{\infty} \gamma^t \mathbb{E}[\mathcal{R}(s_{t+1}) \mid s_0; \theta_\pi] + \gamma^t H(\pi(a \mid s_t; \theta_\pi))$$

Max Entropy RL

In maximum entropy RL, we change the objective

$$\arg\max_{\theta_\pi} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\pi] = \arg\max_{\theta_\pi} \sum_{t=0}^{\infty} \gamma^t \mathbb{E}[\mathcal{R}(s_{t+1}) \mid s_0; \theta_\pi]$$

We consider the entropy in the return

$$\arg\max_{\theta_\pi} \mathbb{E}[\mathcal{H}(\boldsymbol{\tau}) \mid s_0; \theta_\pi] =$$

$$\arg\max_{\theta_\pi} \sum_{t=0}^{\infty} \gamma^t \mathbb{E}[\mathcal{R}(s_{t+1}) \mid s_0; \theta_\pi] + \gamma^t H(\pi(a \mid s_t; \theta_\pi))$$

We want a policy that is both random and maximizes the return

Max Entropy RL

$$\arg\max_{\theta_\pi} \mathbb{E}[\mathcal{H}(\boldsymbol{\tau}) \mid s_0; \theta_\pi] =$$

$$\arg\max_{\theta_\pi} \sum_{t=0}^{\infty} \gamma^t (\mathbb{E}[\mathcal{R}(s_{t+1}) \mid s_0; \theta_\pi] + H(\pi(\cdot \mid s_t; \theta_\pi)))$$

Max Entropy RL

$$\arg\max_{\theta_\pi} \mathbb{E}[\mathcal{H}(\boldsymbol{\tau}) \mid s_0; \theta_\pi] =$$

$$\arg\max_{\theta_\pi} \sum_{t=0}^{\infty} \gamma^t (\mathbb{E}[\mathcal{R}(s_{t+1}) \mid s_0; \theta_\pi] + H(\pi(\cdot \mid s_t; \theta_\pi)))$$

Question: Why do want policy entropy? Lowers the return

Answer: No good theoretical reason, helpful in practice

Max Entropy RL

$$\arg\max_{\theta_\pi} \mathbb{E}[\mathcal{H}(\boldsymbol{\tau}) \mid s_0; \theta_\pi] =$$

$$\arg\max_{\theta_\pi} \sum_{t=0}^{\infty} \gamma^t (\mathbb{E}[\mathcal{R}(s_{t+1}) \mid s_0; \theta_\pi] + H(\pi(\cdot \mid s_t; \theta_\pi)))$$

Question: Why do want policy entropy? Lowers the return

Answer: No good theoretical reason, helpful in practice

- There are theoretical reasons, but they are not very good

Max Entropy RL

$$\arg\max_{\theta_\pi} \mathbb{E}[\mathcal{H}(\boldsymbol{\tau}) \mid s_0; \theta_\pi] =$$

$$\arg\max_{\theta_\pi} \sum_{t=0}^{\infty} \gamma^t \left(\mathbb{E}[\mathcal{R}(s_{t+1}) \mid s_0; \theta_\pi] + H(\pi(\cdot \mid s_t; \theta_\pi)) \right)$$

Question: Why do want policy entropy? Lowers the return

Answer: No good theoretical reason, helpful in practice

- There are theoretical reasons, but they are not very good
- Better exploration during training

Max Entropy RL

$$\arg\max_{\theta_\pi} \mathbb{E}[\mathcal{H}(\boldsymbol{\tau}) \mid s_0; \theta_\pi] =$$

$$\arg\max_{\theta_\pi} \sum_{t=0}^{\infty} \gamma^t (\mathbb{E}[\mathcal{R}(s_{t+1}) \mid s_0; \theta_\pi] + H(\pi(\cdot \mid s_t; \theta_\pi)))$$

Question: Why do want policy entropy? Lowers the return

Answer: No good theoretical reason, helpful in practice

- There are theoretical reasons, but they are not very good
- Better exploration during training
- More stable training (explain further in offline RL)

Max Entropy RL

DDPG is based on a deterministic policy μ

Max Entropy RL

DDPG is based on a deterministic policy μ

A deterministic policy μ has no entropy (it is not random)

Max Entropy RL

DDPG is based on a deterministic policy μ

A deterministic policy μ has no entropy (it is not random)

But for SAC, I talk about π and policy entropy

Max Entropy RL

DDPG is based on a deterministic policy μ

A deterministic policy μ has no entropy (it is not random)

But for SAC, I talk about π and policy entropy

How is this possible?

