Actor-Critic Methods

Sanjiban Choudhury
Recap in 60 seconds!
Recap: Two Ingredients of RL

Exploration Exploitation

Estimate Values $Q(s, a)$

$s$  $a^1$  $a^2$  $a^3$
Curses of Function Approximation

Value Iteration: Bootstrapping

Policy Iteration: Distribution Shift

max

Figure 5 shows the car-on-hill example.
The Power of a Policy!

All we need at the end of the day is a good policy.

Black box: Try different policies and pick the best one

Gray box: Be smarter, push probability mass on actions that lead to high values

\[ \nabla_\theta J = E_{p(\xi | \theta)} \left[ \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t | s_t) Q^{\pi_\theta}(s_t, a_t) \right] \]
The Three Nightmares of Policy Optimization
Nightmare 1: High Variance

\[ \nabla_{\theta} J = E_{s \sim d^{\pi_\theta}(s), a \sim \pi_\theta(a|s)} \left[ \nabla_{\theta} \log \pi_\theta(a|s) Q^{\pi_\theta}(s, a) \right] \]

Solution: Subtract off a baseline!

\[ \nabla_{\theta} J = E_{d^{\pi_\theta}(s)} E_{\pi_\theta(a|s)} \left[ \nabla_{\theta} \log(\pi_\theta(a|s) (Q^{\pi_\theta}(s, a) - V^{\pi_\theta}(s))) \right]. \]

\[ \nabla_{\theta} J = E_{d^{\pi_\theta}(s)} E_{\pi_\theta(a|s)} \left[ \nabla_{\theta} \log(\pi_\theta(a|s) A^{\pi_\theta}(s, a)) \right]. \]
Nightmare 2: Distribution Shift

Solution: Take small steps!

$$\max_{\Delta \theta} J(\theta + \Delta \theta)$$

s.t. $$KL(\pi(\theta + \Delta \theta) || \pi(\theta)) \leq \epsilon$$
Nightmare 3:
Local Optima
The Ring of Fire
The Ring of Fire

Get’s sucked into a local optima!!
Idea: What if we had a “good reset distribution?”
Idea: What if we had a “good reset distribution?”
Idea: What if we had a “good reset distribution?”

Run REINFORCE from different start states
Idea: What if we had a “good reset distribution?”

Run REINFORCE from different start states
Idea: What if we had a “good reset distribution?”

Run REINFORCE from different start states
Solution: Use a good “reset” distribution

Choose a reset distribution $\mu(s)$ instead of start state distribution

Try your best to “cover” states the expert will visit

Justify using the PDL!
Vanilla Policy Gradient (REINFORCE)

Start with an arbitrary initial policy $\pi_\theta(a | s)$

while not converged do

Roll-out $\pi_\theta(a | s)$ to collect trajectories $D = \{s_0^i, a_0^i, r_0^i, \ldots, s_{T-1}^i, a_{T-1}^i, r_{T-1}^i\}_{i=1}^N$

Compute reward-to-go for each timestep for each trajectory

$$\hat{Q}^{\pi_\theta}(s_t^i, a_t^i) = \sum_{t'=t}^{T-1} r(s_t^i, a_t^i)$$

Compute gradient

$$\nabla_\theta J(\theta) = \frac{1}{N} \left[ \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) \hat{Q}^{\pi_\theta}(s_t^i, a_t^i) \right]$$

Update parameters $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$
Let’s apply the fixes to the nightmares!

