Actor Critic I

CISC 7404 - Decision Making

Steven Morad

University of Macau

Admin

Admin

How is homework 2?

Admin

Quiz next week

Study:

- Actor critic (today)
- Policy gradient
- Deep Q learning
- Expected returns

Final Project

Final Project

Final project information is released

Suggest project and group members by next Friday (28th)

Find (or create) a gymnasium environment

- Ensure your task is MDP
- Can also try POMDP, but make sure you are prepared!
- Groups of 5, results should be impressive
- Due just before final exam study week

https://ummoodle.um.edu.mo/pluginfile.php/6900679/mod_resource/content/6/project.pdf

Review

Value Policy Gradient

Value Policy Gradient

Today, we will investigate modern forms of policy gradient

This is what many researchers use today for impressive tasks

One algorithm we learn today can play Pokemon

https://youtu.be/DcYLT37ImBY?si=jJfZyYwFkPYMJYMy

Value Policy Gradient

$$\nabla_{\theta_\pi} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\pi] = \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\pi] \cdot \nabla_{\theta_\pi} \log \pi(a_0 \mid s_0; \theta_\pi)$$

We previously computed the Monte Carlo policy gradient (REINFORCE)

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

Question: Why don't we always use Monte Carlo?

Answer: Requires an infinite return!

Value Policy Gradient

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

Question: Alternative to Monte Carlo return?

Can use Q or V function with TD objective

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

$$Q(s_0, a_0, \theta_\pi) = \mathbb{E}[\mathcal{R}(s_1) \mid s_0, a_0, \theta_\pi] + \gamma Q(s_1, a_1, \theta_\pi)$$

Before:

$a_0 = \arg\max_{a \in A} Q(s, a, \theta_\pi)$

Now: $a \sim \pi(\cdot \mid s; \theta_\pi)$

$V = Q$ in this case

Value Policy Gradient

Policy gradient objective uses the return

$$\nabla_{\theta_\pi} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\pi] = \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\pi] \cdot \nabla_{\theta_\pi} \log \pi(a_0 \mid s_0; \theta_\pi)$$

Estimate return using value function

$$\mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\pi] = V(s_0, \theta_\pi)$$

Combining V / Q with policy gradient called **actor-critic**

Actor pick action

$$\nabla_{\theta_\pi} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\pi] = V(s_0, \theta_\pi) \cdot \nabla_{\theta_\pi} \log \pi(a_0 \mid s_0; \theta_\pi)$$

Critic gives actor score

Value Policy Gradient

Definition: Value Policy Gradient is an iterative process that jointly trains a policy network and value function

$$\theta_{\pi, i+1} = \theta_{\pi, i} + \alpha \cdot \underbrace{V(s_0, \theta_{\pi, i}, \theta_{V, i})}_{\text{Expected return}} \cdot \nabla_{\theta_{\pi, i}} \log \pi(a_0 \mid s_0; \theta_{\pi, i})$$

$$\theta_{V, i+1} =$$

$$\arg\min_{\theta_{V, i}} \underbrace{\left(V(s_0, \theta_{\pi, i}, \theta_{V, i}) - \left(\hat{\mathbb{E}}[\mathcal{R}(s_1) \mid s_0; \theta_\pi] + \neg d \cdot \gamma \cdot V(s_0, \theta_{\pi, i}, \theta_{V, i})\right)\right)^2}_{\text{TD error}}$$

Repeat process until convergence

Can train policy with single transition s_0, a_0, s_1, r_0, d_0

Advantage Actor Critic

Advantage Actor Critic

$$\nabla_{\theta_\pi} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\pi] = V(s_0, \theta_\pi) \cdot \nabla_{\theta_\pi} \log \pi(a_0 \mid s_0; \theta_\pi)$$

Question: Any scenarios where reward is always negative?

Answer: Distance to goal, $\mathcal{R}(s_{t+1}) = -(s_{t+1} - s_g)^2$

Question: What happens if reward is always negative?

Answer: Return always negative

$$\nabla_{\theta_\pi} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\pi] = - \mid V(s_0, \theta_\pi) \mid \cdot \nabla_{\theta_\pi} \log \pi(a_0 \mid s_0; \theta_\pi)$$

Similar results if reward is always positive

$$\nabla_{\theta_\pi} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\pi] = \mid V(s_0, \theta_\pi) \mid \cdot \nabla_{\theta_\pi} \log \pi(a_0 \mid s_0; \theta_\pi)$$