Max Entropy RL

DDPG is based on a deterministic policy μ

A deterministic policy μ has no entropy (it is not random)

But for SAC, I talk about π and policy entropy

How is this possible?

Consider a function $f : S \times \Theta \times \mathbb{R} \mapsto A$

Max Entropy RL

DDPG is based on a deterministic policy μ

A deterministic policy μ has no entropy (it is not random)

But for SAC, I talk about π and policy entropy

How is this possible?

Consider a function $f : S \times \Theta \times \mathbb{R} \mapsto A$

$$a = f(s, \theta_\mu, \eta) = \mu(s, \theta_\mu) + \eta$$

Max Entropy RL

$$a = f(s, \theta_\mu, \eta) = \mu(s, \theta_\mu) + \eta$$

Max Entropy RL

$$a = f(s, \theta_\mu, \eta) = \mu(s, \theta_\mu) + \eta$$

If $\eta \sim \text{Normal}(0, 1)$ our function f is still deterministic

Max Entropy RL

$$a = f(s, \theta_\mu, \eta) = \mu(s, \theta_\mu) + \eta$$

If $\eta \sim \text{Normal}(0, 1)$ our function f is still deterministic

We call this the **reparameterization trick**, used in variational autoencoders (VAE)

Max Entropy RL

$$a = f(s, \theta_\mu, \eta) = \mu(s, \theta_\mu) + \eta$$

If $\eta \sim \text{Normal}(0, 1)$ our function f is still deterministic

We call this the **reparameterization trick**, used in variational autoencoders (VAE)

Mathematically, this is a "deterministic" policy

Max Entropy RL

$$a = f(s, \theta_\mu, \eta) = \mu(s, \theta_\mu) + \eta$$

If $\eta \sim \text{Normal}(0, 1)$ our function f is still deterministic

We call this the **reparameterization trick**, used in variational autoencoders (VAE)

Mathematically, this is a "deterministic" policy

$$\mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0, \eta_0, \eta_1, ...; \theta_\mu]$$

Max Entropy RL

$$a = f(s, \theta_\mu, \eta) = \mu(s, \theta_\mu) + \eta$$

$$\mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0, \eta_0, \eta_1, ...; \theta_\mu]$$

In practice, f behaves just like a stochastic policy π

Max Entropy RL

$$a = f(s, \theta_\mu, \eta) = \mu(s, \theta_\mu) + \eta$$

$$\mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0, \eta_0, \eta_1, ...; \theta_\mu]$$

In practice, f behaves just like a stochastic policy π

Gradient descent will learn θ_μ that generalize over η

Max Entropy RL

$$a = f(s, \theta_\mu, \eta) = \mu(s, \theta_\mu) + \eta$$

$$\mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0, \eta_0, \eta_1, ...; \theta_\mu]$$

In practice, f behaves just like a stochastic policy π

Gradient descent will learn θ_μ that generalize over η

We abuse notation and write f as a random policy

Max Entropy RL

$$a = f(s, \theta_\mu, \eta) = \mu(s, \theta_\mu) + \eta$$

$$\mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0, \eta_0, \eta_1, ...; \theta_\mu]$$

In practice, f behaves just like a stochastic policy π

Gradient descent will learn θ_μ that generalize over η

We abuse notation and write f as a random policy

$$\pi(a \mid s; \theta_\pi)$$

Max Entropy RL

$$a = f(s, \theta_\mu, \eta) = \mu(s, \theta_\mu) + \eta$$

$$\mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0, \eta_0, \eta_1, ...; \theta_\mu]$$

In practice, f behaves just like a stochastic policy π

Gradient descent will learn θ_μ that generalize over η

We abuse notation and write f as a random policy

$$\pi(a \mid s; \theta_\pi)$$

But f is deterministic when we know η

Max Entropy RL

What happens when we combine max entropy RL

Max Entropy RL

What happens when we combine max entropy RL

$$\arg \max_{\theta_\pi} \mathbb{E}[\mathcal{H}(\boldsymbol{\tau}) \mid s_0; \theta_\pi] =$$

$$\arg \max_{\theta_\pi} \sum_{t=0}^{\infty} \gamma^t \mathbb{E}[\mathcal{R}(s_{t+1}) \mid s_0; \theta_\pi] + \gamma^t H(\pi(a \mid s_t; \theta_\pi))$$