Start with an arbitrary initial policy $\pi_{\theta}(a \mid s)$

while not converged do

Roll-out $\pi_{\theta}(a \mid s)$ to collect trajectories $D = \{s^i_0, a^i_0, r^i_0, \ldots, s^i_{T-1}, a^i_{T-1}, r^i_{T-1}\}_{i=1}^N$

Compute reward-to-go for each timestep for each trajectory $\hat{Q}_{\pi_\theta}(s^i_t, a^i_t) = \sum_{t'=t}^{T-1} r(s^i_{t'}, a^i_{t'})$

Compute gradient

$\nabla_\theta J(\theta) = \frac{1}{N} \left[ \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a^i_t \mid s^i_t) \hat{Q}_{\pi_\theta}(s^i_t, a^i_t) \right]$

Update parameters $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$
Fix #1: Subtract baseline

Start with an arbitrary initial policy $\pi_\theta(a \mid s)$

while not converged do

Roll-out $\pi_\theta(a \mid s)$ to collect trajectories $D = \{s_0^i, a_0^i, r_0^i, \ldots, s_{T-1}^i, a_{T-1}^i, r_{T-1}^i\}_{i=1}^N$

Compute reward-to-go for each timestep for each trajectory $\hat{Q}^{\pi_\theta}(s_t^i, a_t^i) = \sum_{t'=t}^{T-1} r(s_t^i, a_t^i)$

Fit value function $\hat{V}^{\pi_\theta}(s_t^i) \approx \sum_{t'=t}^{T-1} r(s_t^i, a_t^i)$

Compute advantage $\hat{A}^{\pi_\theta}(s_t^i, a_t^i) = \hat{Q}^{\pi_\theta}(s_t^i, a_t^i) - \hat{V}^{\pi_\theta}(s_t^i)$

Compute gradient $\nabla_\theta J(\theta) = \frac{1}{N} \left[ \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^i \mid s_t^i) \hat{A}^{\pi_\theta}(s_t^i, a_t^i) \right]$

Update parameters $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$
Fitting values!

**Monte-Carlo**

\[ V(s) \leftarrow V(s) + \alpha(G_t - V(s)) \]

Needs full time-horizon trajectories

**Temporal Difference**

\[ V(s) \leftarrow V(s) + \alpha(c + \gamma V(s') - V(s)) \]

Works with partial segments! 
\((s,a,r,s')\)
Actor-Critic Framework

Policy improvement of $\pi$

Estimates value functions $A^\pi$

Actor

Critic
Actor-Critic Framework

Start with an arbitrary initial policy \( \pi_\theta(a|s) \)

**while not converged do**

Roll-out \( \pi_\theta(a|s) \) to collect trajectories \( D = \{s^i, a^i, r^i, s_{+}^i\}_{i=1}^N \)

Fit value function \( \hat{V}_\pi^\theta(s^i) \) using TD, i.e. minimize \( (r^i + \gamma \hat{V}_\pi^\theta(s_{+}^i) - \hat{V}_\pi^\theta(s^i))^2 \)

Compute advantage \( \hat{A}_\pi^\theta(s^i, a^i) = r(s^i, a^i) + \gamma \hat{V}_\pi^\theta(s_{+}^i) - \hat{V}_\pi^\theta(s^i) \)

Compute gradient

\[
\nabla_\theta J(\theta) = \frac{1}{N} \left[ \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a^i_t|s^i_t) \hat{A}_\pi^\theta(s^i_t, a^i_t) \right]
\]

Update parameters

\( \theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \)
Fix #2: Take small steps

Start with an arbitrary initial policy $\pi_\theta(a \mid s)$

while not converged do

Roll-out $\pi_\theta(a \mid s)$ to collect trajectories $D = \{s^i, a^i, r^i, s^i_+\}_{i=1}^N$

Fit value function $\hat{V}_{\pi_\theta}^{\pi_\theta}(s^i)$ using TD, i.e. minimize $(r^i + \gamma \hat{V}_{\pi_\theta}(s^i_+) - \hat{V}_{\pi_\theta}(s^i))^2$

Compute advantage $\hat{A}_{\pi_\theta}^{\pi_\theta}(s^i, a^i) = r(s^i, a^i) + \gamma \hat{V}_{\pi_\theta}(s^i_+) - \hat{V}_{\pi_\theta}(s^i)$

Compute gradient

$$\nabla_\theta J(\theta) = \frac{1}{N} \left[ \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a^i_t \mid s^i_t) \hat{A}_{\pi_\theta}^{\pi_\theta}(s^i_t, a^i_t) \right]$$

s.t. $KL(\pi(\theta + \Delta \theta) \mid \mid \pi(\theta)) \leq \epsilon$

Update parameters $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

How??
Natural Gradient Descent (rediscovered as TRPO)