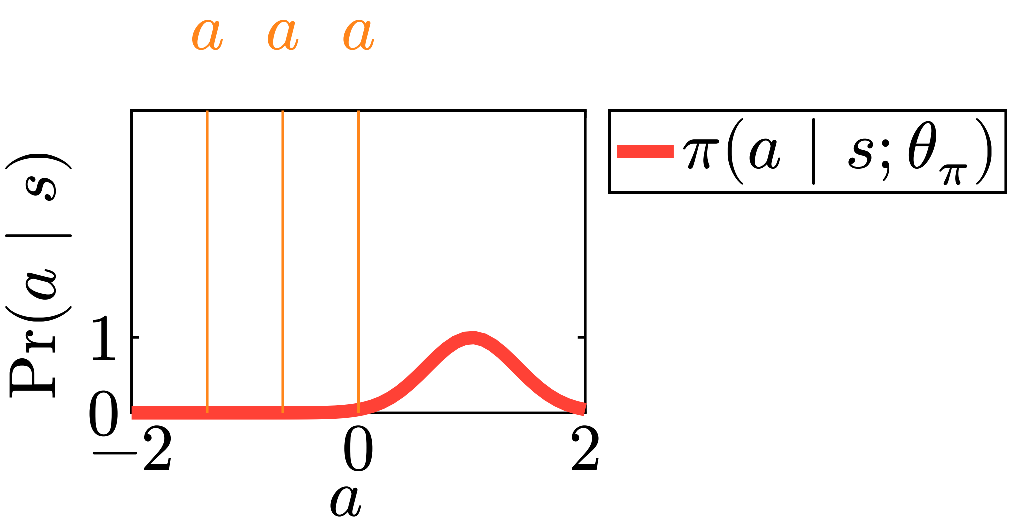
Advantage Actor Critic

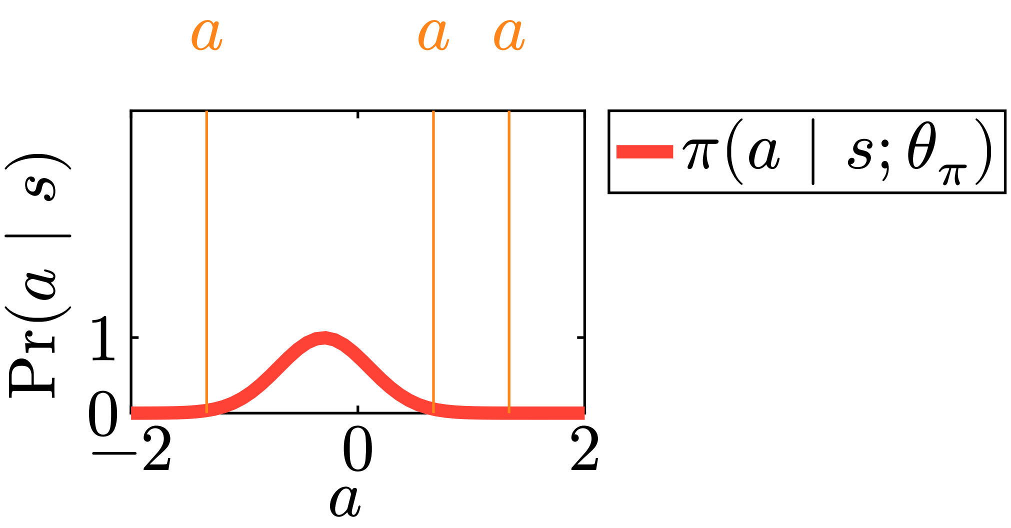
$$\nabla_{\theta_\pi} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\pi] = - \mid V(s_0, \theta_\pi) \mid \cdot \nabla_{\theta_\pi} \log \pi(a_0 \mid s_0; \theta_\pi)$$