Max Entropy RL

What happens when we combine max entropy RL

$$\arg\max_{\theta_\pi} \mathbb{E}[\mathcal{H}(\boldsymbol{\tau}) \mid s_0; \theta_\pi] =$$

$$\arg\max_{\theta_\pi} \sum_{t=0}^{\infty} \gamma^t \mathbb{E}[\mathcal{R}(s_{t+1}) \mid s_0; \theta_\pi] + \gamma^t H(\pi(a \mid s_t; \theta_\pi))$$

With the reparameterization trick?

Max Entropy RL

What happens when we combine max entropy RL

$$\arg\max_{\theta_\pi} \mathbb{E}[\mathcal{H}(\boldsymbol{\tau}) \mid s_0; \theta_\pi] =$$

$$\arg\max_{\theta_\pi} \sum_{t=0}^{\infty} \gamma^t \mathbb{E}[\mathcal{R}(s_{t+1}) \mid s_0; \theta_\pi] + \gamma^t H(\pi(a \mid s_t; \theta_\pi))$$

With the reparameterization trick?

$$a = \mu(s, \theta_\mu) + \eta \cong a \sim \pi(\cdot \mid s; \theta_\pi)$$

Max Entropy RL

What happens when we combine max entropy RL

$$\arg\max_{\theta_\pi} \mathbb{E}[\mathcal{H}(\boldsymbol{\tau}) \mid s_0; \theta_\pi] =$$

$$\arg\max_{\theta_\pi} \sum_{t=0}^{\infty} \gamma^t \mathbb{E}[\mathcal{R}(s_{t+1}) \mid s_0; \theta_\pi] + \gamma^t H(\pi(a \mid s_t; \theta_\pi))$$

With the reparameterization trick?

$$a = \mu(s, \theta_\mu) + \eta \cong a \sim \pi(\cdot \mid s; \theta_\pi)$$

We get SAC!

Max Entropy RL

Definition: Soft Actor Critic (SAC) adds a max entropy objective and stochastic policy to DDPG

Max Entropy RL

Definition: Soft Actor Critic (SAC) adds a max entropy objective and stochastic policy to DDPG

Step 1: Learn a Q function for max entropy policy (Q learning)

Max Entropy RL

Definition: Soft Actor Critic (SAC) adds a max entropy objective and stochastic policy to DDPG

Step 1: Learn a Q function for max entropy policy (Q learning)

$$\theta_{Q, i+1} = \arg \min_{\theta_{Q, i}} \left(Q\left(s_0, a_0, \theta_{\pi, i}, \theta_{Q, i}\right) - y \right)^2$$

Max Entropy RL

Definition: Soft Actor Critic (SAC) adds a max entropy objective and stochastic policy to DDPG

Step 1: Learn a Q function for max entropy policy (Q learning)

$$\theta_{Q,i+1} = \arg\min_{\theta_{Q,i}} \left(Q\left(s_0, a_0, \theta_{\pi,i}, \theta_{Q,i}\right) - y\right)^2$$

$$y = \hat{\mathbb{E}}[\mathcal{R}(s_1) \mid s_0, a_0] + H\left(\pi\left(a \mid s_0; \theta_\mu\right)\right) + \gamma Q\left(s_1, \mu\left(s_1, \theta_\mu, \eta\right), \theta_{\pi,i}, \theta_{Q,i}\right)$$

Max Entropy RL

Definition: Soft Actor Critic (SAC) adds a max entropy objective and stochastic policy to DDPG

Step 1: Learn a Q function for max entropy policy (Q learning)

$$\theta_{Q,i+1} = \arg\min_{\theta_{Q,i}} \left(Q\left(s_0, a_0, \theta_{\pi,i}, \theta_{Q,i}\right) - y\right)^2$$

$$y = \hat{\mathbb{E}}[\mathcal{R}(s_1) \mid s_0, a_0] + H\left(\pi\left(a \mid s_0; \theta_\mu\right)\right) + \gamma Q\left(s_1, \mu\left(s_1, \theta_\mu, \eta\right), \theta_{\pi,i}, \theta_{Q,i}\right)$$

Max Entropy RL

Definition: Soft Actor Critic (SAC) adds a max entropy objective and stochastic policy to DDPG

