Deep Reinforcement Learning (Sp25)

Instructor: Dr. Mohammad Hossein Rohban

Summary of Lecture 6: Advanced Policy Gradients

Summarized By: Arshia Gharooni

Policy Gradient Methods

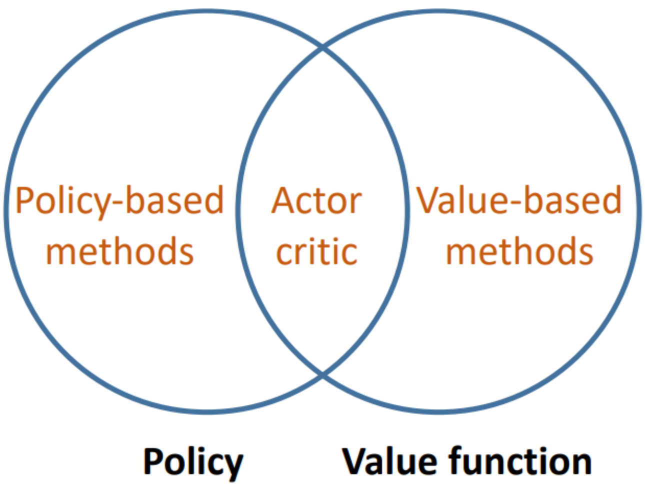
- Policy gradient methods aim to learn the policy directly without using a value function.
- The objective function is the expected sum of rewards from an initial state, denoted as $J(\theta) = E[\sum R_t]$, where θ represents the policy parameters.
- The policy is often parameterized using a neural network, where the inputs are states, and the outputs are probability distributions over actions.
- The goal is to optimize $J(\theta)$ by adjusting θ using gradient ascent.
- Monte Carlo methods are used to approximate the expectation in the objective function.
- **Log trick**: A mathematical transformation is applied to compute the policy gradient:

$$\nabla_\theta J(\theta) = E\left[\sum \nabla_\theta \log \pi_\theta(a_t|s_t) R_t\right] \tag{1}$$

  where the log-derivative trick simplifies the gradient computation.
- The **REINFORCE algorithm** follows this approach by sampling trajectories, computing rewards, and updating the policy parameters using stochastic gradient ascent.

Issues with Policy Gradient Methods

- **High variance**: The policy gradient estimate has high variance, making convergence slow and unstable.
- **Sample inefficiency**: Since each update discards old samples, a large number of episodes is needed to achieve good performance.
- **Imitation Learning vs. Policy Gradient**:
  - **Imitation learning (Behavioral Cloning)** trains a policy to mimic expert demonstrations using supervised learning.
  - **Policy Gradient** optimizes policy based on rewards rather than imitation.
  - A major difference is that policy gradients involve weighting by reward, while imitation learning lacks such weighting.

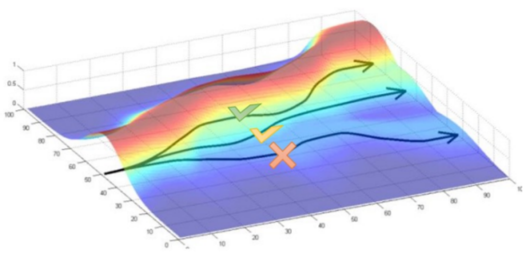


Model-Free RL Overview

- **Value-based methods**: Learn value functions (e.g., Q-learning, Deep Q-Networks) but do not learn policies directly.
- **Policy-based methods**: Learn policies directly without using value functions.
- **Actor-Critic methods**: Combine value-based and policy-based methods by learning both a policy (actor) and a value function (critic).

Policy Gradient Intuition

- The fundamental idea is simple:
  - Actions that lead to higher rewards should become more likely.
  - Actions that lead to lower rewards should become less likely.
  - This formalizes the concept of "trial and error" learning.



Bias and Variance in Policy Gradient

- Policy gradient estimation is **unbiased**, but it suffers from **high variance**, making optimization inefficient.
- The main sources of high variance are:
  - Monte Carlo return estimation.
  - Long-term dependencies on past actions and states.

Reducing Variance in Policy Gradient Estimation

Several techniques can be used to reduce variance while preserving unbiasedness:

1. Causality Trick ("Reward-to-Go")

- Instead of using the total return from the beginning of the trajectory, we only use future rewards:

$$\nabla_\theta J(\theta) = E\left[\sum_{l=0}^{T} \nabla_\theta \log \pi_\theta(a_l|s_l) \sum_{t'=l}^{T} R_{t'}\right] \tag{2}$$

- This reduces variance because an action taken at time t is only responsible for rewards obtained after t, not before.
- Introduces a discount factor γ (typically around 0.99) to reduce the impact of distant rewards:

$$\nabla_\theta J(\theta) = E\left[\sum_{l=0}^{T} \nabla_\theta \log \pi_\theta(a_l|s_l) \sum_{t'=l}^{T} \gamma^{t'-l} R_{t'}\right] \tag{3}$$

RIML

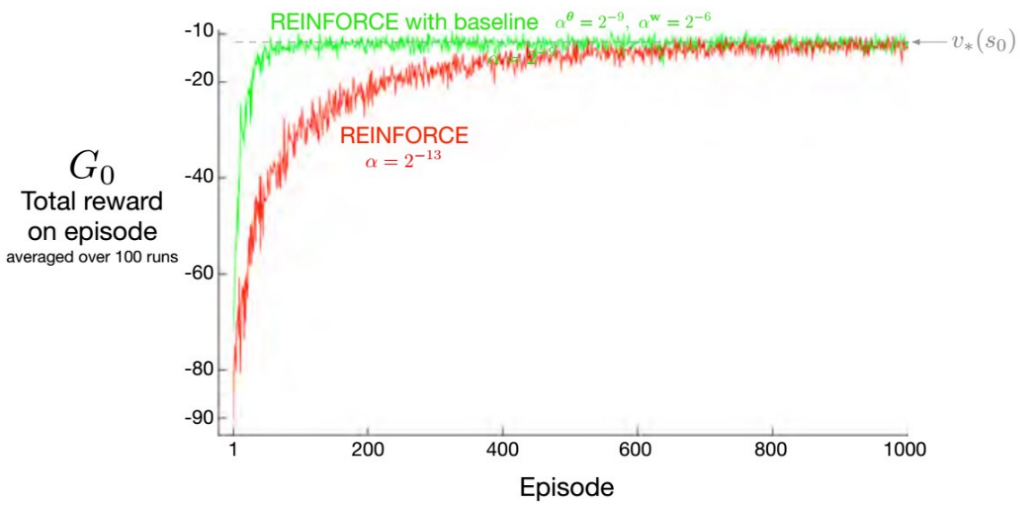
2. Baseline Subtraction

- The gradient formula can be modified by subtracting a baseline $b(s_t)$:

$$\nabla_\theta J(\theta) = E\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)(R_t - b(s_t))\right] \tag{4}$$

- If $b(s_t)$ is chosen properly, it does not introduce bias but can reduce variance significantly.
- A common choice is the **value function** $V(s_t)$, leading to the **Advantage Function**:

$$A(s_t, a_t) = Q(s_t, a_t) - V(s_t) \tag{5}$$



4. Actor-Critic Methods

- **Actor** updates the policy based on the advantage function.
- **Critic** estimates the value function to compute the advantage.
- **Advantage Estimation**:

$$A(s_t, a_t) = R_t - V(s_t) \tag{6}$$

- This method balances bias and variance better than pure policy gradient or value-based methods.