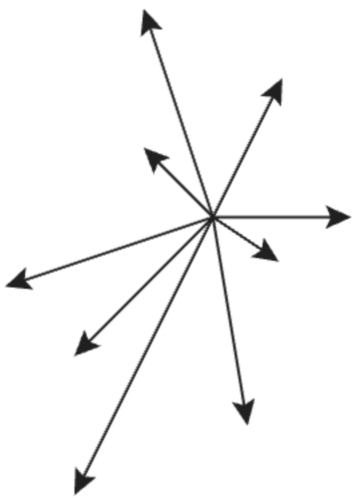
CSCI 1470

Eric Ewing

Wednesday
4/16/25

Deep Learning

Day 31: Actor-Critic and Friends



A vector space, get it?

Goals for Today

1. Review REINFORCE and practice calculations
2. Actor Critic Algorithms
3. PPO (i.e., the RL algorithm used for Chat-GPT)

Goals for Today

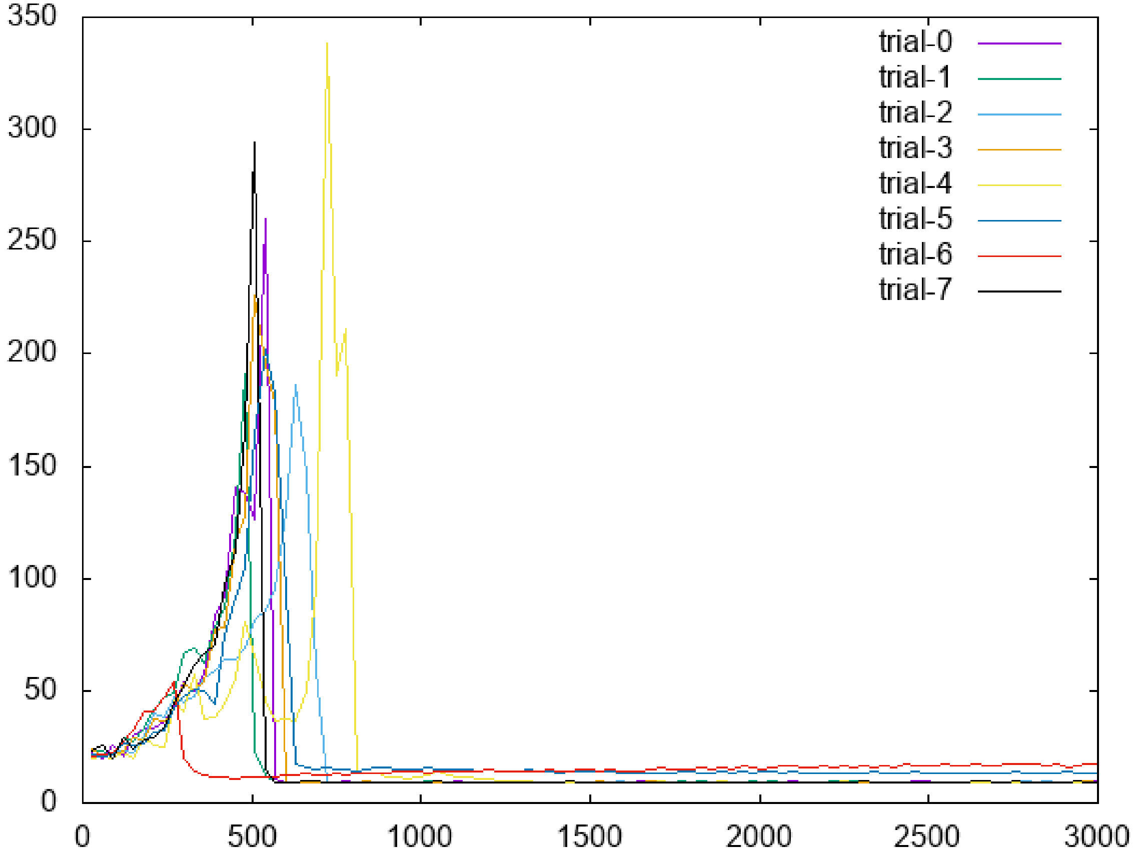
1. Review REINFORCE and practice calculations
2. Actor Critic Algorithms
3. PPO (i.e., the RL algorithm used for Chat-GPT)

Review:

$$J(\theta) = \mathbb{E}[G_0]$$

$$\nabla_\theta J(\theta) = \mathbb{E}[\sum_{t=0}^{T} G_t \nabla_\theta \ln \pi_\theta(a_t|s_t)]$$

Key Idea for today: Variance is the enemy



Cartpole

Policy Collapse: https://stats.stackexchange.com/questions/252685/policy-gradient-reward-collapse

Multi-Arm Bandits

What’s a one-armed bandit?

Multi-Arm Bandits

Multi-Arm Bandits

Single-armed bandit

Multi-Arm Bandits

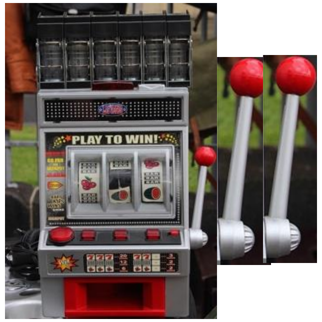
Single-armed bandit

Multi-Arm Bandits

Single-armed bandit

Multi-Arm Bandits

Single-armed bandit

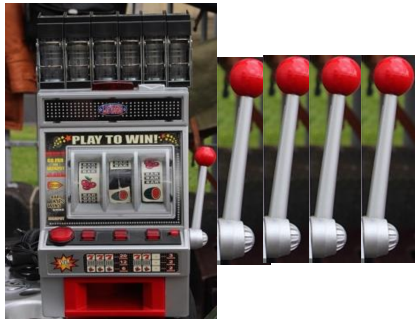


Multi-Arm Bandits

Single-armed bandit

Multi-Arm Bandits

Single-armed bandit

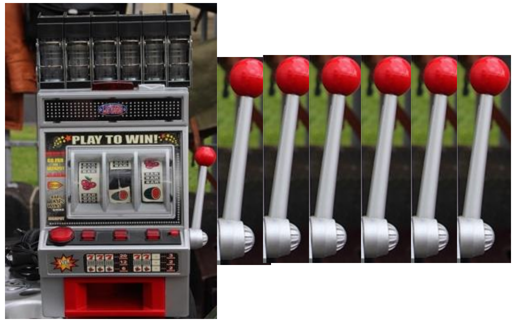


Multi-Arm Bandits

Single-armed bandit

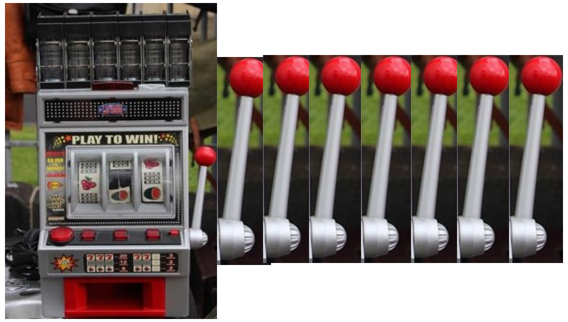
Multi-Arm Bandits

Single-armed bandit



Multi-Arm Bandits

Single-armed bandit

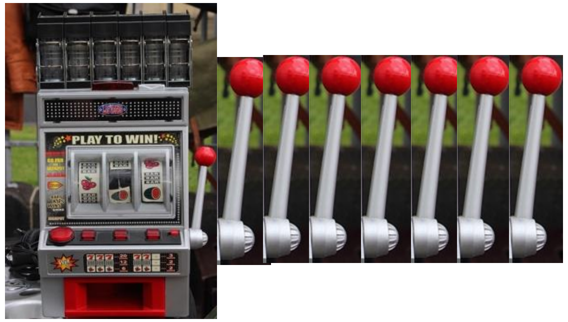


Multi-Arm Bandits

When an arm is pulled, the rewards are random.

Each arm returns a reward with (different) unknown mean and variance

Single-armed bandit



Multi-Arm Bandits

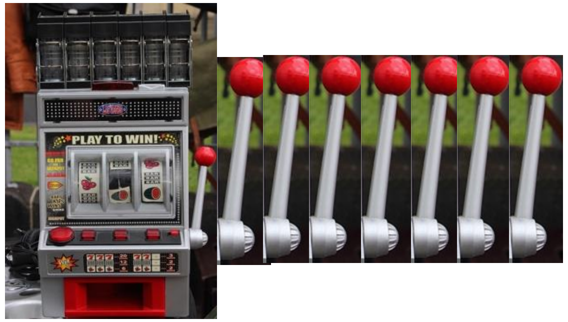
When an arm is pulled, the rewards are random.

Each arm returns a reward with (different) unknown mean and variance

Single-armed bandit

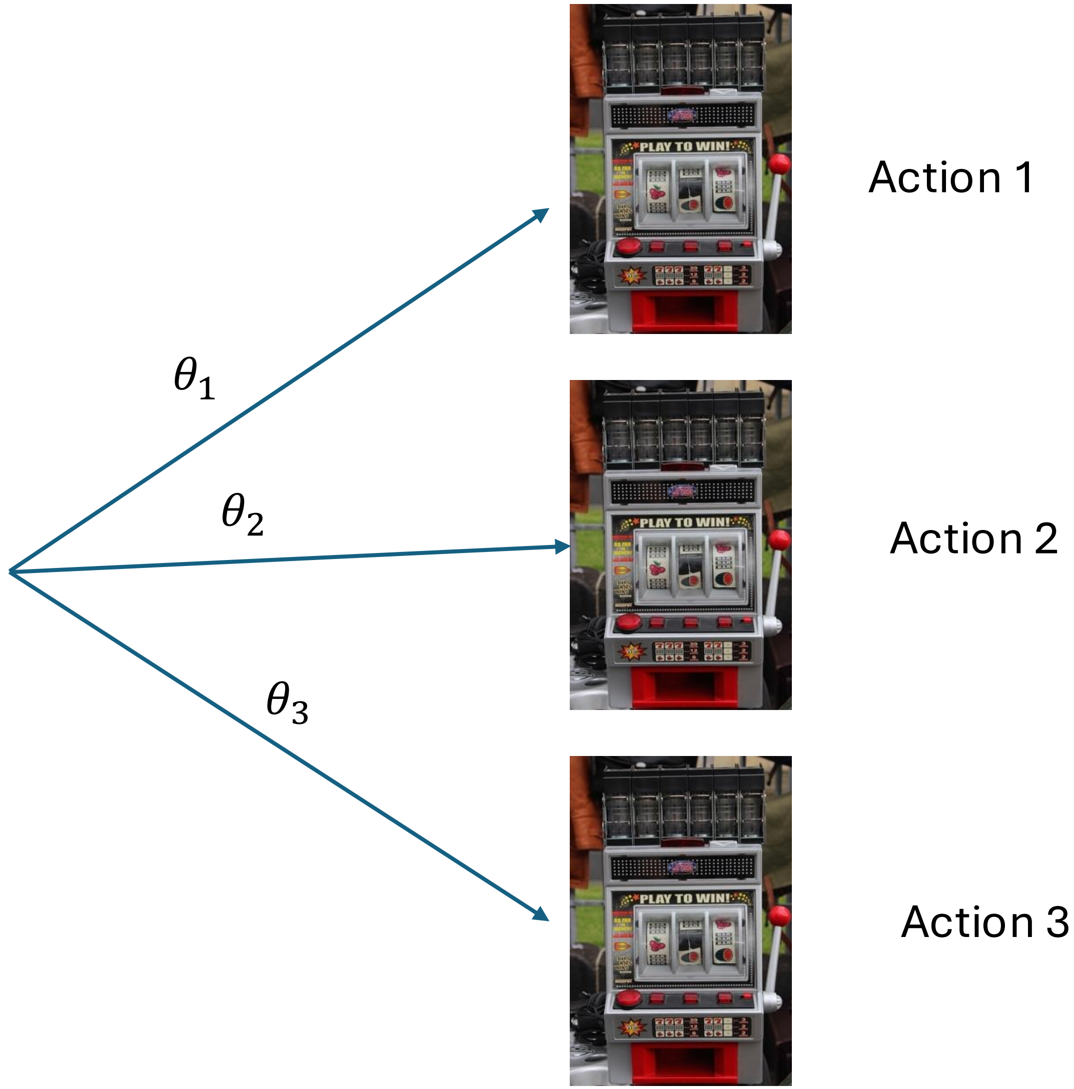
Bandit Problems are essentially MDPs with a single state.