Step 1: Learn a Q function for max entropy policy (Q learning)

$$\theta_{Q,i+1} = \arg\min_{\theta_{Q,i}} \left(Q(s_0, a_0, \theta_{\pi,i}, \theta_{Q,i}) - y \right)^2$$

$$y = \underbrace{\hat{\mathbb{E}}[\mathcal{R}(s_1) \mid s_0, a_0]}_{\text{Reward}} + H(\pi(a \mid s_0; \theta_\mu)) + \gamma Q(s_1, \mu(s_1, \theta_\mu, \eta), \theta_{\pi,i}, \theta_{Q,i})$$

Max Entropy RL

Definition: Soft Actor Critic (SAC) adds a max entropy objective and stochastic policy to DDPG

Step 1: Learn a Q function for max entropy policy (Q learning)

$$\theta_{Q,i+1} = \arg\min_{\theta_{Q,i}} \left(Q(s_0, a_0, \theta_{\pi,i}, \theta_{Q,i}) - y \right)^2$$

$$y = \underbrace{\hat{\mathbb{E}}[\mathcal{R}(s_1) \mid s_0, a_0]}_{\text{Reward}} + \underbrace{H(\pi(a \mid s_0; \theta_\mu))}_{\text{Entropy bonus}} + \gamma Q(s_1, \mu(s_1, \theta_\mu, \eta), \theta_{\pi,i}, \theta_{Q,i})$$

Max Entropy RL

Definition: Soft Actor Critic (SAC) adds a max entropy objective and stochastic policy to DDPG

Step 1: Learn a Q function for max entropy policy (Q learning)

$$\theta_{Q,i+1} = \arg\min_{\theta_{Q,i}} \left(Q\left(s_0, a_0, \theta_{\pi,i}, \theta_{Q,i}\right) - y\right)^2$$

$$y = \underbrace{\hat{\mathbb{E}}[\mathcal{R}(s_1) \mid s_0, a_0]}_{\text{Reward}} + \underbrace{H\left(\pi\left(a \mid s_0; \theta_\mu\right)\right)}_{\text{Entropy bonus}} + \gamma Q\left(s_1, \underbrace{\mu\left(s_1, \theta_\mu, \eta\right)}_{\text{Deterministic } a}, \theta_{\pi,i}, \theta_{Q,i}\right)$$

Max Entropy RL

Definition: Soft Actor Critic (SAC) adds a max entropy objective and stochastic policy to DDPG

Step 1: Learn a Q function for max entropy policy (Q learning)

$$\theta_{Q,i+1} = \arg\min_{\theta_{Q,i}} \left(Q\left(s_0, a_0, \theta_{\pi,i}, \theta_{Q,i}\right) - y\right)^2$$

$$y = \underbrace{\hat{\mathbb{E}}[\mathcal{R}(s_1) \mid s_0, a_0]}_{\text{Reward}} + \underbrace{H\left(\pi\left(a \mid s_0; \theta_\mu\right)\right)}_{\text{Entropy bonus}} + \gamma Q\left(s_1, \underbrace{\mu\left(s_1, \theta_\mu, \eta\right)}_{\text{Deterministic } a}, \theta_{\pi,i}, \theta_{Q,i}\right)$$

where η is randomly sampled

Max Entropy RL

Step 2: Learn a π that maximizes Q (policy gradient)

Max Entropy RL

Step 2: Learn a π that maximizes Q (policy gradient)

$$\theta_{\pi, i+1} = \theta_{\pi, i} + \alpha \cdot \underbrace{Q\big(s_0, \mu\big(s_0, \theta_\mu, \eta\big), \theta_\pi, \theta_Q\big)}_{\text{Replaces } \mathbb{E}\left[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\mu\right]}$$

Max Entropy RL

Step 2: Learn a π that maximizes Q (policy gradient)

Deterministic a

$$\theta_{\pi, i+1} = \theta_{\pi, i} + \alpha \cdot \underbrace{Q\left(s_0, \mu\left(s_0, \theta_\mu, \eta\right), \theta_\pi, \theta_Q\right)}_{\text{Replaces } \mathbb{E}\left[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\mu\right]}$$

Max Entropy RL

Step 2: Learn a π that maximizes Q (policy gradient)

