Deep Reinforcement Learning (Sp25)

Instructor: Dr. Mohammad Hossein Rohban

Summary of Lecture 7: Advanced Methods

Summarized By: Amirhossein Asadi

- After discussing techniques like the causality trick (reward-to-go), discount factor, and baseline subtraction, we are still seeking ways to further **reduce variance** in the policy gradient equation:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\, R_t\right]$$

- To reduce the variance in policy gradient methods, we estimate the **Q-value** ($Q(s_t, a_t)$). By estimating Q, we can approximate the gradient that updates the policy in a direction that maximizes the expected reward. This approach allows us to improve the policy more effectively. This approach is a key idea in **Actor-Critic** methods.

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

$\hat{Q}_{i,t}$: An estimate of the expected reward if we take action $\mathbf{a}_{i,t}$ in state $\mathbf{s}_{i,t}$.

- In the **Actor-Critic** method, there are two main components:

Critic:

The Critic is responsible for estimating the value function (either $Q(s_t, a_t)$ or $V(s_t)$).

It uses environmental data to estimate the value of states or state-action pairs.

The estimated Q or V is used as feedback to improve the policy.

Actor:

The Actor is responsible for updating the policy (π_θ).

It uses gradients provided by the Critic to update the policy parameters (θ).

The goal of the Actor is to improve the policy in a direction that maximizes the expected reward.

- To further reduce variance in policy gradient methods, we can also add a **baseline**. A suitable choice for the baseline is the average return over N trajectories. This baseline can be effectively represented using the **Q-function**, which estimates the expected cumulative reward for a given state-action pair.

$$b_t = \frac{1}{N}\sum_i Q(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \quad \rightarrow \quad V(\mathbf{s}_t) = \mathbb{E}_{\mathbf{a}_t\sim\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}[Q(\mathbf{s}_t, \mathbf{a}_t)]$$

- To estimate both the Q-function and the value function V, we would typically need **two separate critics**. However, maintaining and training two critics is computationally expensive and undesirable. To address this, we combine the two into a **single critic**, which simplifies the learning process while retaining the necessary functionality.
- We now define the **advantage function**, a key concept in **RL**. The advantage function $A^\pi(\mathbf{s}_t, \mathbf{a}_t)$ measures how much better a specific action $\mathbf{a}_t$ is compared to the average action in a given state $\mathbf{s}_t$. It is defined as the difference between the Q-function (the expected return for taking action $\mathbf{a}_t$ in state $\mathbf{s}_t$) and the value function $V^\pi(\mathbf{s}_t)$ (the expected return for being in state $\mathbf{s}_t$):

$$A^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = \overbrace{Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t)}^{\text{Action-Value Function}} - \overbrace{V^{\pi}(\mathbf{s}_t)}^{\text{State-Value Function}}$$

The advantage function provides a **baseline** for evaluating actions, reducing variance in policy gradient methods. By using $A^{\pi}(\mathbf{s}_t, \mathbf{a}_t)$, we can more effectively guide the policy toward actions that yield higher rewards.

- With the **advantage function** $A^{\pi}(s_{i,t}, a_{i,t})$, the policy gradient can now be expressed in a more efficient form. The policy gradient update rule becomes:

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_{\theta} \log \pi_{\theta}(a_{i,t}|s_{i,t})\, A^{\pi}(s_{i,t}, a_{i,t})$$

The better the estimate of the advantage function, the **lower the variance** in the policy gradient updates. This leads to more stable and efficient training of the policy.

- The **Batch Actor-Critic Algorithm** is a **RL** method that combines policy-based (Actor) and value-based (Critic) approaches. It samples a batch of trajectories, fits a value function, computes advantages, and updates the policy using gradient ascent.
- The **Online Actor-Critic Algorithm** Updates the policy π_{θ} and value function V_{ϕ}^{π} after each step and combines **Actor** (policy) and **Critic** (value function). It combines:
  - **Actor**: Policy $\pi_{\theta}(a|s)$.
  - **Critic**: Value function $V_{\phi}^{\pi}(s)$.
- **Proximal Policy Optimization:** In previous methods, learning rate can cause the policy to change drastically, potentially ruining the data. To prevent this, we need to ensure that the new policy does not deviate too much from the old one. **PPO** ensures that the new policy does not deviate too much from the old policy during updates. This prevents unstable training and catastrophic failures caused by large policy changes.

$$D_{KL}(\pi_{\theta'}(.|s) \;||\; \pi_{\theta}(.|s)) \leq \epsilon$$

It uses a clipped objective or KL-divergence constraint to limit updates, making training more stable and reliable.

- In **PPO**, the clipping mechanism limits the policy update ratio $r(\theta)$ to $[1-\epsilon, 1+\epsilon]$. This ensures the new policy does not deviate too much from the old one, preventing unstable updates.