17-740	Spring 2025
Algorithmic Foundations of Interactive Learning	

Lecture 11: Policy Gradients

Lecturer: Steven Wu
Scribe: Jiahao Zhang, Naveen Raman, Riku Arakawa

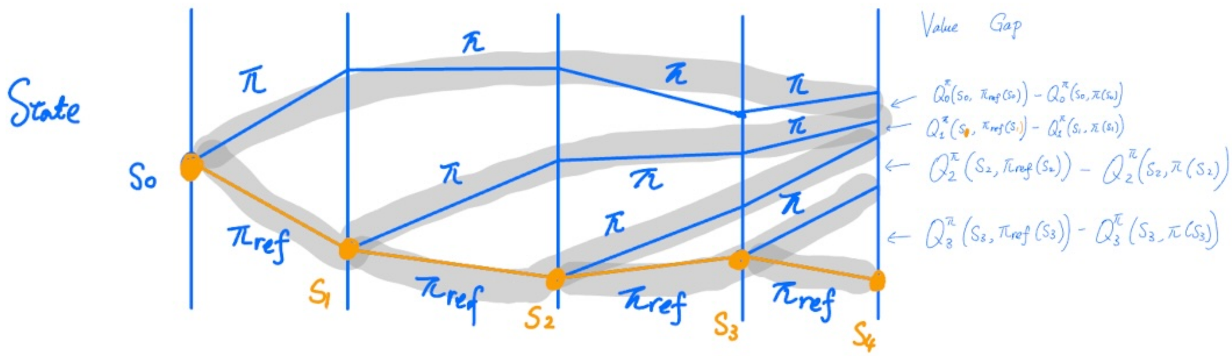

Figure 11.1: A visualization of the Performance Difference Lemma

11.1 Recap: PDL and PSDP

We first provide some more intuition for the *performance difference lemma* (PDL), which bounds the difference between $J(\pi_{\text{ref}})$ and $J(\pi)$. To simplify our reasoning, we will assume that both policies and transitions are deterministic. As shown in Figure 11.1, the reference policy π_{refs} will then visit a sequence of states $s_0, s_1 \ldots s_{H-1}$.

Suppose we are at time step h and at the expert's state S_h. We can compare two trajectories:

1. Taking action $\pi_{\text{ref}}(S_h)$ and then following π for the remaining steps.

2. Following π starting at this step and all future steps.

The difference in their values can be written as:

$$\delta_h = Q_h^\pi(S_h, \pi_{\text{ref}}(S_h)) - Q_h^\pi(S_h, \pi(S_h))$$

Now we do this comparison at state s_0, then δ_0 measures the value gap between the top two trajectories in Figure 11.1. Similarly, δ_1 measures the value gap between the second and third trajectories on top. Observe that the gap we are interested in $J(\pi_{\text{ref}}) - J(\pi)$ is precisely the value gap between the top trajectory (fully in blue) and bottom trajectory (fully in yellow). By telescoping, we can write

$$J(\pi_{\text{ref}}) - J(\pi) = \sum_h \delta_h$$

This is precisely the deterministic version of PDL.

Now with this intuition and picture of PDL in mind, we can also gain an intuitive understanding of the *policy search by dynamic programming* (PSDP) algorithm. The algorithm operates on the assumption that we are given a baseline distribution μ_h at every step h. PSDP essentially ensures the quantity δ_h is small over the distribution μ_h of state s_h. PSDP achieves this via backward induction: at every step h, it optimizes the policy π_h over the state distribution μ_h given the learned policies $\pi_{h+1}, \ldots, \pi_{H-1}$ at later steps. (Note that this then becomes a one-step decision-making problem, or equivalently a classification problem.) Concretely, it first computes the action value for each action a at each sampled state $s_h \sim \mu_h$:

$$Q_h^\pi(s_h, a) = r(s_h, a) + \mathbb{E}_{s' \sim P(\cdot|s_h,a)}[V_{h+1}^\pi(s')]$$

where $V_{h+1}^\pi(s')$ is the estimated value function at the next time step. Then, PSDP updates π_h to select action a that maximizes the estimated $Q_h^\pi(s_h, a)$.

Suppose the algorithm achieves ϵ error for each step over the distribution μ_h–that is,

$$\mathbb{E}_{s_h \sim \mu_h}\left[Q^\pi(s_h, \pi(s_h))\right] \geq \max_{\pi'} \mathbb{E}_{s_h \sim \mu_h}\left[Q^\pi(s_h, \pi'(s_h))\right] - \epsilon$$

By change of measure from the baseline distribution to the state distribution visited by the reference policy, we have

$$\mathbb{E}_{s_h \sim d_h^{\pi_{\text{ref}}}}\left[Q^\pi(s_h, \pi(s_h))\right] \geq \max_{\pi'} \mathbb{E}_{s_h \sim d_h^{\pi_{\text{ref}}}}\left[Q^\pi(s_h, \pi'(s_h))\right] - \epsilon \left\| \frac{d_h^{\pi_{ref}}}{\mu_h} \right\|_\infty$$

By PDL, we can bound the performance difference as

$$J(\pi_{\text{ref}}) - J(\pi) \leq \sum_h \epsilon \left\| \frac{d_h^{\pi_{ref}}}{\mu_h} \right\|_\infty$$

11.2 Policy Gradients

Another paradigm for reinforcement learning is to directly optimize the policy, known as *Policy Gradients.* In this approach, we parameterize the policy as

$$\pi_\theta(a \mid s) = \pi(a \mid s; \theta),$$

where θ denotes the policy parameters.

