Deep Reinforcement Learning Topics

- Online RL:
  - Start with no data; interact with environment and incrementally improve policy. What we did when studying tabular RL.
  - Two broad classes of algorithms:
    - On-policy
    - Off-Policy
- Offline RL:
  - Start with lots of episodic data
  - Train a good policy without interacting with the environment
  - Imitation Learning / Behavioral Cloning is a subcase.
- Other selected, more advanced topics

Policy Gradient Algorithm

- Online
- On-policy
- Monte Carlo (rather than TD) algorithm
- State space huge or even infinite
- Action space finite and small (not essential)
- Elegant theory

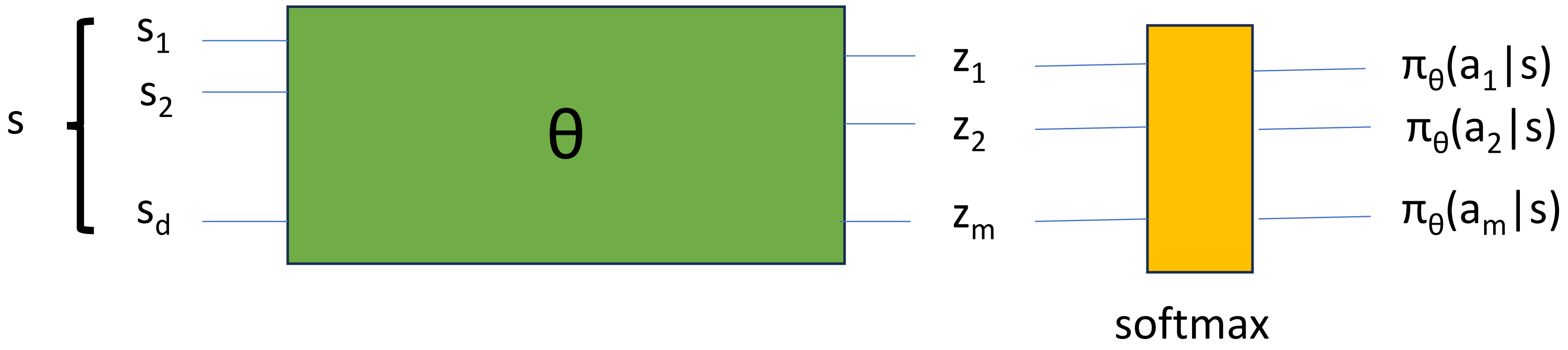
Refresher on some concepts from probability

- Let X be a random variable with probability mass function $p_X(x)$.
- Suppose we want to estimate E[X] := $\sum_x x p_X(x)$ but we don't explicitly know $p_X(x)$ or direct calculation of E[X] is difficult.
- But suppose we can sample x from $p_X(.)$ (write $x \sim p_X$)
- How can we estimate E[X]?
- How can we estimate E[f(X)] ?

- Suppose X is a random variable. X is an unbiased estimator for a deterministic scalar c if .... (fill in the blank)
- Note that X ~ $p_X(.)$ is always an unbiased estimator for E[X].

Back to DRL: Policy Network

- Suppose state space has dimension d: s = $(s_1,\ldots,s_d)$. (Could be an image.)
- Suppose action space $\{a_1,a_2,\ldots,a_m\}$ is small.
- Consider using neural network (e.g., convolutional network) for the policy: Policy network maps state s to probability distribution over action space:



- So when in state s, chose action a with probability $\pi_\theta(a|s)$.
- As we vary parameter θ, we get different policies.

Expected Return under policy π_θ

- $v_\theta := E_{\pi\theta}[\sum_{t=1}^{T} R_t] := E_{\pi\theta}[G_0]$

- Our goal is to find a θ that provides a high value of v_θ
- Note that we are not optimizing over all policies, but instead over all parameterized policies (a subset). Since neural nets are expressive, this is okay.
- Note that $\pi_\theta(a|s)$ is a stochastic policy. But can be nearly deterministic
- Detail: We are assuming in above equation initial state is drawn from a fixed distribution.