Useful testbed for a number of algorithms and very useful for theory



Policy Gradient on Multi-Arm Bandits

Maintain parameter for each action, θ_i

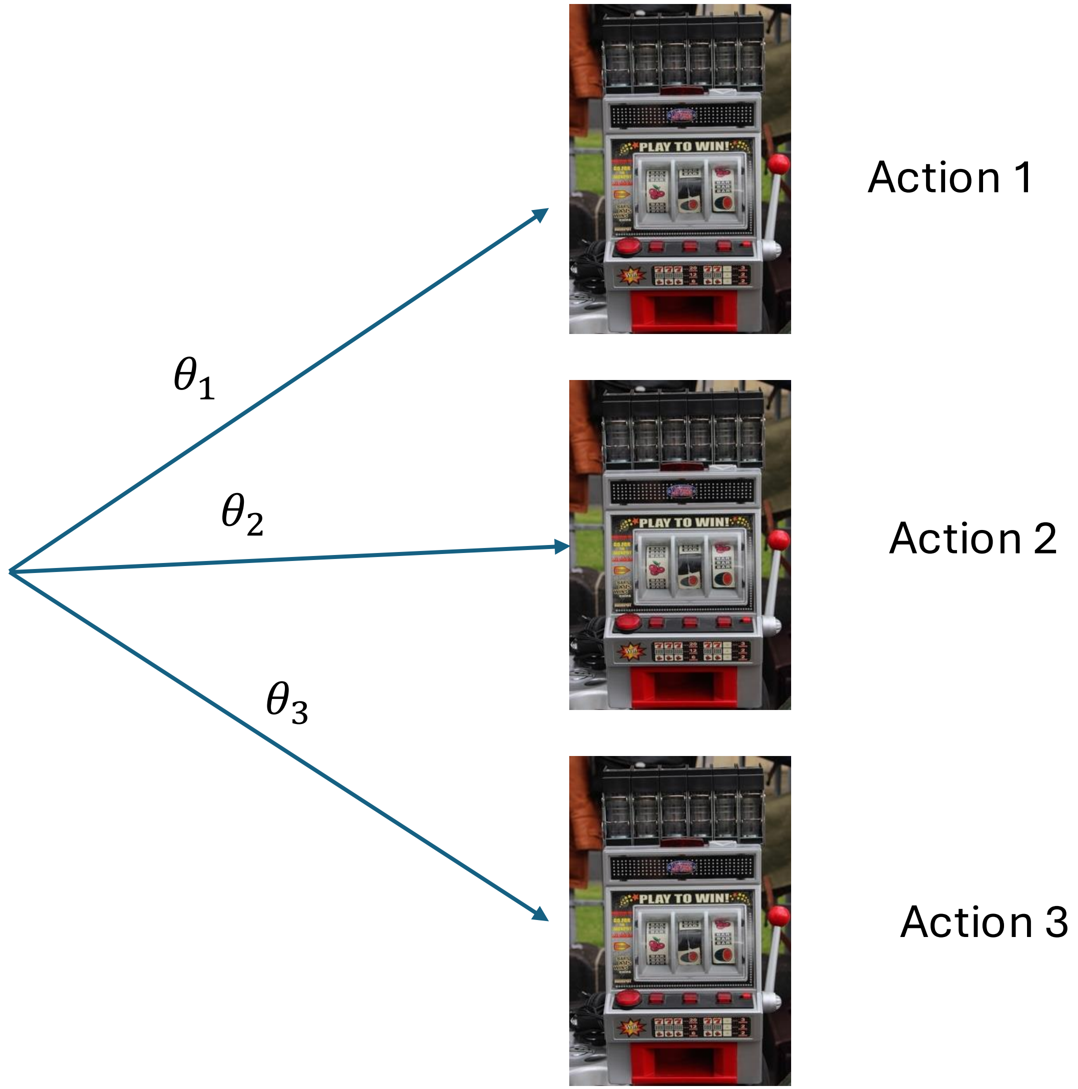


Policy Gradient on Multi-Arm Bandits

Maintain parameter for each action, θ_i

Take Action according to Softmax:

$$\pi_\theta(a_1) = \frac{e^{\theta_1}}{e^{\theta_1} + e^{\theta_2} + e^{\theta_3}}$$

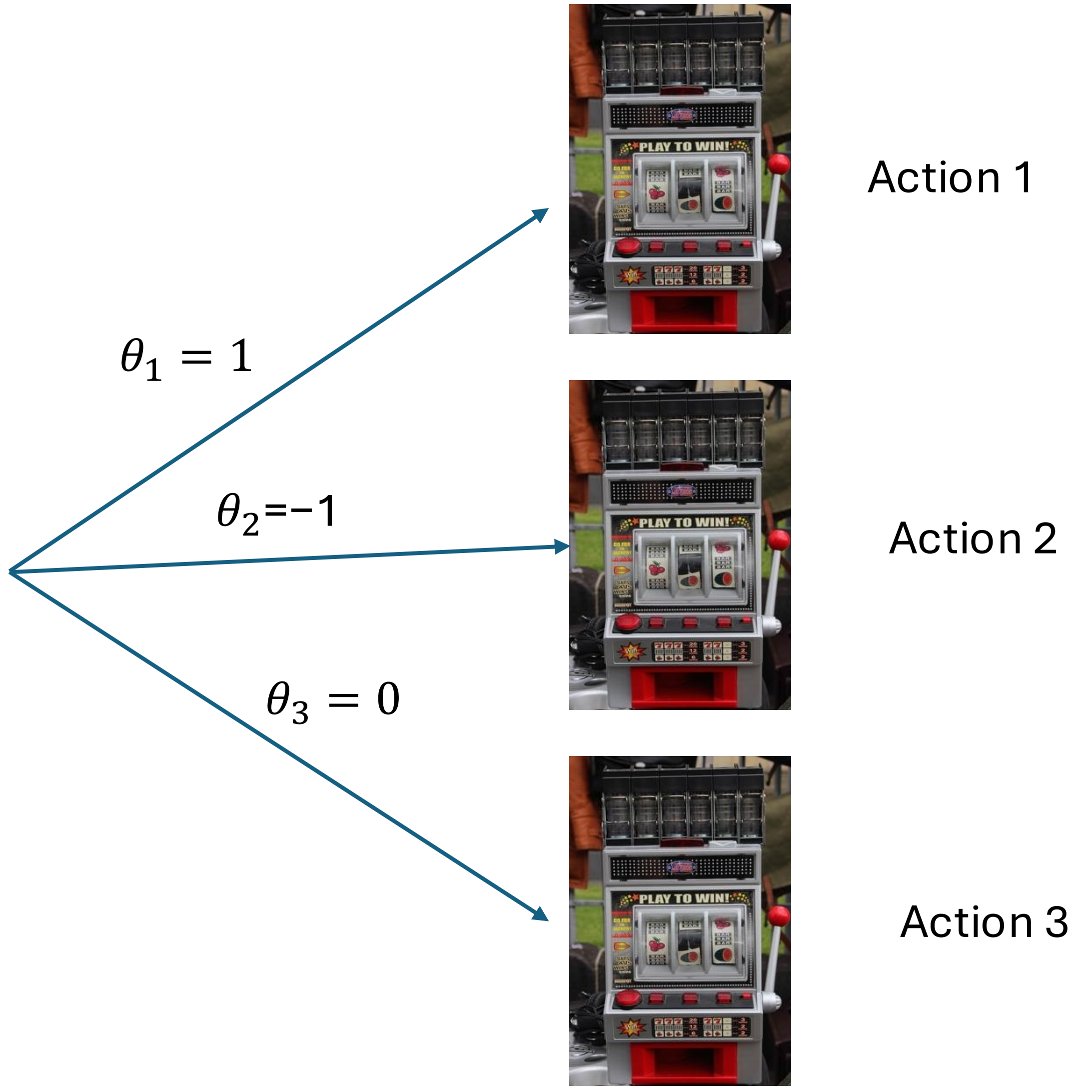


Policy Gradient on Multi-Arm Bandits

Maintain parameter for each action, θ_i

Take Action according to Softmax:

$$\pi_\theta(a_1) = \frac{e^{\theta_1}}{e^{\theta_1} + e^{\theta_2} + e^{\theta_3}}$$



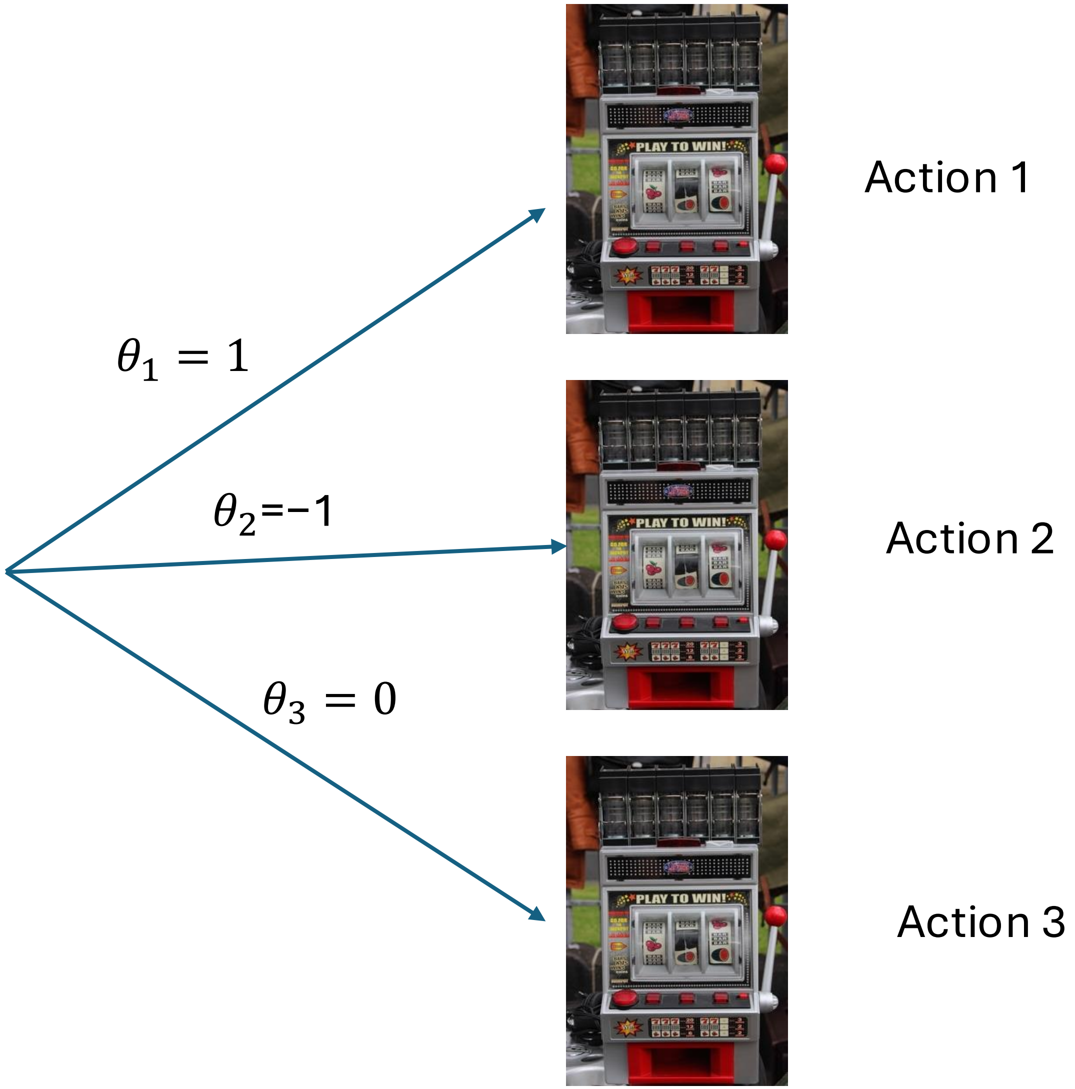
Policy Gradient on Multi-Arm Bandits

Maintain parameter for each action, θ_i

Take Action according to Softmax:

$$\pi_\theta(a_1) = \frac{e^{\theta_1}}{e^{\theta_1} + e^{\theta_2} + e^{\theta_3}}$$

$$\pi_\theta(a_1) = \frac{e}{e + \frac{1}{e} + 1} = 0.66$$



Policy Gradient on Multi-Arm Bandits

$$\theta_1 = 1, \pi_\theta(a_1) = 0.66$$
$$\theta_2 = -1, \pi_\theta(a_2) = 0.09$$
$$\theta_3 = 0, \pi_\theta(a_3) = 0.25$$