Deterministic a

$$\theta_{\pi, i+1} = \theta_{\pi, i} + \alpha \cdot \underbrace{Q\left(s_0, \mu\left(s_0, \theta_\mu, \eta\right), \theta_\pi, \theta_Q\right)}_{\text{Replaces } \mathbb{E}\left[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\mu\right]}$$

where η is randomly sampled

Max Entropy RL

Step 2: Learn a π that maximizes Q (policy gradient)

Deterministic a

$$\theta_{\pi, i+1} = \theta_{\pi, i} + \alpha \cdot \underbrace{Q\left(s_0, \mu\left(s_0, \theta_\mu, \eta\right), \theta_\pi, \theta_Q\right)}_{\text{Replaces } \mathbb{E}\left[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\mu\right]}$$

where η is randomly sampled

Repeat until convergence, $\theta_{\mu, i+1} = \theta_{\mu, i}, \quad \theta_{Q, i+1} = \theta_{Q, i}$

Max Entropy RL

Like PPO, there are many variants of SAC

Max Entropy RL

Like PPO, there are many variants of SAC

- Learn separate value and Q functions

Max Entropy RL

Like PPO, there are many variants of SAC

- Learn separate value and Q functions
- Double Q function

Max Entropy RL

Like PPO, there are many variants of SAC

- Learn separate value and Q functions
- Double Q function
- Lagrangian entropy adjustment

Max Entropy RL

Like PPO, there are many variants of SAC

- Learn separate value and Q functions
- Double Q function
- Lagrangian entropy adjustment
- Reduced variance gradients

Max Entropy RL

Like PPO, there are many variants of SAC

- Learn separate value and Q functions
- Double Q function
- Lagrangian entropy adjustment
- Reduced variance gradients

Like PPO, SAC is complicated – uses many “implementation tricks”

Max Entropy RL

Like PPO, there are many variants of SAC

- Learn separate value and Q functions
- Double Q function
- Lagrangian entropy adjustment
- Reduced variance gradients

Like PPO, SAC is complicated – uses many “implementation tricks”

- Often not documented
- CleanRL describes modern SAC, using tricks from 5+ papers
- https://docs.cleanrl.dev/rl-algorithms/sac/#implementation-details_1

Coding SAC could take an entire lecture, read CleanRL

Max Entropy RL

$\text{DDPG} \approx \text{A2C}$

Max Entropy RL

DDPG $\approx$ A2C

Introduces the concept, simple

Max Entropy RL

DDPG $\approx$ A2C

SAC $\approx$ PPO

Introduces the concept, simple

Max Entropy RL

DDPG $\approx$ A2C

Introduces the concept, simple

SAC $\approx$ PPO

Improves the concept, complex

Max Entropy RL

DDPG $\approx$ A2C

Introduces the concept, simple

I suggest you try DDPG before SAC

SAC $\approx$ PPO

Improves the concept, complex

Max Entropy RL

DDPG $\approx$ A2C

Introduces the concept, simple

SAC $\approx$ PPO

Improves the concept, complex

I suggest you try DDPG before SAC

- DDPG is much easier to implement

Max Entropy RL

DDPG $\approx$ A2C

Introduces the concept, simple

SAC $\approx$ PPO

Improves the concept, complex

I suggest you try DDPG before SAC

- DDPG is much easier to implement
- Fewer hyperparameters

Max Entropy RL

DDPG $\approx$ A2C

Introduces the concept, simple

SAC $\approx$ PPO

Improves the concept, complex

I suggest you try DDPG before SAC

- DDPG is much easier to implement
- Fewer hyperparameters
- Tuned DDPG can likely outperform untuned SAC

Max Entropy RL

What algorithm is best in 2025?

Max Entropy RL

What algorithm is best in 2025?

For discrete actions (Atari), DQN variants still perform best

Max Entropy RL

What algorithm is best in 2025?

For discrete actions (Atari), DQN variants still perform best

For continuous actions (MuJoCo), SAC performs best

Max Entropy RL

What algorithm is best in 2025?

For discrete actions (Atari), DQN variants still perform best

For continuous actions (MuJoCo), SAC performs best

PPO is less sensitive to hyperparameters than SAC/DQN

Max Entropy RL

What algorithm is best in 2025?

For discrete actions (Atari), DQN variants still perform best

For continuous actions (MuJoCo), SAC performs best

PPO is less sensitive to hyperparameters than SAC/DQN

- Noobs like PPO, they cannot tune hyperparameters

Max Entropy RL

What algorithm is best in 2025?