- Quiz: For fixed θ, how can we estimate v_θ using episodes? Hint: Recall refresher on probability.

Quiz solution:

- Suppose we sample J episodes using policy π_θ
- Denote $G^{(j)}$ for the return of jth episode
- $G^{(j)}$ by definition is an unbiased estimator $E_{\pi\theta}[G^{(j)}] = v_\theta$
- Thus $\frac{1}{J}\sum_{j=1}^{J} G^{(j)}$ is an also an unbiased estimator, and has lower variance than just using one sample $G^{(j)}$

- Note that the $G^{(j)}$'s are i.i.d. So strong law of large numbers also applies, implying convergence with probability 1 as $J \to \infty$

Gradient Ascent

- To optimize v_θ , it is natural to consider gradient ascent:

$$\theta \leftarrow \theta + \alpha \nabla_\theta E_\theta [\sum_{t=1}^{T} R_t]$$

- But how can we estimate $\nabla_\theta E_\theta [\sum_{t=1}^{T} R_t]$?

- Quiz: Since $\frac{1}{J}\sum_{j=1}^{J} G^{(j)}$ is an unbiased estimator of $E_\theta [\sum_{t=1}^{T} R_t]$, can we simply use $\nabla_\theta \frac{1}{J}\sum_{j=1}^{J} G^{(j)}$?

- No: This will give zero, which is wrong.

Policy Gradient Theorem

Theorem: ∇_θ $E_\theta[\sum_{t=1}^{T} R_t] = E_\theta[(\sum_{t=1}^{T} R_t)\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(A_t|S_t)]$

This is something we can actually estimate:

Let $\tau = (s_0,a_0,r_1,s_1,a_1,r_2,\dots,s_{T-1},a_{T-1},r_T,s_T)$ be an episode drawn from π_θ and the environment.

$(\sum_{t=1}^{T} r_t)\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t|s_t)$ is an unbiased estimator of
$E_\theta[(\sum_{t=1}^{T} R_t)\nabla_\theta \sum_{t=0}^{T-1} \log \pi_\theta(A_t|S_t)]$

By PG Theorem, $E_\theta[(\sum_{t=1}^{T} R_t)\nabla_\theta \sum_{t=0}^{T-1} \log \pi_\theta(A_t|S_t)] = \nabla_\theta E_\theta[\sum_{t=1}^{T} R_t]$

Thus $(\sum_{t=1}^{T} r_t)\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t|s_t)$ is an unbiased estimator of $\nabla_\theta E_\theta[\sum_{t=1}^{T} R_t]$

Applying Policy Gradient Theorem

- We would like to update θ ← θ + α ∇_θ E_θ [$\sum_{t=1}^{T} R_t$]
- Need to estimate the gradient using episodes.
- Generate an episode Let τ = (s_0,a_0,r_1,s_1,a_1,r_2,…,s_{T-1},a_{T-1},s_T) with policy π_θ
- ($\sum_{t=1}^{T} r_t$) $\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)$ is an unbiased estimator of ∇_θ E_θ [$\sum_{t=1}^{T} R_t$]
- So we can use θ ← θ + α ($\sum_{t=1}^{T} r_t$) $\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)$
- Can obtain $\nabla_\theta \log \pi_\theta(a_t|s_t)$ by backpropagation
- To reduce variance of estimator, run many episodes and average. This is the so-called REINFORCE algorithm

REINFORCE Algorithm (a.k.a. Policy Gradient Algorithm)

Initialize θ in policy network

Repeat:

Use policy π_θ to obtain N episodes:

$$\tau^{(i)} = (\, s_0^{(i)}, a_0^{(i)}, r_1^{(i)}, s_1^{(i)}, a_1^{(i)}, r_2^{(i)}, \ldots, s_{T-1}^{(i)}, a_{T-1}^{(i)}, r_T^{(i)} \,), \quad \text{i=1,...,N}$$