T

Policy Gradient on Multi-Arm Bandits

$$\theta_1 = 1, \pi_\theta(a_1) = 0.66$$
$$\theta_2 = -1, \pi_\theta(a_2) = 0.09$$
$$\theta_3 = 0, \pi_\theta(a_3) = 0.25$$

Take 5 actions according to π_θ:

$$\tau = (a_1, 3), (a_2, -1), (a_3, 2), (a_1, 4), (a_3, 1)$$

T

Policy Gradient on Multi-Arm Bandits

$$\theta_1 = 1, \pi_\theta(a_1) = 0.66$$
$$\theta_2 = -1, \pi_\theta(a_2) = 0.09$$
$$\theta_3 = 0, \pi_\theta(a_3) = 0.25$$

Take 1 action according to π_θ:

$$\tau = (a_1, 3)$$

Policy Gradient (without states): $\nabla_\theta J(\theta) = G_t^{T} \nabla_\theta \ln \pi_\theta(a \quad)$

Take Action according to Softmax:

$$\pi_\theta(a_1) = \frac{e^{\theta_1}}{e^{\theta_1} + e^{\theta_2} + e^{\theta_3}}$$

$$\theta_1 = 1, \pi_\theta(a_1) = 0.66$$
$$\theta_2 = -1, \pi_\theta(a_2) = 0.09$$
$$\theta_3 = 0, \pi_\theta(a_3) = 0.25$$

$$\nabla_\theta J(\theta) = \nabla_\theta J(\theta) = G_t \nabla_\theta \ln \pi_\theta(a \quad)$$

$$\tau = (a_1, 3)$$

At timestep 1, a_1 got a reward of 3:

Need to compute: $G_t \nabla_\theta \ln \pi_\theta(a_1)$

Take Action according to Softmax:

$$\pi_\theta(a_1) = \frac{e^{\theta_1}}{e^{\theta_1} + e^{\theta_2} + e^{\theta_3}}$$

$$\theta_1 = 1, \pi_\theta(a_1) = 0.66$$
$$\theta_2 = -1, \pi_\theta(a_2) = 0.09$$
$$\theta_3 = 0, \pi_\theta(a_3) = 0.25$$

$$\nabla_\theta J(\theta) = G_t \nabla_\theta \ln \pi_\theta(a\quad)$$

$$\tau = (a_1, 3)$$

At timestep 1, a_1 got a reward of 3:
Need to compute: $G_t \nabla_\theta \ln \pi_\theta(a_1)$

What is G_0?
Bandits is not a sequential problem (no future rewards to worry about), so it’s just the observed reward for taking the first action, a_1.

Take Action according to Softmax:

$$\pi_\theta(a_1) = \frac{e^{\theta_1}}{e^{\theta_1} + e^{\theta_2} + e^{\theta_3}}$$

$$\theta_1 = 1, \pi_\theta(a_1) = 0.66$$
$$\theta_2 = -1, \pi_\theta(a_2) = 0.09$$
$$\theta_3 = 0, \pi_\theta(a_3) = 0.25$$

$$\nabla_\theta J(\theta) = G_t \nabla_\theta \ln \pi_\theta(a \quad)$$

$$\tau = (a_1, 3)$$

At timestep 1, a_1 got a reward of 3:
Need to compute: $G_t \nabla_\theta \ln \pi_\theta(a_1)$

What is G_0?
Bandits is not a sequential problem (no future rewards to worry about), so it's just the observed reward for taking the first action, a_1.

$$G_t = 3$$

Take Action according to Softmax:

$$\pi_\theta(a_1) = \frac{e^{\theta_1}}{e^{\theta_1} + e^{\theta_2} + e^{\theta_3}}$$

$$\theta_1 = 1, \pi_\theta(a_1) = 0.66$$
$$\theta_2 = -1, \pi_\theta(a_2) = 0.09$$
$$\theta_3 = 0, \pi_\theta(a_3) = 0.25$$

$$\nabla_\theta J(\theta) = G_t \nabla_\theta \ln \pi_\theta(a \quad)$$

$$\tau = (a_1, 3)$$

At timestep 1, a_1 got a reward of 3:
Need to compute: $G_t \nabla_\theta \ln \pi_\theta(a_1)$

What is G_0?
Bandits is not a sequential problem (no future rewards to worry about), so it's just the observed reward for taking the first action, a_1.

$$G_t = 3$$

What is $\nabla_\theta \ln \pi_\theta(a_1)$?

$$\nabla_\theta \ln \pi_\theta(a_1) = \nabla_\theta \ln \frac{e^{\theta_1}}{e^{\theta_1} + e^{\theta_2} + e^{\theta_3}}$$

Take Action according to Softmax:

$$\pi_\theta(a_1) = \frac{e^{\theta_1}}{e^{\theta_1} + e^{\theta_2} + e^{\theta_3}}$$

$$\theta_1 = 1, \pi_\theta(a_1) = 0.66$$
$$\theta_2 = -1, \pi_\theta(a_2) = 0.09$$
$$\theta_3 = 0, \pi_\theta(a_3) = 0.25$$

$$\nabla_\theta J(\theta) = G_t \nabla_\theta \ln \pi_\theta(a \quad)$$

$$\tau = (a_1, 3)$$

At timestep 1, a_1 got a reward of 3:
Need to compute: $G_t \nabla_\theta \ln \pi_\theta(a_1)$

What is G_0?
Bandits is not a sequential problem (no future rewards to worry about), so it's just the observed reward for taking the first action, a_1.

$$G_t = 3$$

What is $\nabla_\theta \ln \pi_\theta(a_1)$?

$$\nabla_\theta \ln \pi_\theta(a_1) = \nabla_\theta \ln \frac{e^{\theta_1}}{e^{\theta_1} + e^{\theta_2} + e^{\theta_3}}$$

$$= \nabla_\theta \left[\ln e^{\theta_1} - \ln\left(e^{\theta_1} + e^{\theta_2} + e^{\theta_3}\right)\right]$$
$$= \nabla_\theta \theta_1 - \nabla_\theta \ln\left(e^{\theta_1} + e^{\theta_2} + e^{\theta_3}\right)$$

Take Action according to Softmax:

$$\pi_\theta(a_1) = \frac{e^{\theta_1}}{e^{\theta_1} + e^{\theta_2} + e^{\theta_3}}$$

$$\theta_1 = 1, \pi_\theta(a_1) = 0.66$$
$$\theta_2 = -1, \pi_\theta(a_2) = 0.09$$
$$\theta_3 = 0, \pi_\theta(a_3) = 0.25$$

$$\nabla_\theta J(\theta) = G_t \nabla_\theta \ln \pi_\theta(a \quad)$$

$$\tau = (a_1, 3)$$

At timestep 1, a_1 got a reward of 3:
Need to compute: $G_t \nabla_\theta \ln \pi_\theta(a_1)$

What is G_0?
Bandits is not a sequential problem (no future rewards to worry about), so it's just the observed reward for taking the first action, a_1.

$$G_t = 3$$

What is the shape of $\nabla_\theta J(\theta)$?

What is $\nabla_\theta \ln \pi_\theta(a_1)$?

$$\nabla_\theta \ln \pi_\theta(a_1) = \nabla_\theta \ln \frac{e^{\theta_1}}{e^{\theta_1} + e^{\theta_2} + e^{\theta_3}}$$

$$= \nabla_\theta \left[\ln e^{\theta_1} - \ln\left(e^{\theta_1} + e^{\theta_2} + e^{\theta_3}\right)\right]$$
$$= \nabla_\theta \theta_1 - \nabla_\theta \ln\left(e^{\theta_1} + e^{\theta_2} + e^{\theta_3}\right)$$

Take Action according to Softmax:

$$\pi_\theta(a_1) = \frac{e^{\theta_1}}{e^{\theta_1} + e^{\theta_2} + e^{\theta_3}}$$

$$\theta_1 = 1, \pi_\theta(a_1) = 0.66$$
$$\theta_2 = -1, \pi_\theta(a_2) = 0.09$$
$$\theta_3 = 0, \pi_\theta(a_3) = 0.25$$

$$\nabla_\theta J(\theta) = G_t \nabla_\theta \ln \pi_\theta(a \quad)$$

$$\tau = (a_1, 3)$$

At timestep 1, a_1 got a reward of 3:
Need to compute: $G_t \nabla_\theta \ln \pi_\theta(a_1)$

What is G_0?
Bandits is not a sequential problem (no future rewards to worry about), so it's just the observed reward for taking the first action, a_1.

$$G_t = 3$$

What is the shape of $\nabla_\theta J(\theta)$?

What is $\nabla_\theta \ln \pi_\theta(a_1)$?

$$\nabla_\theta \ln \pi_\theta(a_1) = \nabla_\theta \ln \frac{e^{\theta_1}}{e^{\theta_1} + e^{\theta_2} + e^{\theta_3}}$$

$$= \nabla_\theta \left[\ln e^{\theta_1} - \ln\left(e^{\theta_1} + e^{\theta_2} + e^{\theta_3}\right)\right]$$
$$= \nabla_\theta \theta_1 - \nabla_\theta \ln\left(e^{\theta_1} + e^{\theta_2} + e^{\theta_3}\right)$$

$$= \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} \frac{e^{\theta_1}}{e^{\theta_1} + e^{\theta_2} + e^{\theta_3}} \\ \cdots \\ \cdots \end{bmatrix}$$

Take Action according to Softmax:

$$\pi_\theta(a_1) = \frac{e^{\theta_1}}{e^{\theta_1} + e^{\theta_2} + e^{\theta_3}}$$

$$\theta_1 = 1, \pi_\theta(a_1) = 0.66$$
$$\theta_2 = -1, \pi_\theta(a_2) = 0.09$$
$$\theta_3 = 0, \pi_\theta(a_3) = 0.25$$

$$\nabla_\theta J(\theta) = G_t \nabla_\theta \ln \pi_\theta(a \quad)$$

$$\tau = (a_1, 3)$$

At timestep 1, a_1 got a reward of 3:
Need to compute: $G_t \nabla_\theta \ln \pi_\theta(a_1)$

$$G_t \nabla_\theta \ln \pi_\theta(a_1) = 3 \cdot \begin{bmatrix} 0.34 \\ -0.09 \\ -0.25 \end{bmatrix}$$

What is G_0?
Bandits is not a sequential problem (no future rewards to worry about), so it's just the observed reward for taking the first action, a_1.

$$G_t = 3$$

What is the shape of $\nabla_\theta J(\theta)$?

What is $\nabla_\theta \ln \pi_\theta(a_1)$?

$$\nabla_\theta \ln \pi_\theta(a_1) = \nabla_\theta \ln \frac{e^{\theta_1}}{e^{\theta_1} + e^{\theta_2} + e^{\theta_3}}$$

$$= \nabla_\theta \left[\ln e^{\theta_1} - \ln\left(e^{\theta_1} + e^{\theta_2} + e^{\theta_3}\right)\right]$$
$$= \nabla_\theta \theta_1 - \nabla_\theta \ln\left(e^{\theta_1} + e^{\theta_2} + e^{\theta_3}\right)$$

$$= \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} \frac{e^{\theta_1}}{e^{\theta_1} + e^{\theta_2} + e^{\theta_3}} \\ \cdots \\ \cdots \end{bmatrix}$$

RL Conceptual Question Hints

RL Conceptual Question Hints

- For each trajectory, compute $G_t \nabla_\theta \ln \pi_\theta(a^{(t)})$ for each timestep and sum them together.

RL Conceptual Question Hints

- For each trajectory, compute $G_t \nabla_\theta \ln \pi_\theta(a^{(t)})$ for each timestep and sum them together.
- There will be variance between the gradient estimates of different trajectories (i.e., $\nabla_\theta J(\theta)$ will be different for each trajectory)

RL Conceptual Question Hints

- For each trajectory, compute $G_t \nabla_\theta \ln \pi_\theta(a^{(t)})$ for each timestep and sum them together.
- There will be variance between the gradient estimates of different trajectories (i.e., $\nabla_\theta J(\theta)$ will be different for each trajectory)
- Compute mean variance between same elements of different gradients.