For discrete actions (Atari), DQN variants still perform best

For continuous actions (MuJoCo), SAC performs best

PPO is less sensitive to hyperparameters than SAC/DQN

- Noobs like PPO, they cannot tune hyperparameters
- Often performs worse than tuned DQN/SAC

Max Entropy RL

What algorithm is best in 2025?

For discrete actions (Atari), DQN variants still perform best

For continuous actions (MuJoCo), SAC performs best

PPO is less sensitive to hyperparameters than SAC/DQN
- Noobs like PPO, they cannot tune hyperparameters
- Often performs worse than tuned DQN/SAC

Tuned DDPG/A2C perform 95% as good as tuned SAC/PPO

Max Entropy RL

What algorithm is best in 2025?

For discrete actions (Atari), DQN variants still perform best

For continuous actions (MuJoCo), SAC performs best

PPO is less sensitive to hyperparameters than SAC/DQN
- Noobs like PPO, they cannot tune hyperparameters
- Often performs worse than tuned DQN/SAC

Tuned DDPG/A2C perform 95% as good as tuned SAC/PPO
- Much easier to debug

Final Project Tips

Final Project Tips

Log and plot EVERYTHING

Final Project Tips

Log and plot EVERYTHING

- Losses, mean advantage, mean Q, policy entropy, etc
  - Use these to help debug and tune hyperparameters
  - E.g., exploding losses, decrease learning rate
  - E.g., Q values too large? Increase time between target net update

Final Project Tips

Log and plot EVERYTHING

- Losses, mean advantage, mean Q, policy entropy, etc
  - Use these to help debug and tune hyperparameters
  - E.g., exploding losses, decrease learning rate
  - E.g., Q values too large? Increase time between target net update

If you get stuck, visualize your policy

Final Project Tips

Log and plot EVERYTHING

- Losses, mean advantage, mean Q, policy entropy, etc
  - Use these to help debug and tune hyperparameters
  - E.g., exploding losses, decrease learning rate
  - E.g., Q values too large? Increase time between target net update

If you get stuck, visualize your policy

- Record some episodes (videos/frames/etc)

Final Project Tips

Log and plot EVERYTHING

- Losses, mean advantage, mean Q, policy entropy, etc
  - Use these to help debug and tune hyperparameters
  - E.g., exploding losses, decrease learning rate
  - E.g., Q values too large? Increase time between target net update

If you get stuck, visualize your policy

- Record some episodes (videos/frames/etc)
- Watch the policy, what did it learn to do?

Final Project Tips

Log and plot EVERYTHING

- Losses, mean advantage, mean Q, policy entropy, etc
  - Use these to help debug and tune hyperparameters
  - E.g., exploding losses, decrease learning rate
  - E.g., Q values too large? Increase time between target net update

If you get stuck, visualize your policy

- Record some episodes (videos/frames/etc)
- Watch the policy, what did it learn to do?
  - Think about why it learned to do this (exploiting bugs in MDP)

Final Project Tips

Why does Steven spend so much time on theory instead of coding?

Final Project Tips

Why does Steven spend so much time on theory instead of coding?

In supervised learning, follow MNIST tutorial and everything works

Final Project Tips

Why does Steven spend so much time on theory instead of coding?

In supervised learning, follow MNIST tutorial and everything works

This is **NOT** the case in RL

Final Project Tips

Why does Steven spend so much time on theory instead of coding?

In supervised learning, follow MNIST tutorial and everything works

This is **NOT** the case in RL

In RL, your code never works on the first try

Final Project Tips

Why does Steven spend so much time on theory instead of coding?

In supervised learning, follow MNIST tutorial and everything works

This is **NOT** the case in RL

In RL, your code never works on the first try

Even if it is correct, you need to play with hyperparameters

Final Project Tips

Why does Steven spend so much time on theory instead of coding?

In supervised learning, follow MNIST tutorial and everything works

This is **NOT** the case in RL

In RL, your code never works on the first try

Even if it is correct, you need to play with hyperparameters

Theory is absolutely necessary to understand **why** your policy fails, and **how** to fix it

Final Project Tips

Why does Steven spend so much time on theory instead of coding?

In supervised learning, follow MNIST tutorial and everything works

This is **NOT** the case in RL

In RL, your code never works on the first try

Even if it is correct, you need to play with hyperparameters

Theory is absolutely necessary to understand **why** your policy fails, and **how** to fix it

You must use your brain to be successful!