Update the policy:

$$\theta \leftarrow \theta + \frac{\alpha}{N} \sum_{i=1}^{N} \Big(\sum_{t=1}^{T} r_t^{(i)} \Big) \sum_{t=1}^{T-1} \nabla_\theta log\, \pi_\theta(a_t^{(i)} | s_t^{(i)})$$

Proof of Policy Gradient Theorem

- Need to prove: ∇_θ E_θ [$\sum_{t=1}^{T} R_t$] = E_θ [($\sum_{t=1}^{T} R_t$) $\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(A_t|S_t)$]

- Express E_θ [$\sum_{t=1}^{T} R_t$] as $\sum_\tau G(\tau) p_{\pi_\theta}(\tau)$ where we are summing over all possible episodes τ.

- We can then write ∇_θ E_θ [$\sum_{t=1}^{T} R_t$] = $\nabla_\theta \sum_\tau G(\tau)\, p_{\pi_\theta}(\tau)$ = $\sum_\tau G(\tau) \nabla_\theta\, p_{\pi_\theta}(\tau)$

- Proof uses identity: $\nabla_x f(x) = f(x)\, \nabla_x \log f(x)$

- Details of proof given on whiteboard

Normalization reduces variance of estimator

- $G^{(i)} = \sum_{t=1}^{T} r_t^{(i)}$ is the return of episode i
- Let m and σ^2 be the mean and variance of $G^{(1)}, G^{(2)},\ldots,G^{(N)}$
- $G^{(i)} \leftarrow (G^{(i)} - m)/\sigma$
- Put these normalized values into reinforce instead?

Interpret the Policy Gradient Equation

Maximum Likelihood: Push the parameters to increase probability of the observed episodes.

$$\theta \leftarrow \theta + \frac{\alpha}{N} \sum_{i=1}^{N} \nabla_\theta \sum_{t=1}^{T-1} log\, \pi_\theta(a_t^{(i)} | s_t^{(i)})$$

Policy Gradient:

$$\theta \leftarrow \theta + \frac{\alpha}{N} \sum_{i=1}^{N} G^{(i)} \nabla_\theta \sum_{t=1}^{T-1} log\, \pi_\theta(a_t^{(i)} | s_t^{(i)})$$

Push the parameters so that the above-average episodes are more likely and the below-average episodes are less likely

Interpretation from Sergey Levine Berkeley DRL course*

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \right) \left(\sum_{t=1}^{T} r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \right)$$

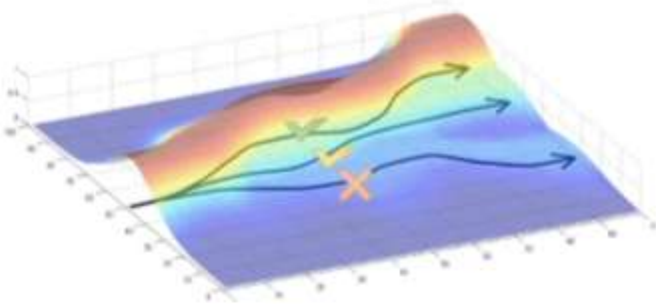
$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \underbrace{\nabla_\theta \log \pi_\theta(\tau_i)}_{\sum_{t=1}^{T} \nabla_\theta \log_\theta \pi_\theta(\mathbf{a}_{i,t} | \mathbf{s}_{i,t})} r(\tau_i)$$

maximum likelihood: $\nabla_\theta J_{\mathrm{ML}}(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \log \pi_\theta(\tau_i)$

good stuff is made more likely

bad stuff is made less likely

simply formalizes the notion of “trial and error”!



REINFORCE algorithm:

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t | \mathbf{s}_t)$ (run it on the robot)
2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i | \mathbf{s}_t^i) \right) \left(\sum_t r(\mathbf{s}_t^i, \mathbf{a}_t^i) \right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Policy Gradient issues

1. Variance issue (normalization does not suffice)
2. Choice of step-size α issue

- PPO addresses both issues.
- But PPO is a bit complicated. Let’s build up to it.