Example: Environment with a single state and continuous actions

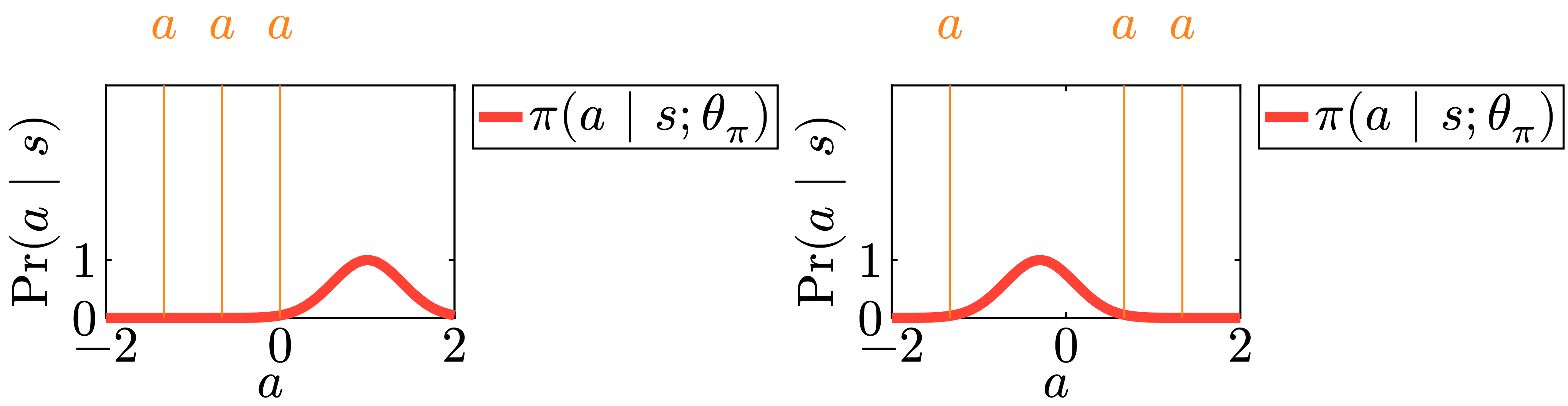
Sample k transitions for each gradient update

What if we cannot sample all possible actions?





Advantage Actor Critic

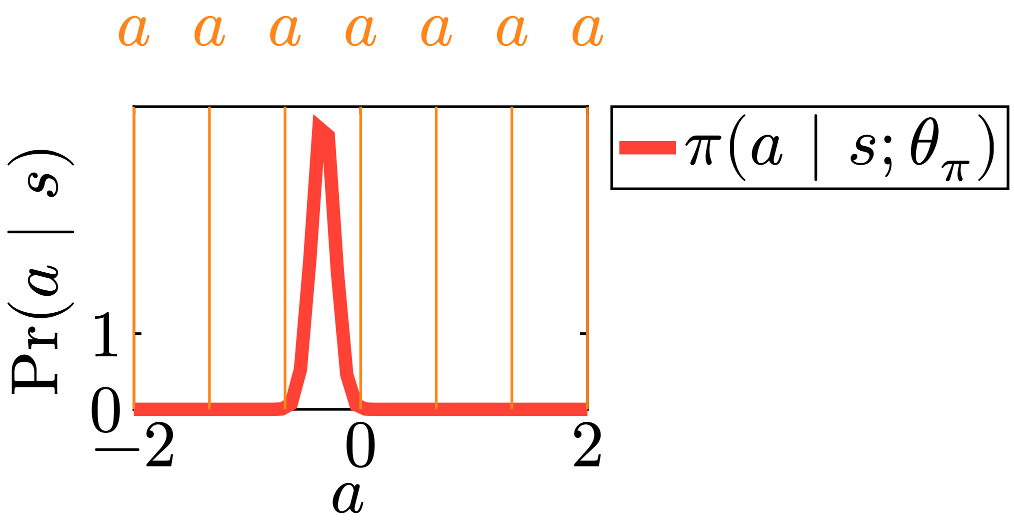


Policy keeps oscillating, can destabilize learning

Question: If we take 8 actions, will this fix it?

https://media0.giphy.com/media/v1.Y2lkPTc5MGI3NjExeGdqZm56NDgzcmY2Ym95dG13Ynczdm9lbDY0cGpjczdtMHBmcnJmMSZlcD12MV9pbnRlcm5hbF9naWZfYnlfaWQmY3Q9Zw/MVUyVpyjakkRW/giphy.gif

Advantage Actor Critic



Question: Any solutions?

Hint: Think about the mean of the return

Answer: Recenter return such that mean is zero

But we can do even better!

Advantage Actor Critic

What if we:

- Almost never update policy
- Update the policy **only** if action is better/worse than expected

Question: What is the expected performance of the policy?

$$\mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\pi] = V(s_0, \theta_\pi)$$

Question: How can we tell the performance of a specific action?

$$\mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0, a_0; \theta_\pi] = Q(s_0, a_0, \theta_\pi)$$

Question: How can we tell if an action is better/worse than expected?

$$\mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0, a_0; \theta_\pi] - \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\pi] = Q(s_0, a_0, \theta_\pi) - V(s_0, \theta_\pi)$$

Advantage Actor Critic

$$A(s_0, a_0, \theta_\pi) = Q(s_0, a_0, \theta_\pi) - V(s_0, \theta_\pi)$$

We call this the **advantage**, tells us if we should change policy

If action a_0 better than expected, increase policy probability

$$\theta_\pi = \theta_\pi + | A(s_0, a_0, \theta_\pi) | \cdot \nabla_{\theta_\pi} \log \pi(a_0 \mid s_0; \theta_\pi)$$

