Actor-Critic Methods

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

Exploration Exploitation

Estimate Values $Q(s, a)$
Curses of Function Approximation

Value Iteration: Bootstrapping

Policy Iteration: Distribution Shift

Upper half of state is BAD

Lower half of state is GOOD
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 rewards

\[
\nabla_\theta J = E_{p(\xi|\theta)} \left[ \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) Q^{\pi_\theta}(s_t, a_t) \right]
\]
Wait ... how did we get around the distribution shift problems?
The Policy Gradient Theorem

\[ \nabla_\theta J = E_{p(\xi|\theta)} \left[ \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) Q^{\pi_\theta}(s_t, a_t) \right] \]

Is this gradient the best descent direction?
What would gradient descent do here?

How can we get it to converge better?
Activity!
Think-Pair-Share

Think (30 sec): How can we get gradient descent to converge better in the example below?

Pair: Find a partner

Share (45 sec): Partners exchange ideas
Gradient Descent as Steepest Descent

Gradient Descent is simply Steepest Descent with L2 norm

$$\max_{\Delta \theta} J(\theta + \Delta \theta) \quad s.t. \quad \|\Delta \theta\| \leq \epsilon \quad \rightarrow \quad \Delta \theta = \nabla_\theta J(\theta)$$

What would update look like for another norm?

$$\max_{\Delta \theta} J(\theta + \Delta \theta) \quad s.t. \quad \Delta \theta^\top G(\theta) \Delta \theta \leq \epsilon \quad \rightarrow \quad \Delta \theta = \frac{1}{2\lambda} G^{-1}(\theta) \nabla_\theta J(\theta)$$
What’s a good norm for distributions?
What is a good norm for distributions?

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

s.t. $KL(P(\theta + \Delta \theta) || P(\theta)) \leq \epsilon$
What is a good norm for distributions?

\[
\max_{\Delta \theta} J(\theta + \Delta \theta)
\]

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

s.t. \( \Delta \theta^T G(\theta) \Delta \theta \leq \epsilon \)

Fischer Information Matrix

\[
G(\theta) = E_{p_\theta} \left[ \nabla_\theta \log(p_\theta) \nabla_\theta \log(p_\theta)^\top \right]
\]
“Natural” Gradient Descent

Start with an arbitrary initial policy $\pi_\theta$

while not converged do

Run simulator with $\pi_\theta$ to collect $\{\xi^{(i)}\}_{i=1}^N$

Compute estimated gradient

$$\hat{\nabla}_\theta J = \frac{1}{N} \sum_{i=1}^N \left[ \left( \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta \left( a^{(i)}_t \mid s^{(i)}_t \right) \right) R(\xi^{(i)}) \right]$$

$$\hat{G}(\theta) = \frac{1}{N} \sum_{i=1}^N \left[ \nabla_\theta \log \pi_\theta(a_i \mid s_i) \nabla_\theta \log \pi_\theta(a_i \mid s_i)^T \right]$$

Update parameters $\theta \leftarrow \theta + \alpha \hat{G}^{-1}(\theta) \hat{\nabla}_\theta J$.

return $\pi_\theta$

Modern variants are TRPO, PPO, etc
But does this work on *real robots*?
Initially, we teach a rudimentary stroke by supervised learning as can be seen in Figure 3 (b); however, it fails to reproduce the behavior as shown in (c); subsequently, we improve the performance using the episodic Natural Actor-Critic which yields the performance shown in (a) and the behavior in (d). After approximately 200-300 trials, the ball can be hit properly by the robot.
Consider the following single roll-out

Return = -100

What would the gradient at $s_t$ be?

Is this a good roll-out or a bad roll out?
It depends on other trajectories!

Return = -100

Return = -1000

How can we incorporate *relative* information?
Problem: High Variance

\[ \nabla_\theta J = E_{p(\xi|\theta)} \left[ \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) Q^{\pi_\theta}(s_t, a_t) \right] \]

One of the reasons for the high variance is that the algorithm does not know how well the trajectories perform compared to other trajectories.
Solution: Subtract a baseline!

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

Prove this does not change the gradient!

\[ = E_{d^{\pi_\theta}(s)} E_{\pi_\theta(a|s)} \left[ \nabla_\theta \log(\pi_\theta(a|s))A^{\pi_\theta}(s,a) \right] \]
Recap (again) in 60 seconds!

1. Local Optima: Use Exploration Distribution

2. Distribution Shift: *Natural* Gradient Descent

3. High Variance: Subtract baseline
If we are estimating values ... can we bring back MC and TD?

<table>
<thead>
<tr>
<th>Monte-Carlo</th>
<th>Temporal Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V(s) \leftarrow V(s) + \alpha (G_t - V(s))$</td>
<td>$V(s) \leftarrow V(s) + \alpha (c + \gamma V(s') - V(s))$</td>
</tr>
<tr>
<td>Zero Bias</td>
<td>Can have bias</td>
</tr>
<tr>
<td>High Variance</td>
<td>Low Variance</td>
</tr>
<tr>
<td>Always convergence (Just have to wait till heat death of the universe)</td>
<td>May not converge if using function approximation</td>
</tr>
</tbody>
</table>
Actor-Critic Algorithms

Actor

Policy improvement of $\pi$

Natural Gradient Descent

Critic

Estimates value functions $Q^\pi_\phi/V^\pi_\phi/A^\pi_\phi$

TD, MC
The General Actor Critic Framework

batch actor-critic algorithm:
1. sample \( \{s_i, a_i\} \) from \( \pi_\theta(a|s) \) (run it on the robot)
2. fit \( \hat{V}_\phi^\pi(s) \) to sampled reward sums \( (TD, MC) \)
3. evaluate \( \hat{A}_\pi^\pi(s_i, a_i) = r(s_i, a_i) + \gamma \hat{V}_\phi^\pi(s'_i) - \hat{V}_\phi^\pi(s_i) \)
4. \( \nabla_\theta J(\theta) \approx \sum_i \nabla_\theta \log \pi_\theta(a_i|s_i) \hat{A}_\pi^\pi(s_i, a_i) \)
5. \( \theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \)

