September 28

1 Intro: Policy Gradient

For simplicity, let us first consider a one-step decision-making process (also known as a contextual bandit).

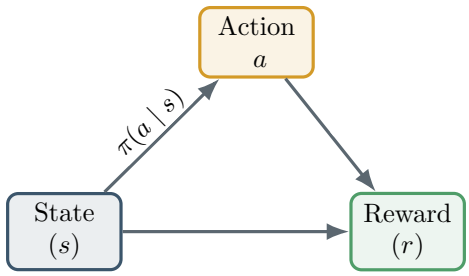
Problem Setup Let $s \in \mathcal{S}$ be the state vector of the environment, and let $a \in \mathcal{A}$ be the action taken by the agent. The reward function is denoted by $r(a, s)$. The agent's behavior is characterized by a policy, which is conveniently represented as a conditional probability distribution, $\pi(a \mid s)$, specifying the probability of selecting each action a given a state s. The objective is to find the optimal policy π^* that maximizes the expected reward:

$$R(\pi) = \mathbb{E}_{s\sim p_0}\mathbb{E}_{a\sim\pi(\cdot|s)}[r(a,s)] = \sum_{s,a} p_\pi(a,s)r(a,s),$$

where p_0 denotes the distribution of states, and we have $p_\pi(a, s)$ denotes the joint distribution of (a, s) when policy π is used:

$$p_\pi(a, s) = \pi(a \mid s)p_0(s).$$

Typically, p_0 is unknown to us, but observed through a collection of data $\{s^{(i)}\}$ drawn from p_0.



Policy Gradient In practice, the policy $\pi(a \mid s) = \pi_\theta(a \mid s)$ is parameterized by a function with parameter θ. Assume the state-action space $\mathcal{S} \times \mathcal{A}$ is finite or accountable, the gradient of the expected reward with respect to θ is

$$\begin{aligned}
\nabla_\theta R(\pi_\theta) &= \nabla_\theta \left(\sum_{s,a} p_0(s)\pi_\theta(a \mid s)r(a,s) \right) \\
&= \sum_{s,a} p_0(s)\nabla_\theta\pi_\theta(a \mid s)r(a,s) \qquad \text{//moving } \nabla_\theta \text{ into sum.} \\
&= \sum_{s,a} p_0(s)\pi_\theta(a \mid s)\frac{\nabla_\theta\pi_\theta(a \mid s)}{\pi_\theta(a \mid s)}r(a,s) \\
&= \mathbb{E}_{p_{\pi_\theta}}\left[r(a,s)\nabla_\theta \log \pi_\theta(a \mid s)\right], \qquad //p_{\pi_\theta}(a,s) = \pi_\theta(a|s)p_0(s),
\end{aligned}$$

where the parts depending on θ are highlighted, and we use the log-derivative trick $\nabla_\theta \log \pi = \nabla_\theta\pi_\theta/\pi_\theta$. This allows us to express the derivative of an expectation with respect to a distribution as the expectation of the reward weighted by $\nabla_\theta \log \pi_\theta(a \mid s)$.

Reward Baseline For any policy π^θ, it is easy to prove the following identity:

$$\mathbb{E}_{a\sim\pi_\theta(\cdot|s)}[\nabla_\theta \log \pi_\theta(a \mid s)] = 0, \qquad \forall s,\ \theta.$$

Hence, we can generalize the gradient formula above to

$$\nabla_\theta R(\pi_\theta) = \mathbb{E}_{p_{\pi_\theta}}\left[(r(a,s) - v(s))\nabla_\theta \log \pi_\theta(a \mid s)\right], \tag{1}$$

where $v(s)$ is *any* function that does not depend on the action a. The choice of v does not alter the expectation of the formula, but a proper choice of v can reduce the variance of the mean estimation given empirical observations.

One common choice of baseline is $v(s) = \mathbb{E}_{a\sim\pi_\theta(\cdot|s)}[r(a,s)]$, in which case the difference $r(a,s) - v(s)$ is known as the *advantage function*:

$$A(a,s) = r(a,s) - v(s).$$

A positive value $A(a,s) > 0$ (respectively, negative < 0) indicates that the action a performs above (respectively, below) the average.