If action a_0 worse than expected, reduce probability

$$\theta_\pi = \theta_\pi - | A(s_0, a_0, \theta_\pi) | \cdot \nabla_{\theta_\pi} \log \pi(a_0 \mid s_0; \theta_\pi)$$

If action a_0 is as expected, do nothing

$$\theta_\pi = \theta_\pi + 0 \cdot \nabla_{\theta_\pi} \log \pi(a_0 \mid s_0; \theta_\pi)$$

Advantage Actor Critic

Definition: The advantage A determines the relative advantage/disadvantage of taking an action a_0 in state s_0 for a policy θ_π

$$A(s_0, a_0, \theta_\pi) = Q(s_0, a_0, \theta_\pi) - V(s_0, \theta_\pi)$$

Advantage Actor Critic

$$A(s_0, a_0, \theta_\pi) = Q(s_0, a_0, \theta_\pi) - V(s_0, \theta_\pi)$$

Advantage requires both Q and V

But earlier, we saw $Q = V$ in some circumstances

Question: Can we replace Q with V? How?

HINT: Think about TD error, use s_1

$$A(s_0, \theta_\pi) = - \underbrace{V(s_0, \theta_\pi)}_{\text{What we expect}} + \underbrace{(\mathbb{E}[\mathcal{R}(s_1) \mid s_0; \theta_\pi] + \neg d \gamma V(s_1, \theta_\pi))}_{\text{What happens}}$$

Better than expected: $|A| > 0$, worse $|A| < 0$

Advantage Actor Critic

Definition: Advantage actor critic (A2C) updates the policy and value functions using the advantage, and repeats until convergence

$$A(s_0, \theta_\pi, \theta_V) = -V(s_0, \theta_\pi, \theta_V) + \left(\hat{\mathbb{E}}[\mathcal{R}(s_1) \mid s_0; \theta_\pi] + \neg d \gamma V(s_1, \theta_\pi, \theta_V)\right)$$

$$\theta_{\pi, i+1} = \theta_{\pi, i} + \alpha \cdot \underbrace{A(s_0, \theta_{\pi, i}, \theta_{V, i})}_{\text{Advantage}} \cdot \underbrace{\nabla_{\theta_{\pi, i}} \log \pi(a_0 \mid s_0; \theta_{\pi, i})}_{\text{Policy gradient}}$$

$$\theta_{V, i+1} =$$

$$\arg\min_{\theta_{V, i}} \underbrace{\left(V(s_0, \theta_{\pi, i}, \theta_{V, i}) - \left(\hat{\mathbb{E}}[\mathcal{R}(s_1) \mid s_0; \theta_\pi] + \neg d \gamma V(s_0, \theta_{\pi, i}, \theta_{V, i})\right)\right)^2}_{\text{TD error}}$$

Off-Policy Gradient

Off-Policy Gradient

$$\nabla_{\theta_\pi} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\pi] = \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\pi] \cdot \nabla_{\theta_\pi} \log \pi(a_0 \mid s_0; \theta_\pi)$$

Question: Is policy gradient off-policy or on-policy?

Answer: On-policy, expected return depends on θ_π

Question: Why do we care about being off-policy?

Answer: Algorithm can reuse data, much more efficient

Question: What do we need to make policy gradient off-policy?

Need to be able to approximate $\mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\pi]$ using $\mathbb{E}\left[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\beta\right]$

Question: Any math students know how to do this?

Off-Policy Gradient

In **importance sampling**, we want to estimate

$$\mathbb{E}[f(x) \mid x \sim \Pr(\cdot \ ; \theta_a)]$$

Unfortunately, we only have data from

$$\mathbb{E}[f(x) \mid x \sim \Pr(\cdot \ ; \theta_b)]$$

We can use their ratio to approximate the expectation

$$\mathbb{E}[f(x) \mid x \sim \Pr(\cdot \mid \theta_a)] = \mathbb{E}\left[f(x) \cdot \frac{\Pr(\cdot \ ; \theta_a)}{\Pr(\cdot \ ; \theta_b)} \middle| x \sim \Pr(\cdot \ ; \theta_b) \right]$$

Question: How can this make policy gradient off policy?