RL Conceptual Question Hints

- For each trajectory, compute $G_t \nabla_\theta \ln \pi_\theta(a^{(t)})$ for each timestep and sum them together.
- There will be variance between the gradient estimates of different trajectories (i.e., $\nabla_\theta J(\theta)$ will be different for each trajectory)
- Compute mean variance between same elements of different gradients.
- Value of each state is average discounted returns from that state

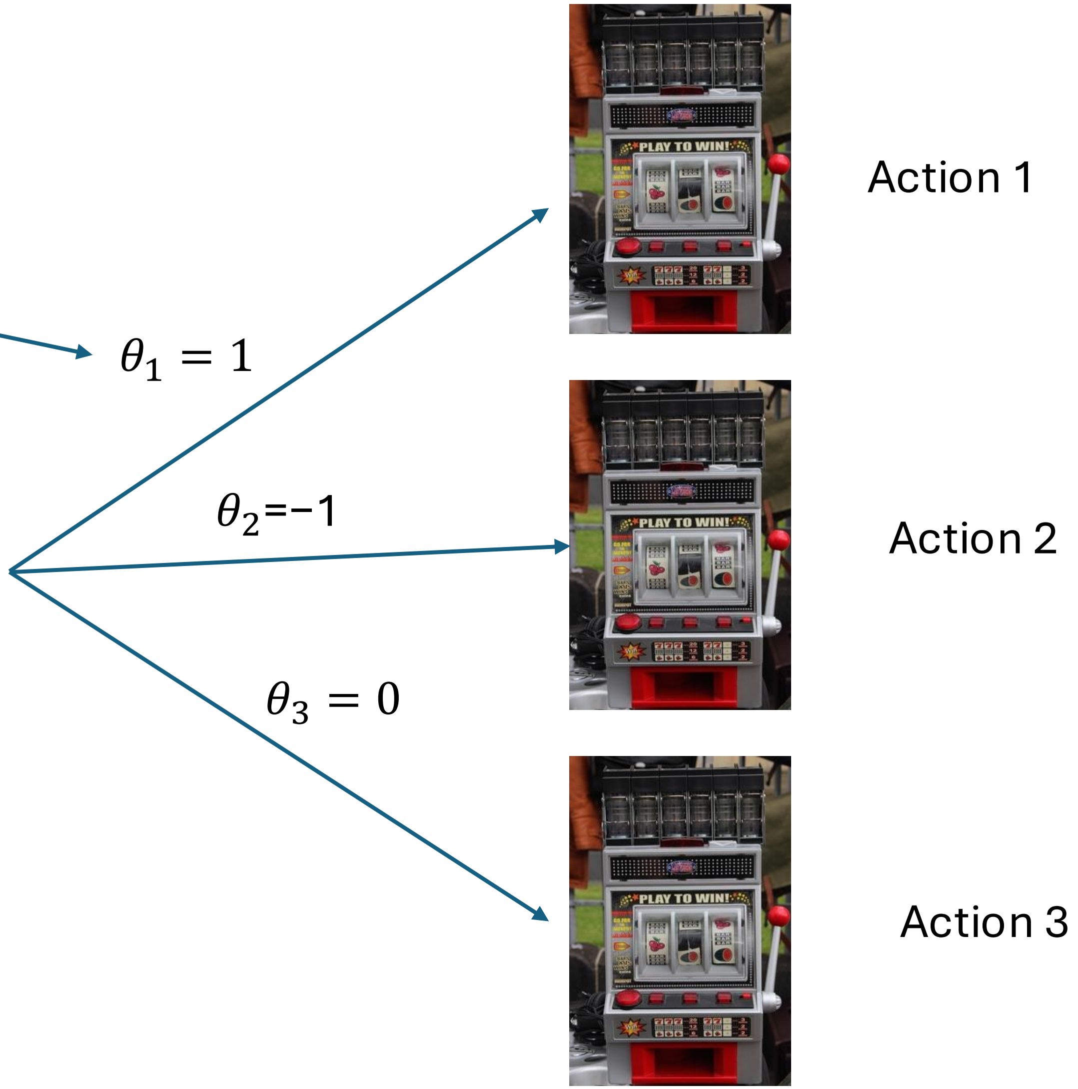
RL Conceptual Question Hints

- For each trajectory, compute $G_t \nabla_\theta \ln \pi_\theta(a^{(t)})$ for each timestep and sum them together.
- There will be variance between the gradient estimates of different trajectories (i.e., $\nabla_\theta J(\theta)$ will be different for each trajectory)
- Compute mean variance between same elements of different gradients.
- Value of each state is average discounted returns from that state
- Key Point: What happens to mean and variance of gradients when baseline function (*V*) is added?

REINFORCE

What if θ is the output of a neural network?

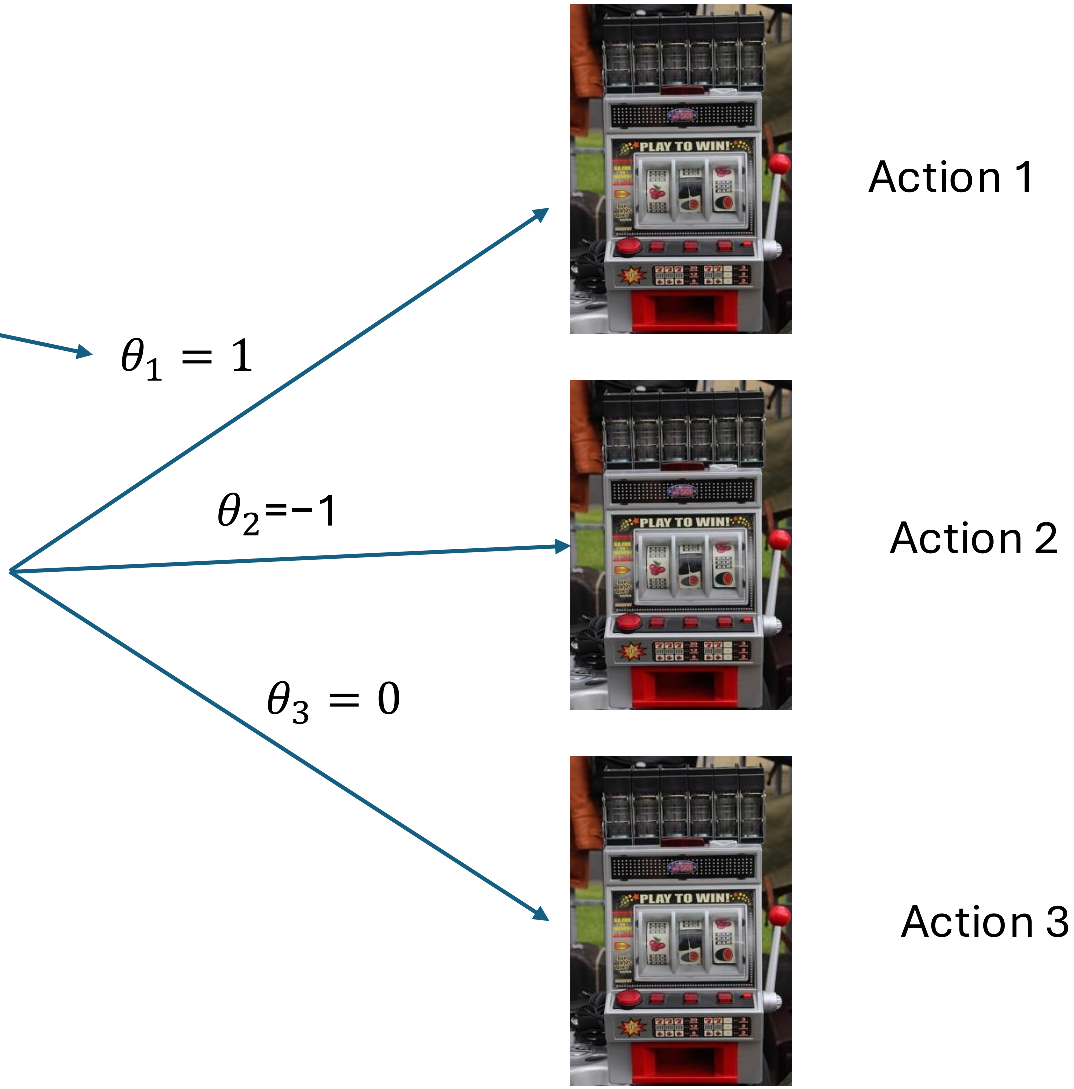
How to compute: $G_t \nabla_\theta \ln \pi_\theta(a_t|s_t)$?



REINFORCE

What if θ is the output of a neural network?

How to compute: $G_t \nabla_\theta \ln \pi_\theta(a_t|s_t)$?



Compute $\ln \pi_\theta(a_t|s_t)$ in gradient tape context.

REINFORCE

What if θ is the output of a neural network?

How to compute: $G_t \nabla_\theta \ln \pi_\theta(a_t|s_t)$?

$\theta_1 = 1$

θ_2=−1

$\theta_3 = 0$

Action 1

Action 2

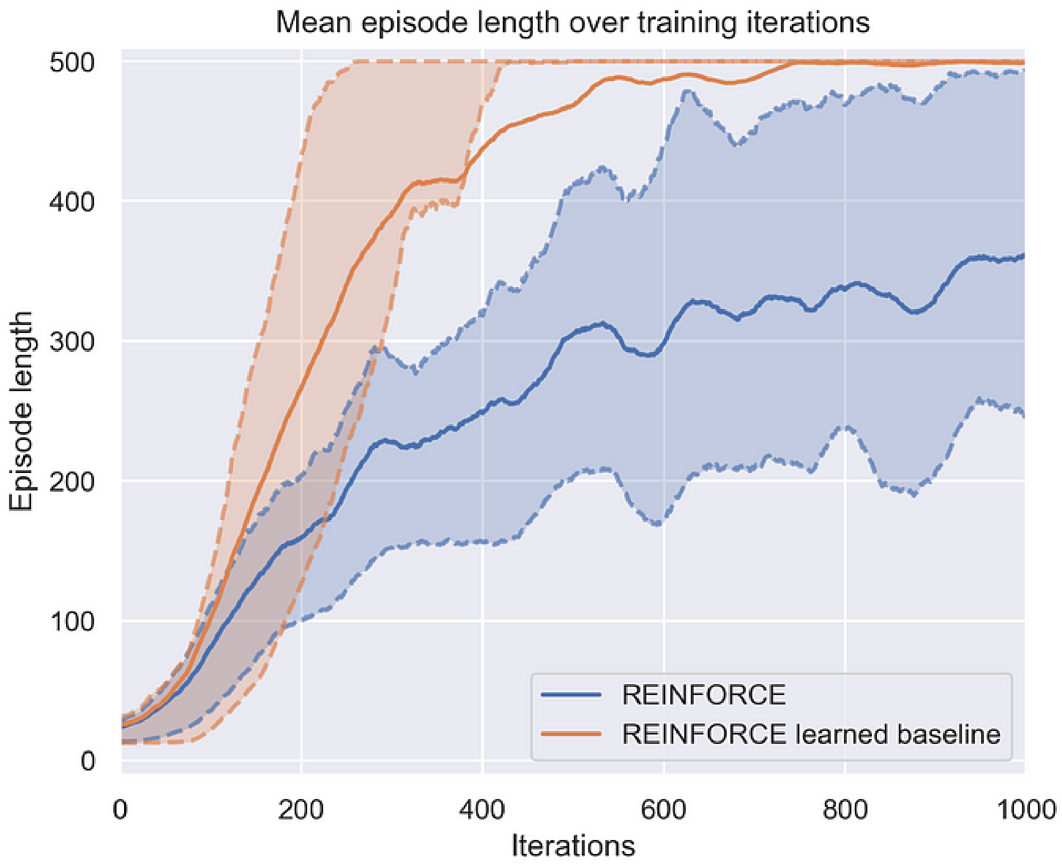
Action 3

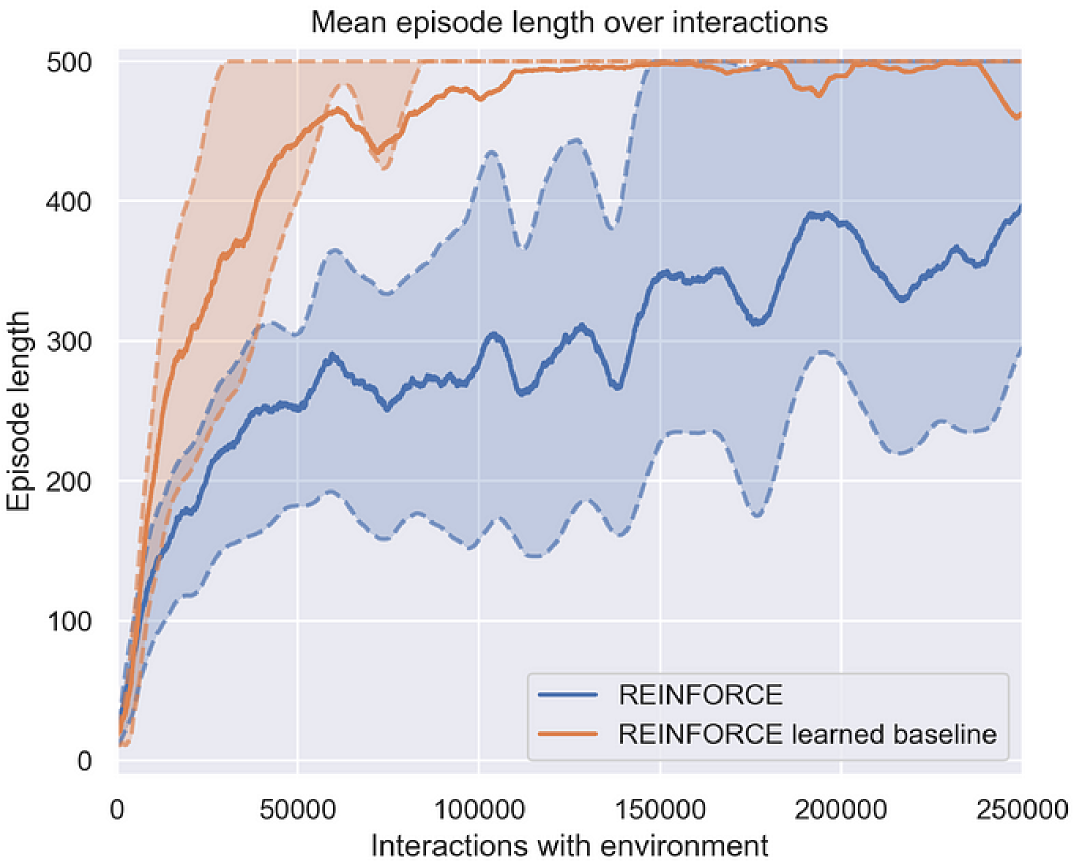
Compute $\ln \pi_\theta(a_t|s_t)$ in gradient tape context.