Start with an arbitrary initial policy $\pi_\theta(a | s)$

\[ \text{while not converged do} \]

Roll-out $\pi_\theta(a | s)$ to collect trajectories $D = \{s^i, a^i, r^i, s_{+}^i\}_{i=1}^N$

Fit value function $\hat{V}_{\pi_\theta}(s^i)$ using TD, i.e. minimize $(r^i + \gamma \hat{V}_{\pi_\theta}(s_{+}^i) - \hat{V}_{\pi_\theta}(s^i))^2$

Compute advantage $\hat{A}_{\pi_\theta}(s^i, a^i) = r(s^i, a^i) + \gamma \hat{V}_{\pi_\theta}(s_{+}^i) - \hat{V}_{\pi_\theta}(s^i)$

Compute gradient
\[
\nabla_\theta J(\theta) = \frac{1}{N} \left[ \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a^i_t | s^i_t) \hat{A}_{\pi_\theta}(s^i_t, a^i_t) \right]
\]

Update parameters $\theta \leftarrow \theta + \alpha G(\theta)^{-1} \nabla_\theta J(\theta)$

s.t. $KL(\pi(\theta + \Delta \theta) \| \pi(\theta)) \leq c$

$\approx \Delta \theta^T G(\theta) \Delta \theta \leq c$

$G(\theta)$ is Fischer Information Matrix

$G(\theta) = \mathbb{E}_{\pi_\theta} \left[ \nabla_\theta \log \pi_\theta \nabla_\theta \log \pi_\theta^T \right]$
Natural Gradient Descent (rediscovered as TRPO!)

Start with an arbitrary initial policy \( \pi_\theta(a \mid s) \)

while not converged do

Roll-out \( \pi_\theta(a \mid s) \) to collect trajectories \( D = \{s^i, a^i, r^i, s^i_+\}_{i=1}^N \)

Fit value function \( \hat{V}^{\pi_\theta}(s^i) \) using TD, i.e. minimize \((r^i + \gamma \hat{V}^{\pi_\theta}(s^i_+) - \hat{V}^{\pi_\theta}(s^i))^2\)

Compute advantage \( \hat{A}^{\pi_\theta}(s^i, a^i) = r(s^i, a^i) + \gamma \hat{V}^{\pi_\theta}(s^i_+) - \hat{V}^{\pi_\theta}(s^i) \)

Compute gradient

\[
\nabla_\theta J(\theta) = \frac{1}{N} \left[ \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a^i_t \mid s^i_t) \hat{A}^{\pi_\theta}(s^i_t, a^i_t) \right]
\]

Update parameters \( \theta \leftarrow \theta + \alpha G(\theta)^{-1} \nabla_\theta J(\theta) \)

Don’t directly compute the inverse, use conjugate gradient to solve \( G(\theta)x = \nabla_\theta J(\theta) \)

s.t. \( KL(\pi(\theta + \Delta \theta) \parallel \pi(\theta)) \leq c \)

\( \approx \Delta \theta^T G(\theta) \Delta \theta \leq c \)

\( G(\theta) \) is Fischer Information Matrix

\( G(\theta) = \mathbb{E}_{\pi_\theta} \left[ \nabla_\theta \log \pi_\theta \nabla_\theta \log \pi_\theta^T \right] \)
Proximal Policy Optimization (PPO)

Computing Fischer Information matrix is expensive and slow!

Idea: Instead of taking small steps, change the loss function so there is no benefit in taking large steps!
Proximal Policy Optimization (PPO)

Computing Fischer Information matrix is expensive and slow!

Idea: Instead of taking small steps, **change the loss function** so there is no benefit in taking large steps!