Off-Policy Gradient

$$\mathbb{E}[f(x) \mid x \sim \Pr(\cdot \ ; \theta_a)] = \mathbb{E}\left[f(x) \cdot \frac{\Pr(\cdot \ ; \theta_a)}{\Pr(\cdot \ ; \theta_b)} \middle| x \sim \Pr(\cdot \ ; \theta_b)\right]$$

Consider our current policy is θ_π, we want $\mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\pi]$

We use a **behavior policy** θ_β to collect data $\mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\beta]$

θ_β can be an old policy or some other policy

Reward following θ_π

$$\mathbb{E}[\mathcal{R}(s_1) \mid s_0; \theta_\pi] = \mathbb{E}\left[\mathcal{R}(s_1) \cdot \frac{\pi(a \mid s_0; \theta_\pi)}{\pi(a \mid s_0; \theta_\beta)} \middle| s_0; \theta_\beta\right]$$

Reward following θ_β

Off-Policy Gradient

$$\mathbb{E}[\mathcal{R}(s_1) \mid s_0; \theta_\pi] = \mathbb{E}\left[\mathcal{R}(s_1) \cdot \frac{\pi(a \mid s_0; \theta_\pi)}{\pi(a \mid s_0; \theta_\beta)} \middle| s_0; \theta_\beta\right]$$

How does this actually work?

Rewrite without expectation to clarify

$$\mathbb{E}[\mathcal{R}(s_1) \mid s_0; \theta_\pi] = \underbrace{\sum_{s_1 \in S} \mathcal{R}(s_1) \sum_{a_0 \in A} \mathrm{Tr}(s_1 \mid s_0, a_0) \pi(a_0 \mid s_0; \theta_\beta)}_{\text{Expected reward}} \underbrace{\frac{\pi(a_0 \mid s_0; \theta_\pi)}{\pi(a_0 \mid s_0; \theta_\beta)}}_{\text{Correction}}$$

Off-Policy Gradient

$$\mathbb{E}[\mathcal{R}(s_1) \mid s_0; \theta_\pi] = \underbrace{\sum_{s_1 \in S} \mathcal{R}(s_1) \sum_{a_0 \in A} \mathrm{Tr}(s_1 \mid s_0, a_0) \pi(a_0 \mid s_0; \theta_\beta)}_{\text{Expected reward}} \underbrace{\frac{\pi(a_0 \mid s_0; \theta_\pi)}{\pi(a_0 \mid s_0; \theta_\beta)}}_{\text{Correction}}$$

Terms cancel!

$$\mathbb{E}[\mathcal{R}(s_1) \mid s_0; \theta_\pi] = \sum_{s_1 \in S} \mathcal{R}(s_1) \sum_{a_0 \in A} \mathrm{Tr}(s_1 \mid s_0, a_0) \cancel{\pi(a_0 \mid s_0; \theta_\beta)} \frac{\pi(a_0 \mid s_0; \theta_\pi)}{\cancel{\pi(a_0 \mid s_0; \theta_\beta)}}$$

$$\mathbb{E}[\mathcal{R}(s_1) \mid s_0; \theta_\pi] = \sum_{s_1 \in S} \mathcal{R}(s_1) \sum_{a_0 \in A} \mathrm{Tr}(s_1 \mid s_0, a_0) \pi(a_0 \mid s_0; \theta_\pi)$$

Left with expression for expected reward following θ_π

Off-Policy Gradient

$$\mathbb{E}[\mathcal{R}(s_1) \mid s_0; \theta_\pi] = \mathbb{E}\left[\mathcal{R}(s_1) \cdot \frac{\pi(a \mid s_0; \theta_\pi)}{\pi(a \mid s_0; \theta_\beta)} \middle| s_0; \theta_\beta\right]$$

$$\mathbb{E}[\mathcal{R}(s_1) \mid s_0; \theta_\pi] = \sum_{s_1 \in S} \mathcal{R}(s_1) \sum_{a_0 \in A} \mathrm{Tr}(s_1 \mid s_0, a_0) \pi(a_0 \mid s_0; \theta_\beta) \frac{\pi(a_0 \mid s_0; \theta_\pi)}{\pi(a_0 \mid s_0; \theta_\beta)}$$

We can apply the same approach to find the off-policy return

I won't derive it, just trust me

$$\mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\pi] = \mathbb{E}\left[\mathcal{G}(\boldsymbol{\tau}) \prod_{t=0}^{\infty} \frac{\pi(a_t \mid s_t; \theta_\pi)}{\pi(a_t \mid s_t; \theta_\beta)} \middle| s_0; \theta_\beta\right]$$

Off-Policy Gradient

Definition: Off-policy gradient uses importance sampling to learn from off-policy data

$$\mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\pi] = \mathbb{E}\left[\mathcal{G}(\boldsymbol{\tau}) \prod_{t=0}^{\infty} \frac{\pi(a_t \mid s_t; \theta_\pi)}{\pi(a_t \mid s_t; \theta_\beta)} \middle| s_0; \theta_\beta \right]$$

$$\theta_{\pi, i+1} = \theta_{\pi, i} + \alpha \cdot \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\pi] \cdot \nabla_{\theta_{\pi, i}} \log \pi\left(a_0 \mid s_0; \theta_{\pi, i}\right)$$

Off-Policy Gradient

$$\mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_\pi] = \mathbb{E}\left[\mathcal{G}(\boldsymbol{\tau}) \prod_{t=0}^{\infty} \frac{\pi(a_t \mid s_t; \theta_\pi)}{\pi(a_t \mid s_t; \theta_\beta)} \middle| s_0; \theta_\beta\right]$$

Question: Why did I tell you policy gradient is on policy?