On-Policy Estimation In practice, given observations $\{s^{(i)}\}$ drawn from p_0 and $a^{(i)} \sim \pi^\theta(\cdot|s^{(i)})$ drawn from the ongoing policy π^θ, the gradient can be estimated by

$$\nabla_\theta R(\pi_\theta) \approx \frac{1}{n}\sum_{i=1}^n A^{(i)} \nabla_\theta \log \pi_\theta(a^{(i)} \mid s^{(i)}), \qquad A^{(i)} \stackrel{\text{def}}{=} A(a^{(i)}, s^{(i)}). \tag{2}$$

The gradient ascent update, also known as REINFORCE,

$$\theta \leftarrow \theta + \epsilon \nabla_\theta R(\pi_\theta).$$

Intuitively, the gradient $\nabla_\theta \log \pi_\theta(a^{(i)} \mid s^{(i)})$ for data points $(a^{(i)}, s^{(i)})$ with positive advantages $A^{(i)} > 0$ is positively weighted, increasing the probability $\pi_\theta(a \mid s)$ during gradient ascent. Conversely, for data points with negative advantages, the gradient is negatively weighted, decreasing the probability.

Connection to Weighted MLE Assuming the data $\{a^{(i)}, s^{(i)}\}$ are fixed, the policy gradient $\nabla_\theta R(\pi_\theta)$ coincides with the gradient of the $A^{(i)}$-weighted log-likelihood function:

$$\ell(\pi_\theta) = \frac{1}{n}\sum_{i=1}^n A^{(i)} \log \pi_\theta(a^{(i)} \mid s^{(i)}).$$

This aligns with the intuition of maximizing $\log \pi(a \mid s)$ for data points with large (and positive) $A^{(i)}$, while minimizing the log-likelihood for negative $A^{(i)}$.

The key difference, however, is that the data $\{a^{(i)}, s^{(i)}\}$ are drawn from the policy π_θ and thus depend on the current parameter θ. As θ is updated during policy optimization, the data must either be regenerated or reweighted to reflect changes in π_θ. In contrast, maximum likelihood estimation assumes the data are fixed.

Hence, policy optimization can be viewed as an adaptively weighted maximum likelihood estimation (MLE), where the weights $(A^{(i)})$ are adjusted across iterations using newly generated data.

Off-Policy Estimation via Importance Sampling The gradient estimation in (2) is *on-policy*, because the actions a must be drawn from the current policy $\pi_\theta(\cdot|s)$. This does not yield efficient use of data as new actions must be rolled out once the policy is updated, and data from different policies cannot be used. In comparison, *off-policy* methods allow us to leverage actions from different policies.

One common approach to off-policy estimation is importance sampling. Assume that the data $\{a^{(i)}, s^{(i)}\}$ are drawn from a *behavior policy* $\pi^{\texttt{data}}$. Using importance sampling, we have

$$\begin{aligned}\nabla_\theta R(\pi_\theta) &= \mathbb{E}_{p_{\pi^{\texttt{data}}}}\left[\frac{\pi_\theta(a|s)}{\pi^{\texttt{data}}(a|s)} A(a,s)\nabla_\theta \log \pi_\theta(a \mid s)\right] \\ &\approx \frac{1}{z}\sum_{i=1}^n w^{(i)} A^{(i)} \nabla_\theta \log \pi_\theta(a^{(i)} \mid s^{(i)}),\end{aligned}$$

where $w^{(i)}$ is the importance weight:

$$w^{(i)} = \frac{\pi_\theta(a^{(i)}|s^{(i)})}{\pi^{\texttt{data}}(a^{(i)}|s^{(i)})},$$

and the normalization constant z is often taken as $z = \sum_i w^{(i)}$.

Remark 1. In RL, *on-policy methods* update the policy using data collected from the current policy itself. *Off-policy methods* learn from data collected under a different policy, allowing reuse of past experiences and improving sample efficiency, but at the cost of a distribution mismatch between the behavior policy π^{data} and the target policy π_θ. *Offline RL* is the case of off-policy learning where training relies entirely on a fixed dataset with no further interaction.

References