Policy Gradient: Variance problem

- Policy Gradient Theorem:

  $$\nabla_\theta\, v_\theta = E_\theta\left[\, \left(\sum_{t=1}^{T} R_t \right) \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(A_t|S_t) \,\right]$$

- Thus X = $\left(\sum_{t=1}^{T} R_t \right) \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(A_t|S_t)$ is an unbiased estimator of the gradient $\nabla_\theta\, E_\theta\left[\sum_{t=1}^{T} R_t \right]$ when episode is drawn from π_θ

- In practice X has a very high variance, i.e., it will change a lot from one episode to the next. This leads to slow convergence or no convergence at all.

Reducing Variance: Returns to Go

Let $G_t = \sum_{t'=t+1}^{T} R_{t'}$ is the "return to go" from time t.

- Can be shown $\sum_{t=0}^{T-1} G_t \nabla_\theta \log \pi_\theta(A_t|S_t)$ is also an unbiased estimator of ∇_θ E_θ [$\sum_{t=1}^{T} R_t$] and has lower variance.

- New update equation

$$\theta \leftarrow \theta + \frac{\alpha}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} G_t^{(i)} \nabla_\theta log\, \pi_\theta(a_t^{(i)}|s_t^{(i)})$$

Improved Policy Gradient Algorithm

Initialize θ in policy network

Repeat:

Use policy π_θ to obtain N episodes:

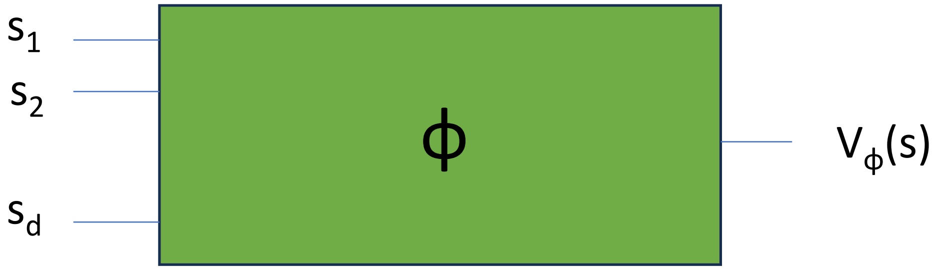
$$\tau^{(i)} = (\, s_0^{(i)}, a_0^{(i)}, r_1^{(i)}, s_1^{(i)}, a_1^{(i)}, r_2^{(i)}, \ldots, s_{T-1}^{(i)}, a_{T-1}^{(i)}, r_T^{(i)} \,), \quad \text{i=1,…,N}$$

Calculate the returns to go: $G_t^{(i)} = \sum_{t'=t+1}^{T} r_{t'}^{(i)}$ t=0,..,T-1, i=1,…,N

Update the policy:

$$\theta \leftarrow \theta + \frac{\alpha}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} G_t^{(i)} \nabla_\theta log\, \pi_\theta(a_t^{(i)} | s_t^{(i)})$$

Critic Network



- Train ϕ so that V_ϕ(s) ≈ v_{π_θ}(s) using same data from the N episodes.
- If we knew $v_{\pi_\theta}(s_t^{(i)})$ for t=1,..,T, i=1,..,N, then could use supervised learning with regression:

$$\phi \leftarrow \phi + \frac{\alpha}{N} \nabla_\phi \sum_{i=1}^{N} \sum_{t=1}^{T} \left(V_\phi(s_t^{(i)}) - v_{\pi_\theta}(s_t^{(i)}) \right)^2 \quad \text{(regression)}$$

- But we don't know $v_{\pi_\theta}(s_t^{(i)})$. However, $G_t^{(i)}$ is an unbiased estimator of $v_{\pi_\theta}(s_t^{(i)})$.
- So instead approximate $v_{\pi_\theta}(s_t^{(i)}) \approx G_t^{(i)}$:

$$\phi \leftarrow \phi + \frac{\alpha}{N} \nabla_\phi \sum_{i=1}^{N} \sum_{t=1}^{T} \left(V_\phi(s_t^{(i)}) - G_t^{(i)} \right)^2$$