But also, remember, you have to perform gradient ASCENT.
If an optimizer minimizes by default, you can use $-\nabla_\theta J(\theta)$

REINFORCE Variance

If we could calculate $\nabla_\theta J(\theta)$ exactly (not just for single trajectory/sample), then Policy Gradient would be a great algorithm! (with some minor flaws)





Results on Cartpole

Image Source: https://medium.com/@fork.tree.ai/understanding-baseline-techniques-for-reinforce-53a1e2279b57

Actor-Critic Methods

Actor-Critic Methods

REINFORCE uses the return for a trajectory G_t:

Actor-Critic Methods

REINFORCE uses the return for a trajectory G_t:

$$\nabla_\theta J(\theta) = \mathbb{E}[\sum_{t=0}^{T} G_t \nabla_\theta \ln \pi_\theta(a_t|s_t)]$$

Actor-Critic Methods

REINFORCE uses the return for a trajectory G_t:

$$\nabla_\theta J(\theta) = \mathbb{E}[\sum_{t=0}^{T} G_t \nabla_\theta \ln \pi_\theta(a_t|s_t)]$$

Variance of Returns is always a problem…

Actor-Critic Methods

REINFORCE uses the return for a trajectory G_t:

$$\nabla_\theta J(\theta) = \mathbb{E}[\sum_{t=0}^{T} G_t \nabla_\theta \ln \pi_\theta(a_t|s_t)]$$

Variance of Returns is always a problem...

Actor-Critic Methods learn an approximation of G_t

Actor-Critic Methods

REINFORCE uses the return for a trajectory G_t:

$$\nabla_\theta J(\theta) = \mathbb{E}[\sum_{t=0}^{T} G_t \nabla_\theta \ln \pi_\theta(a_t|s_t)]$$

Actor-Critic Methods learn an approximation of G_t

$$Q^\pi(s_t, a_t) = \mathbb{E}[G_t]$$

Variance of Returns is always a problem…

Actor-Critic Methods

REINFORCE uses the return for a trajectory G_t:

$$\nabla_\theta J(\theta) = \mathbb{E}[\sum_{t=0}^{T} G_t \nabla_\theta \ln \pi_\theta(a_t|s_t)]$$

Actor-Critic Methods learn an approximation of G_t

$$Q^\pi(s_t, a_t) = \mathbb{E}[G_t]$$

Variance of Returns is always a problem…

$V(s_t) = \mathbb{E}[G_t]$ as well, but technically it's not a critic function. Critic functions critique actions.

Actor-Critic Methods

REINFORCE uses the return for a trajectory G_t:

$$\nabla_\theta J(\theta) = \mathbb{E}[\sum_{t=0}^{T} G_t \nabla_\theta \ln \pi_\theta(a_t|s_t)]$$

Variance of Returns is always a problem…

Actor-Critic Methods learn an approximation of G_t

$$Q^\pi(s_t, a_t) = \mathbb{E}[G_t]$$

$V(s_t) = \mathbb{E}[G_t]$ as well, but technically it's not a critic function. Critic functions critique actions.

$$\nabla_\theta J(\theta) = \mathbb{E}[\sum_{t=0}^{T} Q^{\pi_\theta}(s_t, a_t) \nabla_\theta \ln \pi_\theta(a_t|s_t)]$$

Actor-Critic Algorithm: Learn Q^{π} and $\pi(a|s)$

Actor-Critic Algorithm: Learn Q^{π} and $\pi(a|s)$

Actor (policy): Takes actions

Actor-Critic Algorithm: Learn Q^{π} and $\pi(a|s)$

Actor (policy): Takes actions

Critic: Scores the action

Actor-Critic Algorithm: Learn Q^{π} and $\pi(a|s)$

Actor (policy): Takes actions

Critic: Scores the action

Actor-Critic Algorithm: Learn Q^{π} and $\pi(a|s)$

Policy network has parameters θ
Q network has parameters w

Initialize $\pi_\theta, Q_w, \alpha_\theta, \alpha_w$

Repeat forever:

Take action a, get new state s' and reward r

Sample next action $a' \sim \pi_\theta(a|s)$

update $\theta \leftarrow \theta + \alpha_\theta Q_w(s,a) \nabla_\theta \ln \pi_\theta(a|s)$

Calculate TD Error: $\delta = r + \gamma Q_w(s', a') - Q_w(s,a)$

update $w \leftarrow w + \alpha_w \delta \nabla_w Q_w(s,a)$

$a \leftarrow a', s \leftarrow s'$

Actor-Critic Algorithm: Learn Q^{π} and $\pi(a|s)$

Policy network has parameters θ
Q network has parameters w

Initialize $\pi_\theta, Q_w, \alpha_\theta, \alpha_w$

Repeat forever:

Take action a, get new state s' and reward r

Sample next action $a' \sim \pi_\theta(a|s)$

update $\theta \leftarrow \theta + \alpha_\theta Q_w(s,a) \nabla_\theta \ln \pi_\theta(a|s)$

Calculate TD Error: $\delta = r + \gamma Q_w(s', a') - Q_w(s,a)$

update $w \leftarrow w + \alpha_w \delta \nabla_w Q_w(s,a)$

$a \leftarrow a', s \leftarrow s'$

Like Q-learning and REINFORCE at the same time

Variations on a Theme…

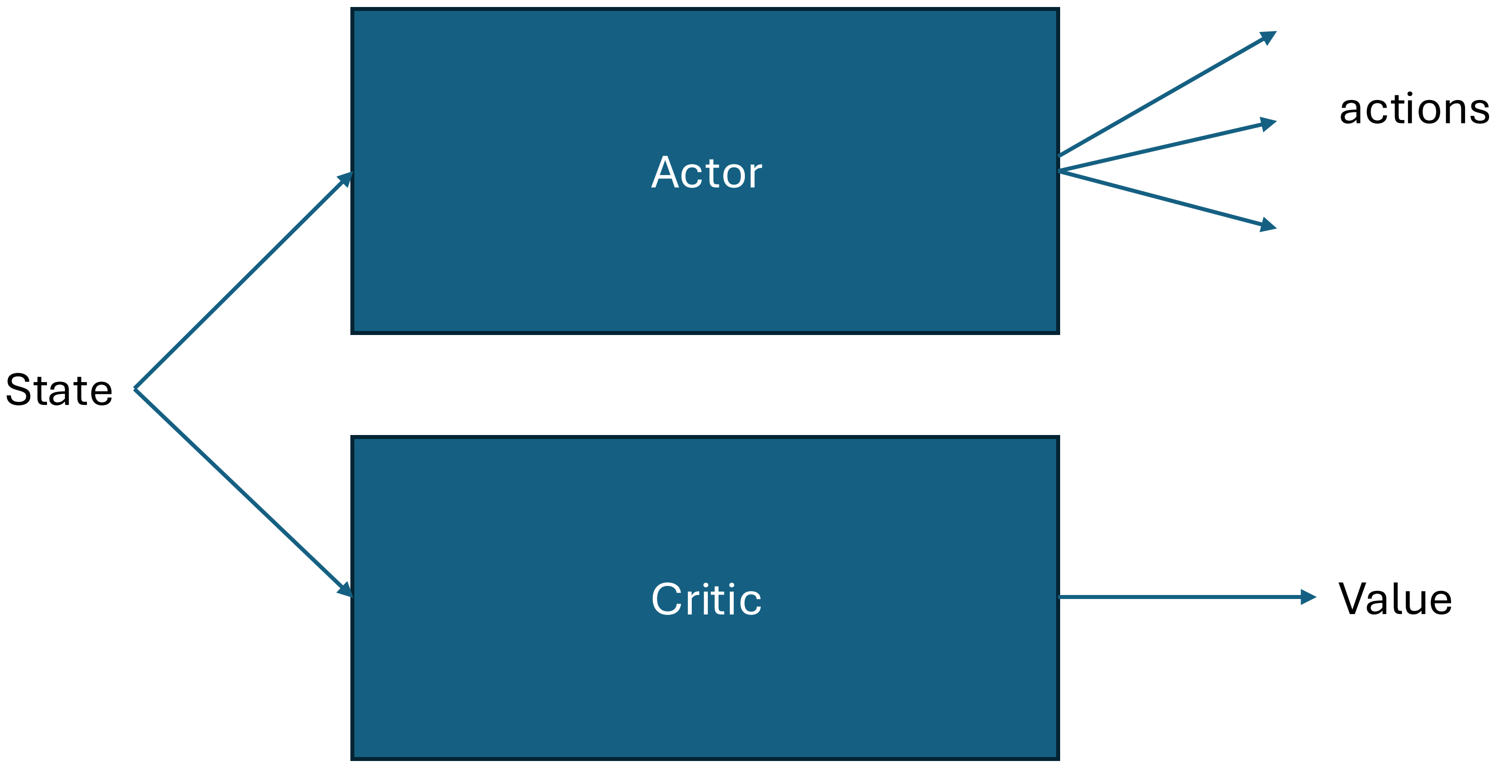
How to estimate $J(\theta)$

(Wikipedia uses R_t instead of G_t)