A trajectory (or episode) is defined as:

$$\tau = (s_0, a_0, s_1, a_1, \ldots, s_{H-1}, a_{H-1}),$$

and the performance objective is given by:

$$J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{h=0}^{H-1} r_h\right] = \mathbb{E}_{\tau\sim\pi_\theta}\left[R(\tau)\right],$$

where $R(\tau)$ denotes the return of trajectory τ.

11.2.1 High-level Idea

The fundamental idea behind policy gradient methods is to update the policy parameters using gradient ascent. In its simplest form, the update rule is:

$$\theta_{t+1} = \theta_t + \eta\, \nabla_\theta J(\pi_{\theta_t}),$$

where η is the learning rate (step-size). In order to apply gradient ascent, it is necessary to make $J(\pi_\theta)$ differentiable with respect to θ.

There are several ways to parameterize the policy:

1. **Tabular Case:**
   When the state and action spaces are small enough to be represented in a table, the policy can be defined as:
   $$\pi_\theta(a \mid s) = \frac{\exp(\theta_{s,a})}{\sum_{a'} \exp(\theta_{s,a'})}.$$

2. **Log-Linear Policies:**
   In this setting, a feature vector $\phi_{s,a}$ is associated with each state-action pair (s, a). The policy is then defined as:
   $$\pi_\theta(a \mid s) = \frac{\exp(\langle\theta, \phi_{s,a}\rangle)}{\sum_{a'} \exp(\langle\theta, \phi_{s,a'}\rangle)}.$$

3. **Neural Softmax Policies:**
   For more complex scenarios, a neural network can be used to parameterize the policy:
   $$\pi_\theta(a \mid s) = \frac{\exp(f_\theta(s, a))}{\sum_{a'} \exp(f_\theta(s, a'))},$$
   where $f_\theta(s, a)$ is a function approximated by a neural network.

11.2.2 Warm Up

Consider a simplified objective function defined as:

$$J(\theta) = \mathbb{E}_{x \sim P_\theta}\left[f(x)\right].$$

Taking the gradient with respect to θ, we have:

$$\nabla_\theta J(\theta) = \nabla_\theta \sum_x P_\theta(x) f(x) = \sum_x \nabla_\theta P_\theta(x)\, f(x).$$

Using the identity

$$\nabla_\theta P_\theta(x) = P_\theta(x) \nabla_\theta \ln P_\theta(x),$$

we obtain:

$$\nabla_\theta J(\theta) = \sum_x P_\theta(x)\, \nabla_\theta \ln P_\theta(x)\, f(x) = \mathbb{E}_{x \sim P_\theta}\left[f(x) \nabla_\theta \ln P_\theta(x)\right].$$

11.2.3 Policy Gradient Theorem

Theorem 1 (Policy Gradient Theorem)

$$\text{(REINFORCE)}\ \nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim d^{\pi_\theta}}\left[\nabla_\theta \left(\sum_{h=0}^{H-1} \ln \pi_\theta(a_h \mid s_h) \cdot R(\tau)\right)\right].$$

Equivalently,

$$\text{(ADVANTAGE)}\ \nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim d^{\pi_\theta}}\left[\nabla_\theta \left(\sum_{h=0}^{H-1} \ln \pi_\theta(a_h \mid s_h) \cdot A_h^{\pi_\theta}(s_h, a_h)\right)\right]$$

where $A_h^{\pi_\theta}(s_h, a_h) = Q_h^{\pi_\theta}(s_h, a_h) - V^{\pi_\theta}(s_h)$.

Proof:

$$\begin{aligned}
\nabla_\theta J(\pi_\theta) &= \mathbb{E}_\tau[\nabla_\theta \ln p_\theta(\tau) \cdot R(\tau)] \\
&= \mathbb{E}_\tau[\nabla_\theta(\ln \mu(s_0) + \ln \pi_\theta(a_0 \mid s_1) + \cdots + \ln \pi_\theta(a_{H-1} \mid s_{H-1}) + \ln p(s_H \mid a_{H-1}, s_{H-1})) \cdot R(\tau)] \\
&= \mathbb{E}_\tau\left[\nabla_\theta \left(\sum_{h=0}^{H-1} \ln \pi_\theta(a_h \mid s_h) \cdot R(\tau)\right)\right].
\end{aligned}$$

■

Interpretation $\sum_h \ln \pi_\theta(a_h \mid s_h)$ is a maximum likelihood estimation (MLE). Therefore, $\sum_h \ln \pi_\theta(a_h \mid s_h) A(s_h, a_h)$ can be viewed as some kind of advantage-weighted MLE.