- With N large and sufficient updates, should get V_ϕ(s) ≈ v_{π_θ}(s)

Reducing Variance Further*

- Policy gradient theorem: $\nabla_\theta \, \mathrm{E}_\theta [\, \sum_{t=1}^{T} R_t \,] = \mathrm{E}_\theta [\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(A_t|S_t)\, G_t]$

- Can be further shown that
  $\nabla_\theta \, \mathrm{E}_\theta [\, \sum_{t=1}^{T} R_t \,] = \mathrm{E}_\theta [\, \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(A_t|S_t) A_{\pi_\theta}(S_t, A_t)]$

  where

  $A_\pi(s,a) := q_\pi(s,a) - v_\pi(s)$ "advantage function"

- Thus $Z = \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(A_t|S_t)\, A_{\pi_\theta}(S_t, A_t)$ is also an unbiased estimator
- Also var(Z) much smaller
- But we do not know $q_{\pi_\theta}(s,a)$ and $v_{\pi_\theta}(s)$. We can approximate

$$A_{\pi_\theta}(S_t, A_t) \approx G_t - V_\phi(S_t) := \hat{A}_t$$

where G_t is the return to go, and $V_\phi(s) \approx v_{\pi_\theta}(s)$ is the critic network.

- Thus $\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(A_t|S_t)\, \hat{A}_t$ is a biased estimator but with much lower variance

Policy Gradient Algorithm w/ Advantage Estimate

Initialize θ in policy network and ϕ in critic network

Repeat:

Use policy π_θ to obtain N episodes:

$$\tau^{(i)} = (s_0^{(i)}, a_0^{(i)}, r_1^{(i)}, s_1^{(i)}, a_1^{(i)}, r_2^{(i)}, \ldots, s_{T-1}^{(i)}, a_{T-1}^{(i)}, r_T^{(i)}), \quad i=1,\ldots,N$$

Calculate the returns to go: $G_t^{(i)} = \sum_{t'=t+1}^{T} r_{t'}^{(i)}$ t=0,..,T-1, i=1,...,N

Update critic:

$$\phi \leftarrow \phi + \frac{\alpha}{N} \nabla_\phi \sum_{i=1}^{N} \sum_{t=1}^{T} (V_\phi(s_t^{(i)}) - G_t^{(i)})^2$$

Calculate advantage estimates: $\hat{A}_t^{(i)} = G_t^{(i)} - V_\phi(s_t^{(i)})$ t=0,..,T-1, i=1,...,N

Update the policy:

$$\theta \leftarrow \theta + \frac{\alpha}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \hat{A}_t^{(i)} \nabla_\theta log\, \pi_\theta(a_t^{(i)} | s_t^{(i)})$$

Typically normalize all the $\hat{A}_t^{(i)}$, t=0,..,T-1, i=1,...,N

How do we pick step size α ?

If α is too large, no convergence/learning.

If α is too small, lot’s of environment interactions for very little gain

Possible fix: Use small α, but update policy many times with same data. This is the main idea behind the algorithms TRPO and PPO.

Issue with fix: Updating current policy with data from old policy. PG theorem doesn’t apply.

TRPO/PPO solution: Only use the old data if current policy hasn’t deviated too much from old policy that generated old data.

Aside: Importance Sampling

- Suppose X and Y are two random variables taking on values in {1,2,..,n}
- X ~ p(x)   Y ~ q(y)

- Importance sampling theorem: $E[f(X)] = E[f(Y)\frac{p(Y)}{q(Y)}]$

- Thus $f(y)\frac{p(y)}{q(y)}$ with y sampled from q(.) is an unbiased estimator $E[f(X)]$

- In our case, we are going to want to re-use the episode sampled from the original policy to estimate an expectation under another policy.