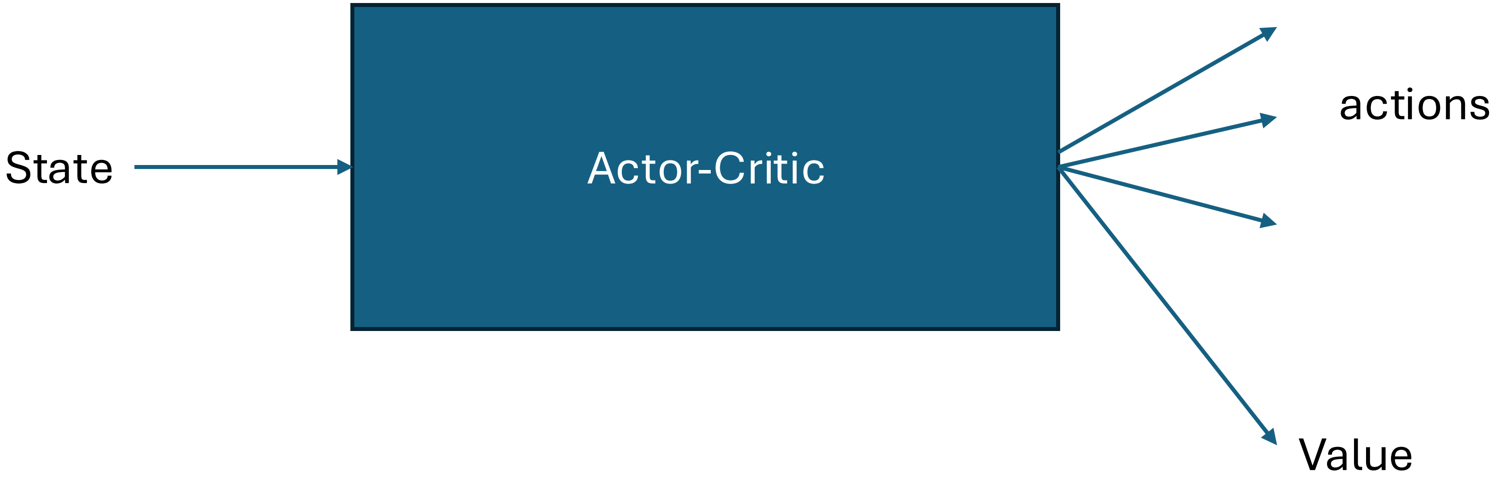
- $\sum_{0 \le i \le T} (\gamma^i R_i)$.
- $\gamma^j \sum_{j \le i \le T} (\gamma^{i-j} R_i)$: the **REINFORCE** algorithm.
- $\gamma^j \sum_{j \le i \le T} (\gamma^{i-j} R_i) - b(S_j)$: the **REINFORCE with baseline** algorithm. Here b is an arbitrary function.
- $\gamma^j \left(R_j + \gamma V^{\pi_\theta}(S_{j+1}) - V^{\pi_\theta}(S_j)\right)$: TD(1) learning.
- $\gamma^j Q^{\pi_\theta}(S_j, A_j)$.
- $\gamma^j A^{\pi_\theta}(S_j, A_j)$: **Advantage Actor-Critic (A2C)**.[3]
- $\gamma^j \left(R_j + \gamma R_{j+1} + \gamma^2 V^{\pi_\theta}(S_{j+2}) - V^{\pi_\theta}(S_j)\right)$: TD(2) learning.
- $\gamma^j \left(\sum_{k=0}^{n-1} \gamma^k R_{j+k} + \gamma^n V^{\pi_\theta}(S_{j+n}) - V^{\pi_\theta}(S_j)\right)$: TD(n) learning.
- $\gamma^j \sum_{n=1}^{\infty} \frac{\lambda^{n-1}}{1-\lambda} \cdot \left(\sum_{k=0}^{n-1} \gamma^k R_{j+k} + \gamma^n V^{\pi_\theta}(S_{j+n}) - V^{\pi_\theta}(S_j)\right)$: TD(λ) learning, also known as **GAE (generalized advantage estimate)**.[4] This is obtained by an exponentially decaying sum of the TD(n) learning terms.

Source: https://en.wikipedia.org/wiki/Actor-critic_algorithm

Actor-Critic Networks



Actor-Critic Networks



Just make sure you use the correct activation function for the different outputs

Deep Q-Learning Revisited

Compute TD-Error: $\delta = r + \gamma \max_{a'} Q(s', a') - Q(s, a)$

Loss Function: $L = \delta^2$

Update model with Gradient Descent

Deep Q-Learning Revisited

Compute TD-Error: $\delta = r + \gamma \max_{a'} Q(s', a') - Q(s, a)$

Loss Function: $L = \delta^2$

Update model with Gradient Descent

Q-Learning uses:

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

Actor-Critic Uses:

$$\delta = r + \gamma Q(s', a') - Q(s, a)$$

Deep Q-Learning Revisited

Compute TD-Error: $\delta = r + \gamma \max_{a'} Q(s', a') - Q(s, a)$

Loss Function: $L = \delta^2$

Update model with Gradient Descent

Q-Learning uses:

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

Actor-Critic Uses:

$$\delta = r + \gamma Q(s', a') \ - Q\ (s, a)$$

Q-Learning is learning *Optimal Q-values*

Actor-Critic is learning the Q-values for following a specific policy Q^{π}

On-Policy Vs. Off-Policy Learning

RL algorithms collect experiences and learn from these experiences

On-Policy Algorithms have to collect experiences with the policy they are learning

Off-Policy Algorithms can use **any** policy to collect experiences

DQNs Are Off-Policy

In Q-Learning, we typically collect experiences using an ϵ-greedy policy in training

With probability ϵ, take random action.

Else, take action $\mathrm{argmax}_a\ Q(s, a)$

At test time, we should take the best actions, not the ϵ-greedy actions

$$\mathrm{argmax}_a\ Q(s, a)$$

DQNs Are Off-Policy

In Q-Learning, we typically collect experiences using an ϵ-greedy policy in training

With probability ϵ, take random action.

Else, take action $\text{argmax}_a\, Q(s, a)$

At test time, we should take the best actions, not the ϵ-greedy actions

$$\text{argmax}_a\, Q(s, a)$$

These are different policies! DQNs can be trained with any data collection policy at training time

Actor-Critic Algorithm: Learn Q^{π} and $\pi(a|s)$

Initialize $\pi_{\theta}, Q_w, \alpha_{\theta}, \alpha_w$

Repeat forever:

Take action a, get new state s' and reward r

Sample next action $a' \sim \pi_{\theta}(a|s)$

On Policy: Have to take actions according to π_{θ}

update $\theta \leftarrow \theta + \alpha_{\theta} Q_w(s,a) \nabla_{\theta} \ln \pi_{\theta}(a|s)$

Calculate TD Error: $\delta = r + \gamma Q_w(s',a') - Q_w(s,a)$

update $w \leftarrow w + \alpha_w \delta \nabla_w Q_w(s,a)$

$a \leftarrow a', s \leftarrow s'$

On-Policy vs Off-Policy Learning

On-Policy vs Off-Policy Learning

Advantage of On-Policy Learning:

On-Policy vs Off-Policy Learning

Advantage of On-Policy Learning:

- More sample efficient, tend to converge faster

On-Policy vs Off-Policy Learning

Advantage of On-Policy Learning:

- More sample efficient, tend to converge faster

Disadvantages of On-Policy Learning:

On-Policy vs Off-Policy Learning

Advantage of On-Policy Learning:

- More sample efficient, tend to converge faster

Disadvantages of On-Policy Learning:

- Can get stuck in local minima
  - Is there a simple policy that performs ok? Would small changes to that policy cause returns to go down temporarily?
  - How do balance exploration in our policy?

On-Policy vs Off-Policy Learning

Advantage of On-Policy Learning:

- More sample efficient, tend to converge faster

Disadvantages of On-Policy Learning:

- Can get stuck in local minima
  - Is there a simple policy that performs ok? Would small changes to that policy cause returns to go down temporarily?
  - How do balance exploration in our policy?

Advantages of Off-Policy Learning:

On-Policy vs Off-Policy Learning

Advantage of On-Policy Learning:

- More sample efficient, tend to converge faster

Disadvantages of On-Policy Learning:

- Can get stuck in local minima
  - Is there a simple policy that performs ok? Would small changes to that policy cause returns to go down temporarily?
  - How do balance exploration in our policy?

Advantages of Off-Policy Learning:

- Can learn from any policy

On-Policy vs Off-Policy Learning

Advantage of On-Policy Learning:

- More sample efficient, tend to converge faster

Disadvantages of On-Policy Learning:

- Can get stuck in local minima
    - Is there a simple policy that performs ok? Would small changes to that policy cause returns to go down temporarily?
    - How do balance exploration in our policy?

Advantages of Off-Policy Learning:

- Can learn from any policy
- More likely to learn an optimal policy

On-Policy vs Off-Policy Learning

Advantage of On-Policy Learning:

- More sample efficient, tend to converge faster

Disadvantages of On-Policy Learning:

- Can get stuck in local minima
  - Is there a simple policy that performs ok? Would small changes to that policy cause returns to go down temporarily?
  - How do balance exploration in our policy?

Advantages of Off-Policy Learning:

- Can learn from any policy
- More likely to learn an optimal policy

Disadvantages of Off-Policy Learning:

On-Policy vs Off-Policy Learning

Advantage of On-Policy Learning:

- More sample efficient, tend to converge faster

Disadvantages of On-Policy Learning:

- Can get stuck in local minima
  - Is there a simple policy that performs ok? Would small changes to that policy cause returns to go down temporarily?
  - How do balance exploration in our policy?

Advantages of Off-Policy Learning:

- Can learn from any policy
- More likely to learn an optimal policy

Disadvantages of Off-Policy Learning:

- Slower…

For the Record: On-Policy Q-Learning (SARSA)

There is an On-Policy Q-learning algorithm:

$$\delta = r + \gamma Q(s', a') \; - Q(s, a)$$

Why is it called SARSA?

$$\delta = \gamma Q(s', a') + r \; - Q \; (s, a)$$

Off-Policy Learning

Most of the time in RL, collecting the data is computationally expensive.

So far, we've looked at an example, learned from it, and discarded it.

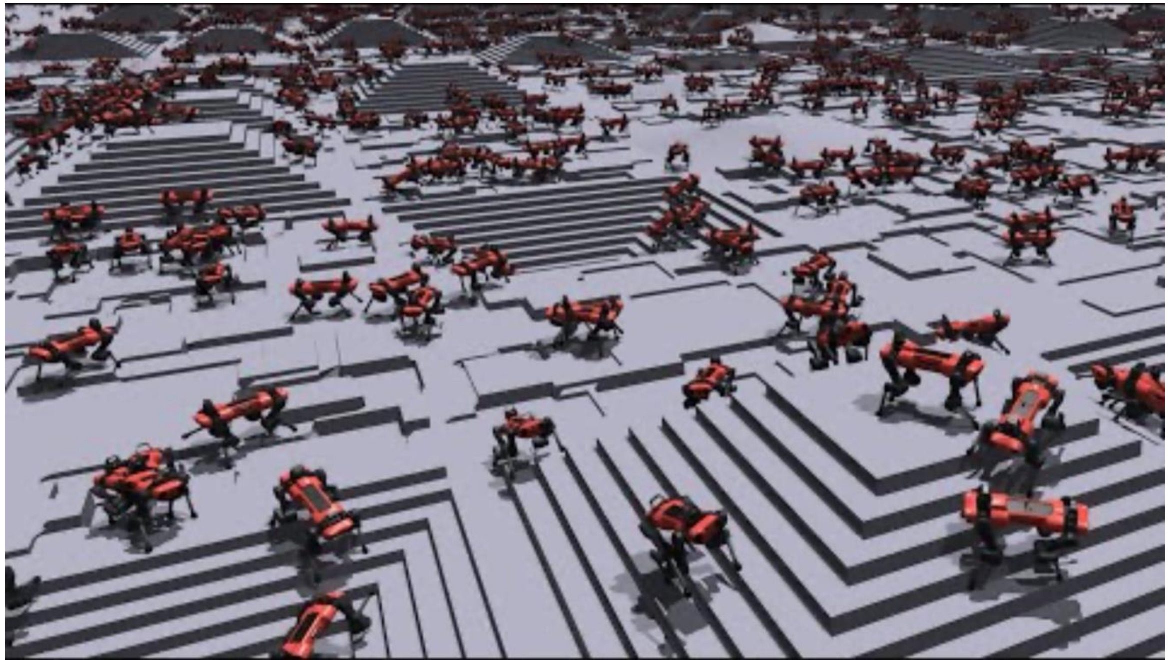
In all our other problems, we always learned from data multiple times (i.e., epochs)

Off-Policy Learning

Most of the time in RL, collecting the data is computationally expensive.

So far, we've looked at an example, learned from it, and discarded it.

In all our other problems, we always learned from data multiple times (i.e., epochs)



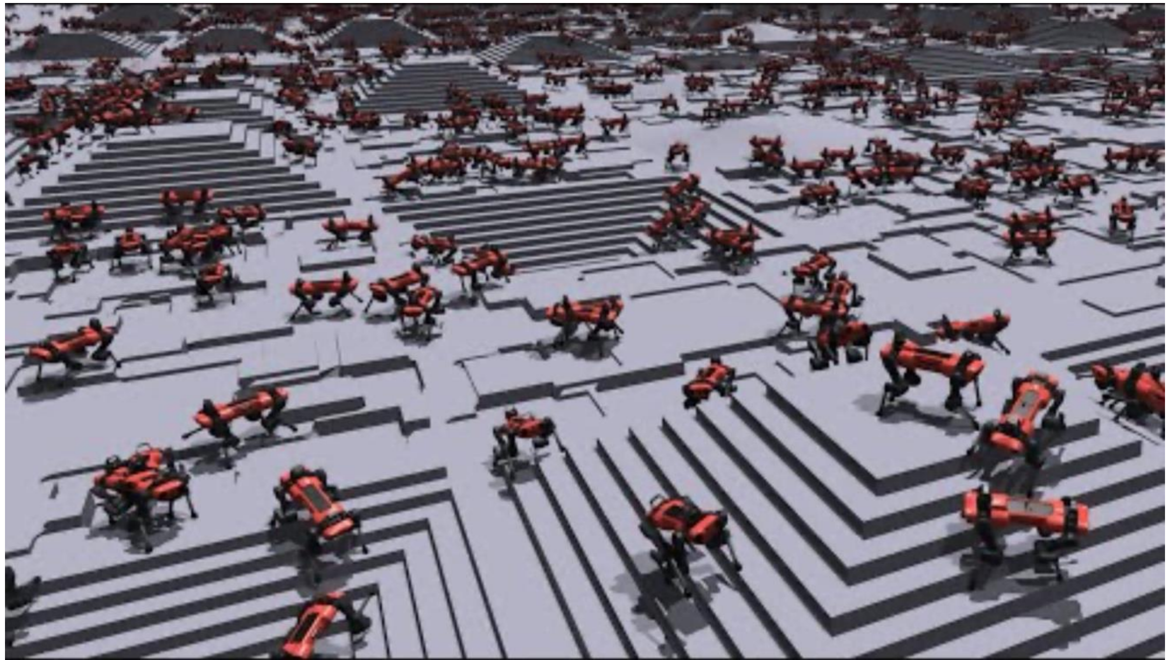
Off-Policy Learning

Most of the time in RL, collecting the data is computationally expensive.