Credit: Sergey Levine
Practical Issues and Fixes
Problem 1: How do we make Actor Critic off-policy?

batch actor-critic algorithm:

1. **sample** \( \{s_i, a_i\} \) from \( \pi_\theta(a|s) \) (run it on the robot)
2. fit \( \hat{V}_\phi^\pi(s) \) to sampled reward sums
3. evaluate \( \hat{A}^\pi(s_i, a_i) = r(s_i, a_i) + \gamma \hat{V}_\phi^\pi(s'_i) - \hat{V}_\phi^\pi(s_i) \)
4. \( \nabla_\theta J(\theta) \approx \sum_i \nabla_\theta \log \pi_\theta(a_i|s_i) \hat{A}^\pi(s_i, a_i) \)
5. \( \theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \)
Problem 1: How do we make Actor Critic off-policy?

get \((s, a, s', r)\) → 
update \(\theta\) → 
get \((s, a, s', r)\) → 
update \(\theta\) → 

replay buffer

transitions that we saw in prior time steps

Credit: Sergey Levine
Solution: Carefully assign credit to correct actions!

1. take action $a \sim \pi_\theta(a|s)$, get $(s, a, s', r)$, store in $\mathcal{R}$
2. sample a batch $\{s_i, a_i, r_i, s'_i\}$ from buffer $\mathcal{R}$
3. update $\hat{Q}_\phi^\pi$ using targets $y_i = r_i + \gamma\hat{Q}_\phi^\pi(s'_i, a'_i)$ for each $s_i, a_i$
4. $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i \nabla_\theta \log \pi_\theta(a_i^\pi|s_i)^\pi(s_i, a_i^\pi)$ where $a_i^\pi \sim \pi_\theta(a|s_i)$
5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$
Problem 2: How can we be robust to changes in the environment?
Problem 2: How can we be robust to changes in the environment?

\[
\max_{\pi} \min_{\tilde{p} \in \tilde{P}, \tilde{r} \in \tilde{R}} \mathbb{E}_{\tilde{p}(s_{t+1} \mid s_t, a_t), \pi(a_t \mid s_t)} \left[ \sum_{t=1}^{T} \tilde{r}(s_t, a_t) \right].
\]
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:
\[ \pi_{\text{new}} = \arg \min_{\pi' \in \Pi} D_{KL} \left( \pi' (\cdot | s_t) \mid \frac{\exp (Q^{\pi_{\text{old}}}(s_t, \cdot))}{Z^{\pi_{\text{old}}}(s_t)} \right) \]

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

where

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

54 min

5x
Algorithm 1 Advantage-Weighted Regression

1: \( \pi_1 \leftarrow \text{random policy} \)
2: \( \mathcal{D} \leftarrow \emptyset \)
3: for iteration \( k = 1, \ldots, k_{\text{max}} \) do
4: \hspace{1em} add trajectories \( \{\tau_i\} \) sampled via \( \pi_k \) to \( \mathcal{D} \)
5: \hspace{1em} \( V_k^\mathcal{D} \leftarrow \arg \min_V \mathbb{E}_{s,a \sim \mathcal{D}} \left[ \| R_{s,a}^\mathcal{D} - V(s) \|^2 \right] \)
6: \hspace{1em} \( \pi_{k+1} \leftarrow \arg \max_\pi \mathbb{E}_{s,a \sim \mathcal{D}} \left[ \log \pi(a|s) \exp \left( \frac{1}{\beta} (R_{s,a}^\mathcal{D} - V_k^\mathcal{D}(s)) \right) \right] \)
7: end for

Peng et al, 2019
“Natural” Gradient Descent

Start with an arbitrary initial policy $\pi_0$
while not converged do
Run simulator with $\pi_\theta$ to collect $(s^{(i)}, a^{(i)})_{i=1}^N$
Compute estimated gradient

$$\nabla_\theta J = \frac{1}{N} \sum_{i=1}^N \left[ \left( \sum_{a} \nabla_a \log \pi_\theta (a|s^{(i)}) \right) R(s^{(i)}) \right]$$

$$G(\theta) = \frac{1}{N} \sum_{i=1}^N \left[ \nabla_\theta \log \pi_\theta (a|s^{(i)}) \nabla_\theta \log \pi_\theta (a|s^{(i)})^T \right]$$

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

Modern variants are TRPO, PPO, etc

Recap (again) in 60 seconds!

1. Local Optima: Use Exploration Distribution
2. Distribution Shift: Natural Gradient Descent
3. High Variance: Subtract baseline

The General Actor Critic Framework

batch actor-critic algorithm:

1. sample $\{s_i, a_i\}$ from $\pi_\theta(a|s)$ (run it on the robot)
2. fit $\hat{V}_\theta^\pi(s)$ to sampled reward sums (TD, MC)
3. evaluate $\hat{A}_\theta^\pi(s_i, a_i) = r(s_i, a_i) + \gamma \hat{V}_\theta^\pi(s'_i) - \hat{V}_\theta^\pi(s_i)$
4. $\nabla_\theta J(\theta) \approx \sum_i \nabla_\theta \log \pi_\theta (a_i|s_i) \hat{A}_\theta^\pi(s_i, a_i)$
5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Credit: Sergey Levine