Answer: Off-policy gradient does not work in most cases

Question: Why? HINT: What happens to $\prod$?

$$\prod_{t=0}^{\infty} \frac{\pi(a_t \mid s_t; \theta_\pi)}{\pi(a_t \mid s_t; \theta_\beta)} \to 0, \infty$$

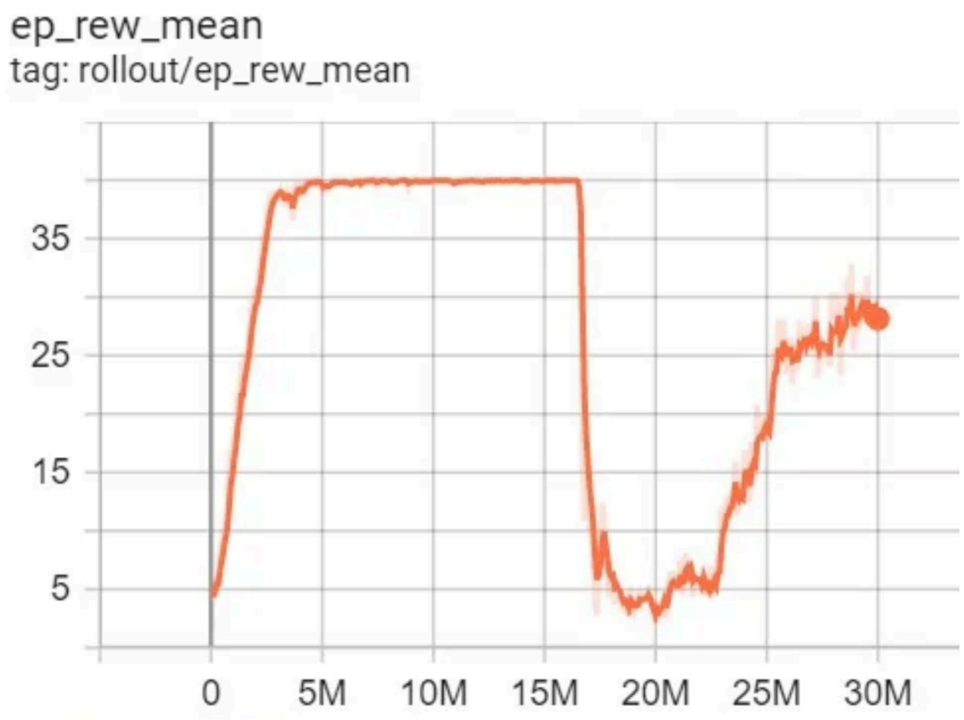
Only works if $\pi(a_t \mid s_t; \theta_\pi) \approx \pi(a_t \mid s_t; \theta_\beta)$