Instead of defining gradient, we will define a surrogate loss function
(Lets say we are at iteration $k$)

$$
\mathcal{L}(\theta) = \mathbb{E}_{s,a \sim \pi_{\theta_k}} \left[ \frac{\pi_\theta A_{\pi_{\theta_k}}(s, a)}{\pi_{\theta_k}} \right]
$$
Proximal Policy Optimization (PPO)

Computing Fischer Information matrix is expensive and slow!

Idea: Instead of taking small steps, change the loss function so there is no benefit in taking large steps!

Clip the loss if the policy $\pi_\theta$ deviates too much from $\pi_{\theta_k}$

$$\mathcal{L}(\theta) = \mathbb{E}_{s,a \sim \pi_{\theta_k}} \left[ \min \left( \frac{\pi_\theta}{\pi_{\theta_k}} A^{\pi_{\theta_k}}(s, a), \text{clip} \left( \frac{\pi_\theta}{\pi_{\theta_k}}, 1 - \epsilon, 1 + \epsilon \right) A^{\pi_{\theta_k}}(s, a) \right) \right]$$
Fix #3: Use a reset distribution

Start with an arbitrary initial policy $\pi_\theta(a | s)$

\[
\text{while not converged do}
\]

Roll-out $\pi_\theta(a | s)$ to collect trajectories $D = \{s^i, a^i, r^i, s^i_+\}_{i=1}^N$

Fit value function $\hat{V}^{\pi_\theta}(s^i)$ using TD, i.e. minimize $(r^i + \gamma \hat{V}^{\pi_\theta}(s^i_+) - \hat{V}^{\pi_\theta}(s^i))^2$

Compute advantage $\hat{A}^{\pi_\theta}(s^i, a^i) = r(s^i, a^i) + \gamma \hat{V}^{\pi_\theta}(s^i_+) - \hat{V}^{\pi_\theta}(s^i)$

Compute gradient
\[
\nabla_\theta J(\theta) = \frac{1}{N} \left[ \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a^i_t | s^i_t) \hat{A}^{\pi_\theta}(s^i_t, a^i_t) \right]
\]

s.t. $KL(\pi(\theta + \Delta \theta) || \pi(\theta)) \leq \epsilon$

Update parameters $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Instead of rolling out from the start state, rollout from states expert visits
How do we make Actor-Critic more robust to randomness of the environment?
We never see the actual environment in RL

Credit: Ben Eyesenbach
We want our policy to be robust against all possible environments that can explain the data.

\[
\max_{\pi} \min_{\hat{p} \in \hat{P}, \hat{r} \in \hat{R}} \mathbb{E}_{\hat{p}(s_{t+1} | s_t, a_t), \pi(a_t | s_t)} \left[ \sum_{t=1}^{T} \hat{r}(s_t, a_t) \right].
\]
Solution: Use Maximum Entropy RL!

\[ J_{\text{MaxEnt}}(\pi; p, r) \triangleq \mathbb{E}_{a_t \sim \pi(a_t | s_t), s_{t+1} \sim p(s_{t+1} | s_t, a_t)} \left[ \sum_{t=1}^{T} r(s_t, a_t) + \alpha \mathcal{H}_{\pi}(a_t | s_t) \right] \]

Intuition: There are many policies that can achieve the same cumulative rewards. MaxEntRL keeps alive all of those policies. Learns many different ways to solve the same task.
Solution: Use Maximum Entropy RL!

Trained and evaluated without the obstacle:

Trained without the obstacle, but evaluated with the obstacle:
"Soft" Actor Critic

Actor

\[ \pi_{\text{new}} = \arg \min_{\pi' \in \Pi} D_{\text{KL}} \left( \pi' (\cdot | s_t) \left\| \frac{\exp (Q_{\text{old}} (s_t, \cdot))}{Z_{\text{old}} (s_t)} \right. \right) \]

Critic

\[ T^\pi Q(s_t, a_t) \triangleq r(s_t, a_t) + \gamma E_{s_{t+1} \sim p} [V(s_{t+1})], \]

where

\[ V(s_t) = E_{a_t \sim \pi} [Q(s_t, a_t) - \log \pi(a_t | s_t)] \]
“Soft” Actor Critic

54 min

5x