- Recall $\nabla_\theta v_\theta = E_\theta[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(A_t|S_t) A_{\pi_\theta}(S_t, A_t)]$
- From importance sampling theorem:

$$E_\theta[\nabla_\theta \log \pi_\theta(A_t|S_t) A_{\pi_\theta}(S_t, A_t)] = E_{\theta'}[\nabla_\theta \log \pi_\theta(A_t|S_t) A_{\pi_\theta}(S_t, A_t) \frac{\pi_\theta(A_t|S_t)}{\pi_{\theta'}(A_t|S_t)}]$$

$$= E_{\theta'}[A_{\pi_\theta}(S_t, A_t) \frac{\nabla_\theta \pi_\theta(A_t|S_t)}{\pi_{\theta'}(A_t|S_t)}]$$

where last inequality follows from identity $\nabla_x f(x) = f(x) \nabla_x \log f(x)$

- Thus $\sum_{t=0}^{T-1} A_{\pi_\theta}(s_t, a_t) \frac{\nabla_\theta \pi_\theta(a_t|s_t)}{\pi_{\theta'}(a_t|s_t)}$ with $a_t \sim \pi_{\theta'}(.|s_t)$ is an unbiased estimator for $\nabla_\theta v_\theta$. Thus we can generate an episode from $\pi_{\theta'}$ to estimate $\nabla_\theta v_\theta$

1. Approximate $A_{\pi_\theta}(s_t, a_t)$ as $A_{\pi_{\theta'}}(s_t, a_t)$
2. Approximate $A_{\pi_{\theta'}}(s_t, a_t)$ as $\widehat{A_t'}$ where A_t' is the approximate advantage obtained with the critic using episode generated with $\pi_{\theta'}$

- Our (biased) estimator of $\nabla_\theta v_\theta$ becomes $\sum_{t=0}^{T-1} \frac{\nabla_\theta \pi_\theta(a_t|s_t)}{\pi_{\theta'}(a_t|s_t)} \widehat{A_t'}$

Proximal Policy Optimization (PPO)

- Very popular on-policy algorithm: robust, even used in original ChatGPT
- Uses critic network ϕ and advantage estimates $\hat{A}_t$
- Use current policy π_{θ_k} to generate N episodes (π_{θ_k} is $\pi_{\theta'}$ on previous slide)
- Use data from these episodes to update actor θ and critic ϕ multiple times.
- Multiple updates with same data: sample efficient!
- When updating actor, begin at $\theta = \theta_k$ and take several small gradient steps, always using the same N episodes from π_{θ_k} . Similar for critic.
- After the several small updates on θ, set $\theta_{k+1} = \theta$, and run N new episodes using θ_{k+1} .
- In this way, we can use a small learning rate α and re-use the data from π_{θ_k} for many updates.

Almost PPO:

Initialize θ_0 and ϕ

For k = 0,1,2,... :

Use policy π_{θ_k} to generate N episodes:

$\tau^{(i)} = (\, s_0^{(i)}, a_0^{(i)}, r_1^{(i)}, s_1^{(i)}, a_1^{(i)}, r_2^{(i)}, \ldots, s_{T-1}^{(i)}, a_{T-1}^{(i)}, r_T^{(i)}\,)$, i=1,...,N

Calculate the returns-to-go $G_t^{(i)}$ and the advantage estimates $\hat{A}_t^{(i)}$

Set θ_{k0}=θ_k

For j = 0,...,J-1:

$$\theta_{k(j+1)} = \theta_{kj} + \frac{\alpha}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \frac{\nabla_\theta \pi_\theta(a_t|s_t)|_{\theta=\theta_{kj}}}{\pi_{\theta'}(a_t|s_t)} \widehat{A'_t}$$

Set $\pi_{\theta_{k+1}} = \pi_{\theta_{kJ}}$

Update critic:

$$\phi \leftarrow \phi + \frac{\alpha}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\phi (V_\phi(s_t^{(i)}) - G_t^{(i)})^2$$