Trust Regions

Trust Regions

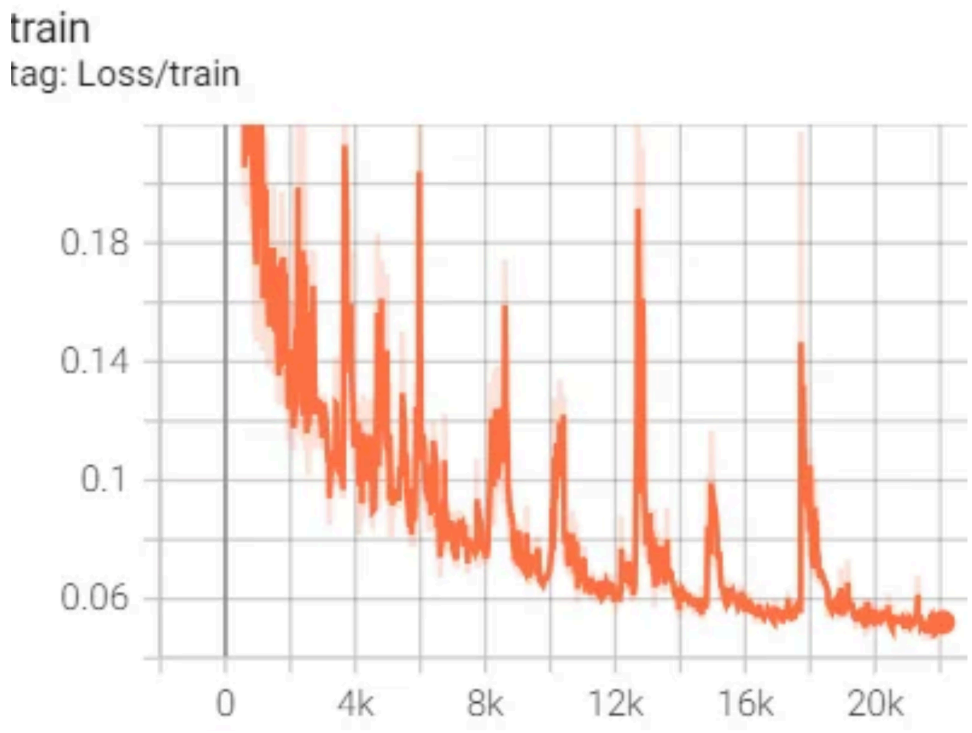
Training policies in RL is difficult

We often see behavior like this



Question: Any idea why?

Trust Regions



See it in supervised learning too

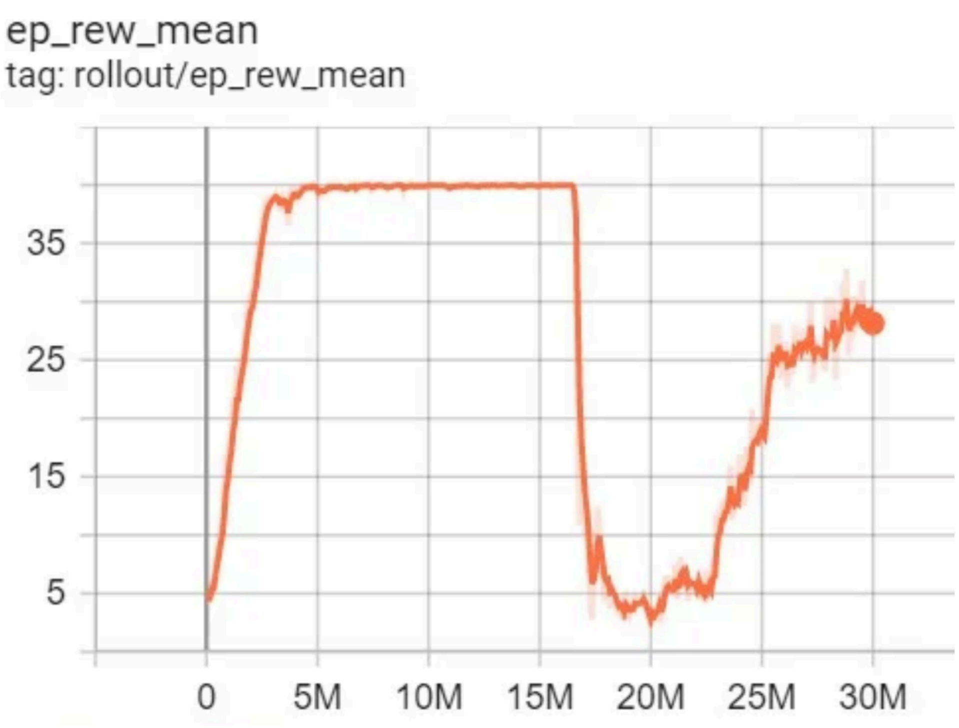
Sometimes, the gradient is inaccurate producing a bad update

In supervised learning, the network can easily recover

With policy gradient, it is much harder to recover

Question: Why?

Trust Regions



Our policy provides the training data $a \sim \pi(\cdot \mid s; \theta_\pi)$

One bad update breaks the policy

Policy collects useless data

Off-policy methods recover from "good" data from replay buffer

On-policy methods cannot!

We must be very careful when updating our neural network policy

Trust Regions

Question: How can we make sure our policy does not change too much?

Lower learning rate? Can help a little

Small parameter updates cause large changes in deep networks

$$\pi(a \mid s_A; \theta_{\pi,i}) = \begin{bmatrix} 0.4 \\ 0.6 \end{bmatrix} \qquad \pi(a \mid s_A; \theta_{\pi,i+1}) = \begin{bmatrix} 1.0 \\ 0.0 \end{bmatrix}$$

Constraining changes in parameter space does not work!

Question: What else can we constrain?