So far, we've looked at an example, learned from it, and discarded it.

In all our other problems, we always learned from data multiple times (i.e., epochs)

Maybe we shouldn't throw away useful data immediately...

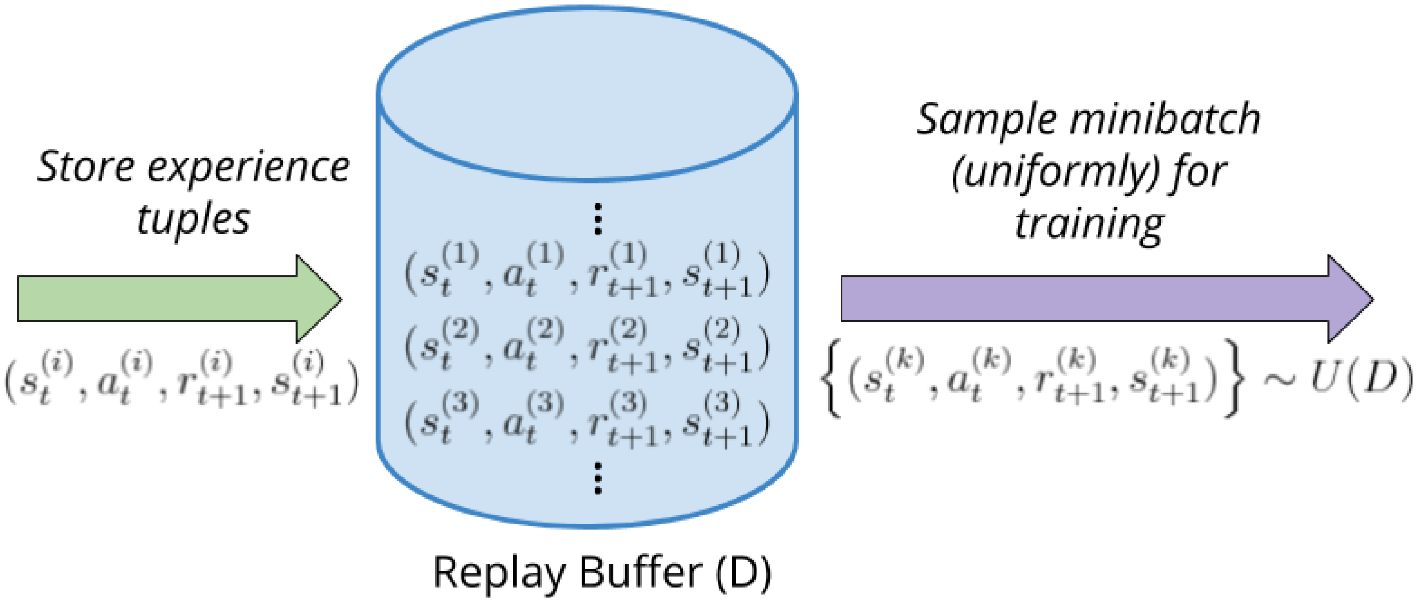


Experience Replay and Replay Buffers

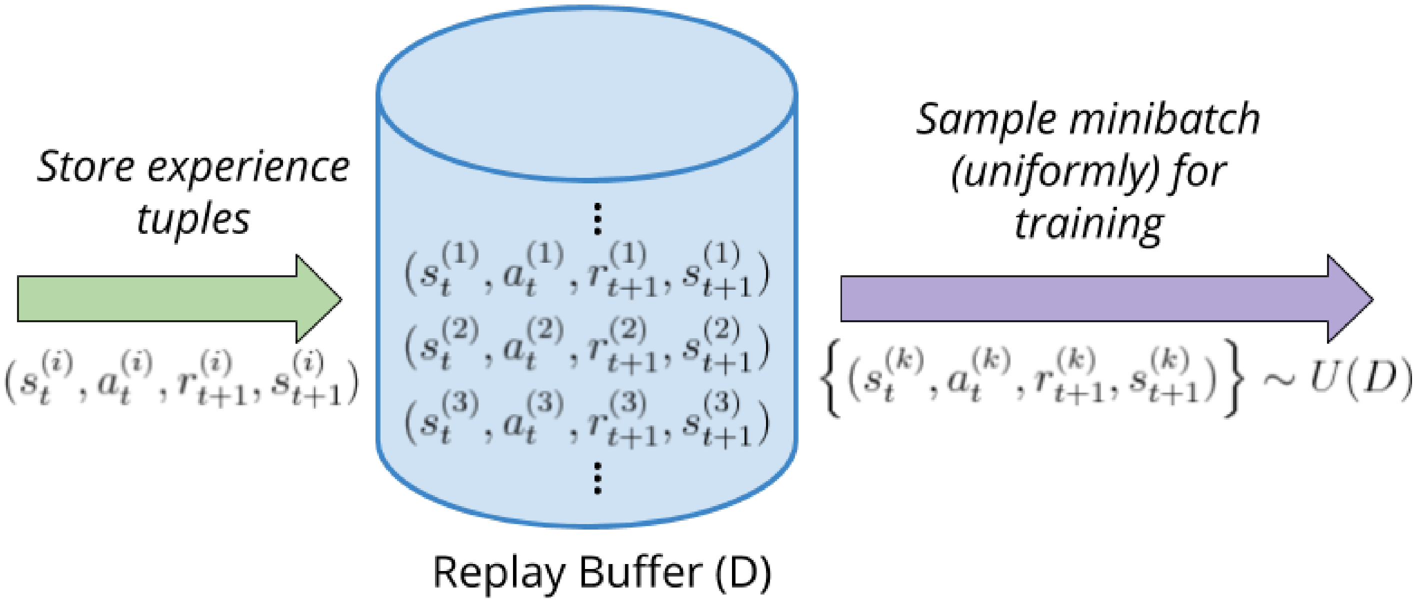
Keep a memory of experiences (state, action, reward, next_state)

As you collect new experiences, remove oldest experiences from buffer

To train model, sample batch of data from buffer

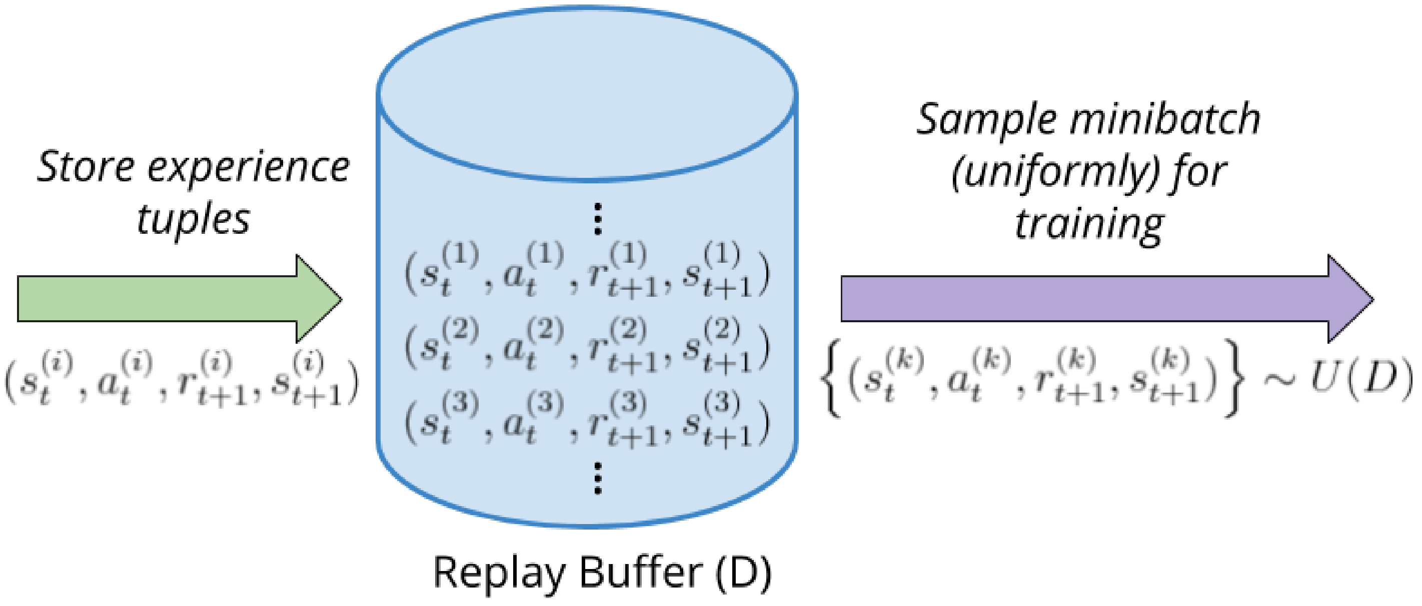


On-Policy Learning



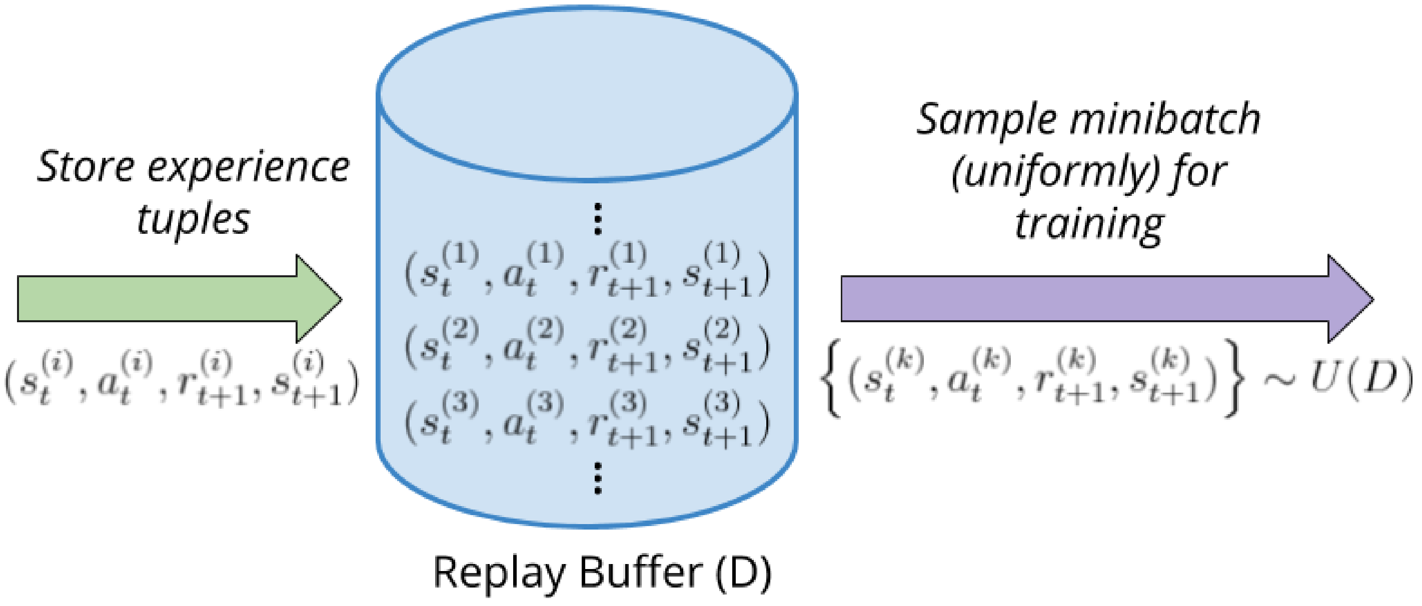
On-Policy Learning

Can we use Replay Buffers with On-Policy learning algorithms (e.g., REINFORCE, Actor-Critic, etc.)?