Issue

- Recall we approximate $A_{\pi_\theta}(s_t, a_t)$ as $A_{\pi_{\theta'}}(s_t, a_t)$
- In words, this approximation says ”the advantage function for the new policy is approximated using the the data from the original policy”.
- In Almost PPO, the original policy is π_{θ_k} and the new policy is $\pi_{\theta_{kj}}$ which is being updated in the inner loop
- If $\pi_{\theta_{kj}}$ drifts very far from π_{θ_k} the approximation becomes bad.
- PPO: when $\pi_{\theta_{kj}}$ drifts very far from π_{θ_k} use clipping to stop gradient updates.
- Clipping keeps $1 - \varepsilon \leq \frac{\pi_{\theta_{kj}}}{\pi_{\theta_k}} \leq 1 + \varepsilon$ (see Spinning Up from OpenAI)

Algorithm 1 PPO-Clip

Input: initial policy parameters θ_0, initial value function parameters ϕ_0

for $k = 0, 1, 2, \ldots$ **do**

Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.

Compute rewards-to-go $\hat{R}_t$.

Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.

Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), \;\; g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$$

typically via stochastic gradient ascent with Adam.

Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

typically via some gradient descent algorithm.

end for

importance sampling

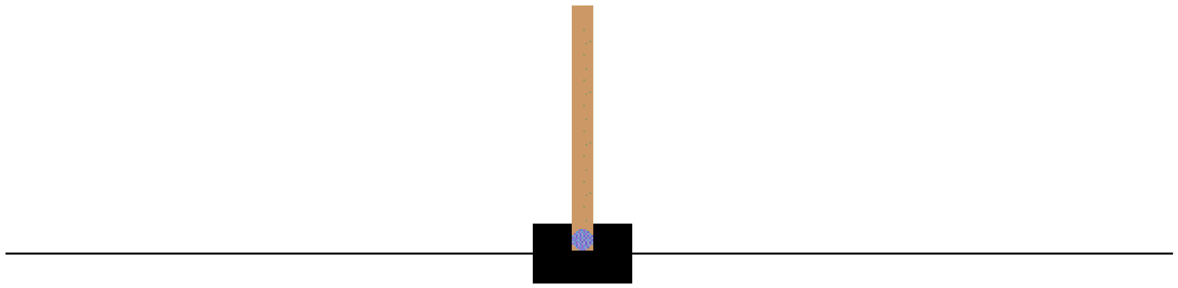
Clipping term, small typo

Take several gradient steps here

$$g(\epsilon, A) = \begin{cases} (1+\epsilon)A & A \geq 0 \\ (1-\epsilon)A & A < 0. \end{cases}$$

Homework 5

- You will apply Policy Gradient and PPO to the cartpole environment.
- Two actions: push car left, push cart right
- State space is 4D and continuous: cart position, cart velocity, pole angle, pole angular velocity
- The episode ends if any one of the following occurs:
    1. Pole Angle is greater than ±12°
    2. Cart Position is greater than ±2.4 (center of the cart reaches the edge of the display)
    3. Episode length is greater than 500
- Reward: +1 for every time step. So goal is maximize length of episode



Increasing Exploration

- Policy gradient intrinsically provides some exploration since the policy network provides a stochastic policy via the softmax.
- But often this entropy is not enough: policy network policy can become close to deterministic.
- Often add an entropy term to objective function to enhance exploration:

$$H(\pi_\theta | \mathrm{s}) := -\sum_a \pi_\theta(a|s) \log \pi_\theta(a|s)$$

- Along a single episode, we would use

$$-\frac{1}{T}\sum_{t=1}^{T}\sum_{a} \pi_\theta(a|s_t) \log \pi_\theta(a|s_t)$$

- Note that in the above formula, we use the states in the episode but not the actions.

Update with Advantage Estimate & Entropy

Update the policy:

$$\theta \leftarrow \theta + \frac{\alpha}{TN} \sum_{i=1}^{N} \sum_{t=1}^{T} \hat{A}_t^{(i)} \nabla_\theta log\, \pi_\theta(a_t^{(i)} | s_t^{(i)})$$

$$- \frac{\lambda}{TN} \sum_{i=1}^{N} \sum_{t=1}^{T} \sum_a \nabla_\theta\, \pi_\theta(a|s_t^{(i)}) log\, \pi_\theta(a|s_t^{(i)})$$