Answer: The action distributions

Trust Regions

Can measure the difference in distributions using KL divergence

$$\mathrm{KL}[\Pr(X), \Pr(Y)] \in [0, \infty]$$

Policies are just action distributions

$$\mathrm{KL}\left[\pi\left(a \mid s; \theta_{\pi,i}\right), \pi\left(a \mid s; \theta_{\pi,i+1}\right)\right]$$

Introduce **trust region** k to prevent large policy changes

$$\nabla_{\theta_{\pi,i}} \mathbb{E}\left[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_{\pi,i}\right] = V\left(s_0, \theta_{\pi,i}\right) \cdot \nabla_{\theta_\pi} \log \pi\left(a_0 \mid s_0; \theta_{\pi,i}\right)$$

$$s.t.\ \ \mathrm{KL}\left[\pi\left(a \mid s; \theta_{\pi,i-1}\right), \pi\left(a \mid s; \theta_{\pi,i}\right)\right] < k$$

See Trust Region Policy Optimization (TRPO), Natural Policy Gradient

Trust Regions

$$\nabla_{\theta_{\pi, i}} \mathbb{E}[\mathcal{G}(\boldsymbol{\tau}) \mid s_0; \theta_{\pi, i}] = V(s_0, \theta_{\pi, i}) \cdot \nabla_{\theta_\pi} \log \pi(a_0 \mid s_0; \theta_{\pi, i})$$

$$s.t. \; \mathrm{KL}[\pi(a \mid s; \theta_{\pi, i-1}), \pi(a \mid s; \theta_{\pi, i})] < k$$

Constrained optimization can be expensive and tricky to implement

Often requires computing the Hessian (second-order gradient)

Hack: Add KL term to the objective (soft constraint)

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

$$\nabla_{\theta_\pi} [\log \pi(a_0 \mid s_0; \theta_{\pi, i}) - \mathrm{KL}[\pi(a \mid s; \theta_{\pi, i-1}), \pi(a \mid s; \theta_{\pi, i})]]$$

Proximal Policy Optimization

Proximal Policy Optimization

Proximal policy optimization (PPO) combines everything we learned today

- Value function
- Advantage
- Off-policy gradient
- Trust regions

PPO designed to be very sample efficient

It is *almost* on-policy (but very slightly off-policy)

Proximal Policy Optimization

```python
for epoch in range(epochs):
    batch = collect_rollout(theta_beta)
    # Minibatching learns faster
    # but is very slightly off-policy!
    for minibatch in batch:
        theta_pi = update_pi(
            theta_pi, theta_beta, theta_V, batch
        )
        theta_V = update_V(theta_V, batch)
    theta_beta = theta_pi
```

Proximal Policy Optimization

There are different variations of PPO

- PPO clip
- PPO KL penalty
- PPO clip + KL penalty
- PPO clip + KL penalty + entropy

We will focus on the simplest version (PPO KL penalty)

Proximal Policy Optimization

$$\theta_{\pi, i+1} = \theta_{\pi, i} + \alpha \cdot \underbrace{\left(\overbrace{\frac{\pi(a \mid s; \theta_{\pi, i})}{\pi(a \mid s; \theta_\beta)}}^{\text{Off-policy correction for minibatch}} \overbrace{A(s, \theta_\beta, \theta_V)}^{\text{Advantage}} \right)}_{\text{Value}}$$

$$\cdot \left(\underbrace{\nabla_{\theta_{\pi,i}} \left[\log \pi(a_0 \mid s_0; \theta_{\pi, i}) \right]}_{\text{Policy gradient}} - \rho \nabla_{\theta_{\pi, i+1}} \left[\underbrace{\mathrm{KL}\left[\pi(a_0 \mid s_0; \theta_{\pi, i+1}), \pi(a_0 \mid s_0; \theta_\beta) \right]}_{\text{Trust region}} \right] \right)$$

$$A(s_0, \theta_\beta, \theta_V) = -V(s_0, \theta_\beta, \theta_V) + \left(\hat{\mathbb{E}}[\mathcal{R}(s_1) \mid s_0; \theta_\beta] + \neg d \gamma V(s_1, \theta_\beta, \theta_V) \right)$$

$$\theta_{V, i+1} = \arg \min_{\theta_{V,i}} \left(V(s_0, \theta_\beta, \theta_{V,i}) - \left(\hat{\mathbb{E}}[\mathcal{R}(s_1) \mid s_0; \theta_\beta] + \neg d \gamma V(s_0, \theta_\beta, \theta_{V,i}) \right) \right)^2$$

Proximal Policy Optimization

Personal opinion: PPO is overrated, for some reason very popular

Many hyperparameters, hard to implement, computationally expensive

Cohere finds REINFORCE better than PPO for LLM training

https://arxiv.org/pdf/2402.14740v1

Our experiments find that Q learning outperforms PPO

My suggestion:

- Try A2C first, solid actor-critic method, easy to implement
- Regularization (weight decay, layer norm, etc) helpful
- You can make any algorithm work with enough effort!