On-Policy Learning

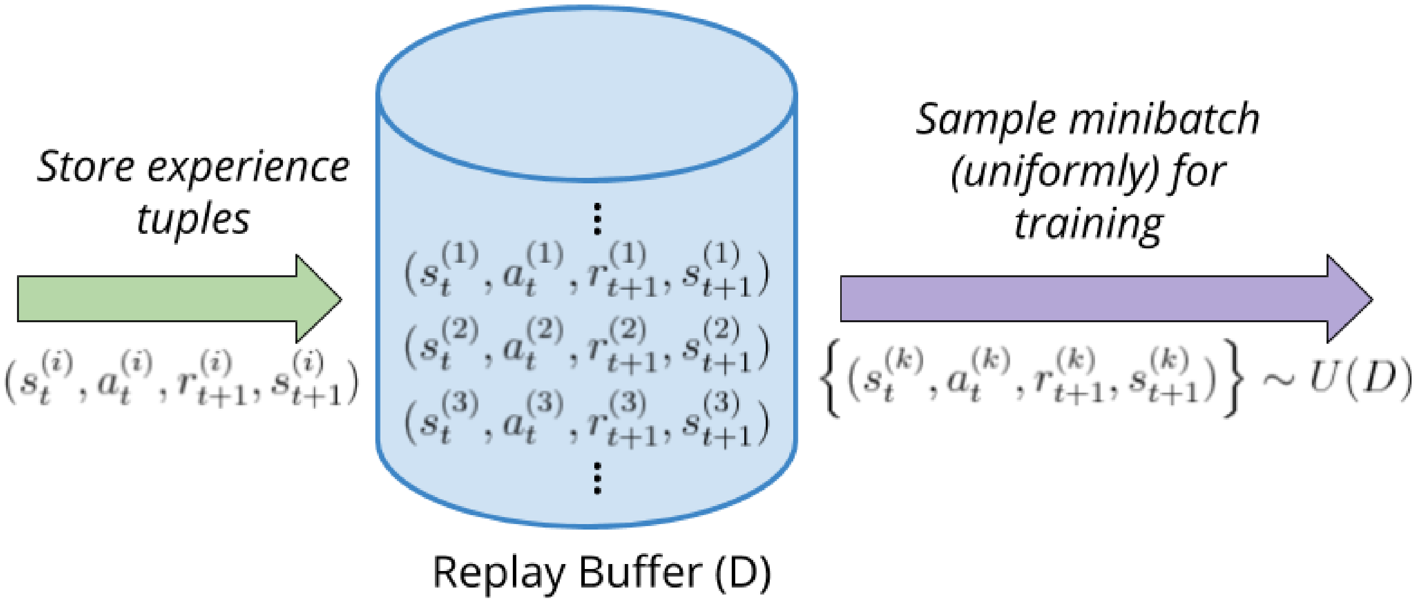
Can we use Replay Buffers with On-Policy learning algorithms (e.g., REINFORCE, Actor-Critic, etc.)?



No! Data in the buffer was collected with an older policy and we can only learn on experiences collected using the current policy...

On-Policy Learning

Can we use Replay Buffers with On-Policy learning algorithms (e.g., REINFORCE, Actor-Critic, etc.)?



No! Data in the buffer was collected with an older policy and we can only learn on experiences collected using the current policy…

But what if we actually could...

Off-Policy Policy Gradient:

Data collected under policy $\beta(a|s)$ (i.e., older version of policy)

We can re-weight our gradient according to the old policy:

$$\rho = \frac{\pi(a|s)}{\beta(a|s)}$$

$$\nabla_\theta J(\theta) = \sum_{(s,a)\in batch} \rho \cdot Q^\pi(s,a) \nabla_\theta \ln \pi(s,a)$$

Actor-Critic with Importance Sampling

Off-Policy Actor Critic: https://arxiv.org/pdf/1205.4839

But what if we actually could...

Off-Policy Policy Gradient:

Data collected under policy $\beta(a|s)$ (i.e., older version of policy)

We can re-weight our gradient according to the old policy:

$$\rho = \frac{\pi(a|s)}{\beta(a|s)}$$

$$\nabla_\theta J(\theta) = \sum_{(s,a)\in batch} \rho \cdot Q^\pi(s,a) \nabla_\theta \ln \pi(s,a)$$

Actor-Critic with Importance Sampling

Store action probabilities $\beta(a|s)$ in replay buffer

Off-Policy Actor Critic: https://arxiv.org/pdf/1205.4839

Trust Region Policy Optimization

Insight: the reason that variance is bad is that it can cause large updates to π_θ

Add a constraint to how large of an update can be applied:

KL-Divergence between old and new policy must be below some hyperparameter Δ

$$D_{KL}(\pi_\theta^{new}(\cdot\,|s)|\left|\pi_\theta^{old}(\cdot\,|s)\right) \leq \Delta$$

$$\rho = \frac{\pi^{new}(a|s)}{\pi^{old}(a|s)}$$

$$J^{TRPO}(\theta) = \mathbb{E}\left[\rho \cdot (r + \gamma V^{\pi^{old}}(s') - V^{\pi^{\mathrm{old}}}(s)\right)]$$

Paper: https://arxiv.org/pdf/1502.05477

Proximal Policy Optimization

TRPO is complicated…

What if instead of constraining the update with KL-Divergence, we clipped the update if it's too big…

$$\rho_{clipped} = clip[\frac{\pi^{new}(a|s)}{\pi^{old}(a|s)}, 1 - \epsilon, 1 + \epsilon]$$

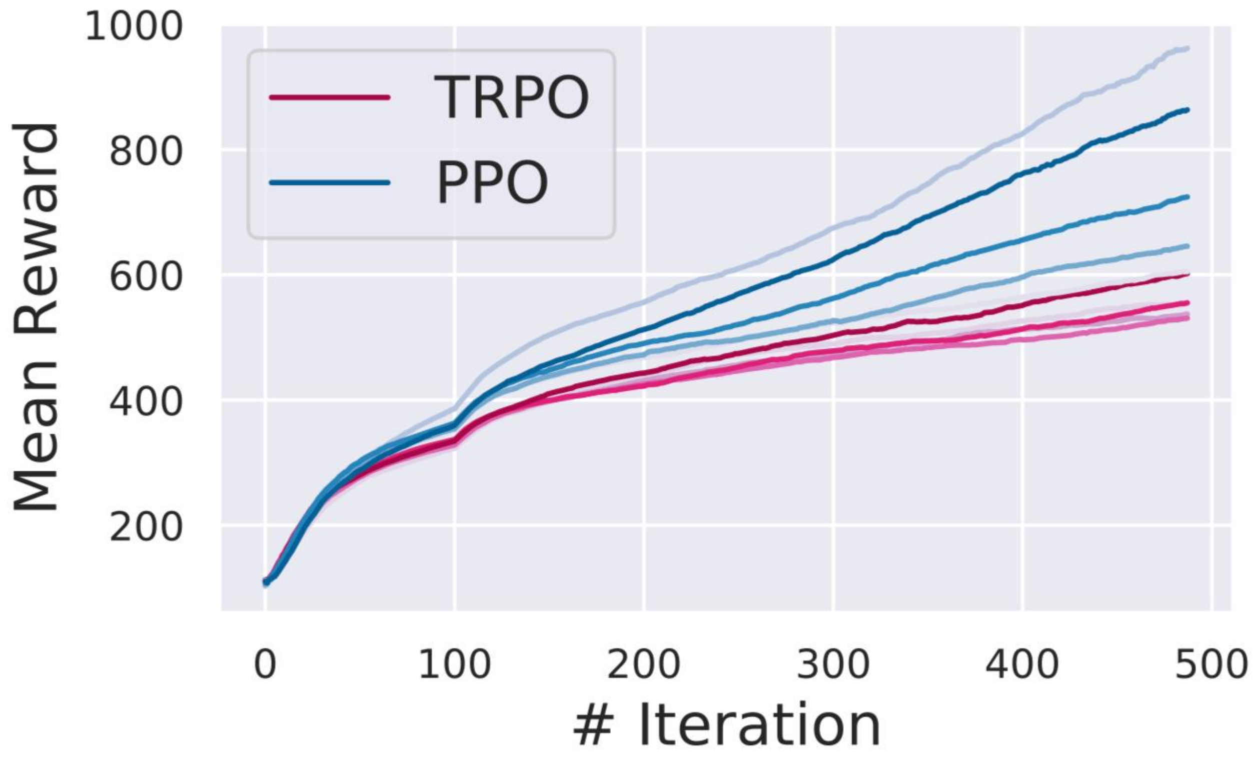
$$J^{PPO}(\theta) = \mathbb{E}[\min(\rho_{clipped} \cdot \left(r + \gamma V^{\pi^{old}}(s') - V^{\pi^{old}}(s)\right), \rho\left(r + \gamma V^{\pi^{old}}(s') - V^{\pi^{old}}(s)\right)]$$

Spinning Up PPO: https://spinningup.openai.com/en/latest/algorithms/ppo.html

PPO

PPO is (basically) State-Of-The-Art (SOTA)

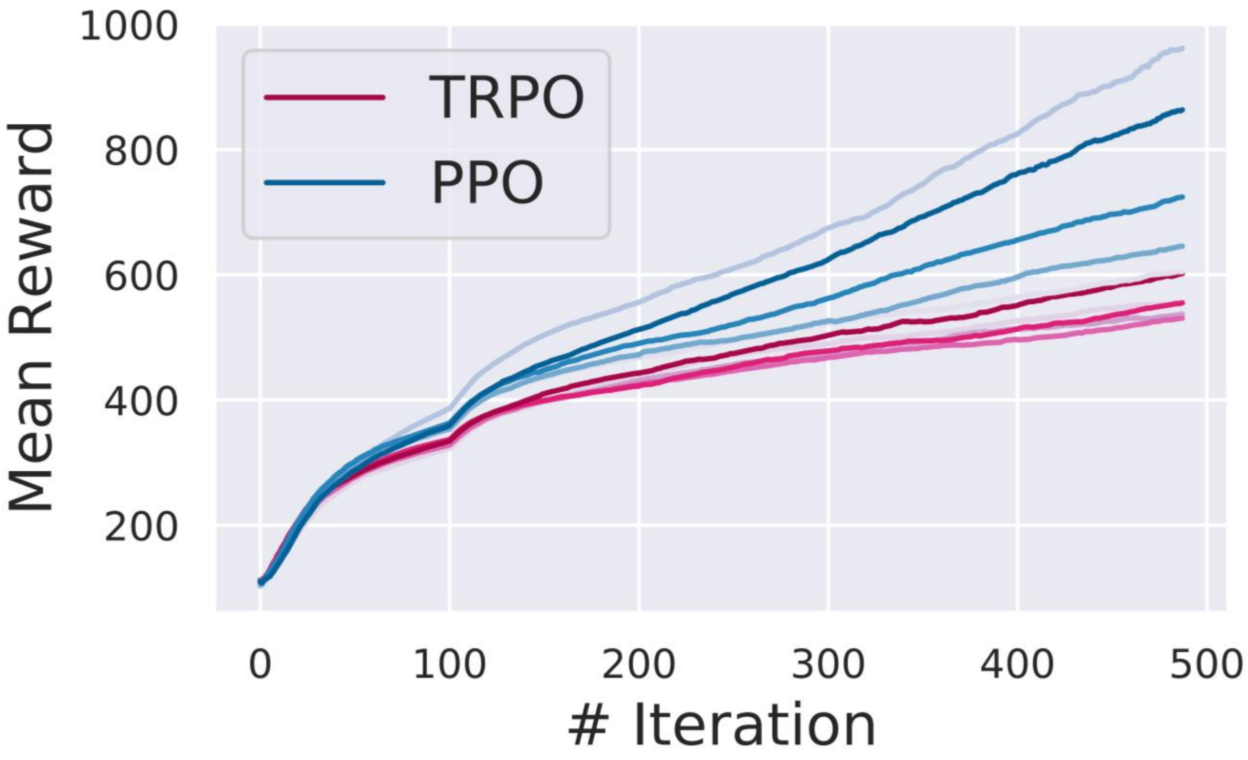
Provides **fast**, **sample-efficient**, and **stable** training



PPO

PPO is (basically) State-Of-The-Art (SOTA)

Provides **fast**, **sample-efficient**, and **stable** training



PPO: OpenAI5

PPO

Final phase of training ChatGPT