ESE 6510 Note 1: Policy Gradient

Jefferson Ng

Feb 5st, 2026

Introduction to Reinforcement Learning

Reinforcement Learning (RL) studies how an *agent* interacts with an *environment* in order to maximize cumulative reward. A common mathematical framework is an *episodic Markov Decision Process (MDP)*, defined by a state space $\mathcal{S}$, an action space $\mathcal{A}$, a transition distribution $P(s_{t+1} \mid s_t, a_t)$, and an (optional) initial state distribution μ.

At each timestep t, the agent observes a state s_t, samples an action $a_t \sim \pi(a_t \mid s_t)$, receives a reward r_t, and transitions to a new state s_{t+1}. The goal in RL is to learn a policy π that controls the agent to take actions which maximize expected cumulative reward.

Notations that we will use

- **State space $\mathcal{S}$**: possible configurations of the environment.
- **Action space $\mathcal{A}$**: choices available to the agent at each state.
- **Transition distribution $P(s_{t+1} \mid s_t, a_t)$**: probability of moving to the next state s_{t+1} (also written s') after taking action a_t in state s_t.
- **Policy $\pi(a_t \mid s_t)$**: probability of choosing action a_t at time t given state s_t.
- **Trajectory τ**: a full rollout of the policy from $t = 0$ until termination.
- **Reward r_t**: A deterministic function of the state (and action).
- **Return $R(\tau)$**: the sum $\sum_t r_t$ of rewards along a trajectory τ.

1 Policy Gradient Methods

1.1 The Main Idea of Policy Gradient

Policy Gradient (PG) methods form the foundation of many modern RL algorithms. PG methods optimize the policy *directly* by performing gradient ascent on the expected return. Intuitively, we update the policy by following gradients of the expected reward with respect to the policy parameters—hence the name *policy gradient*.

We parameterize the policy by a vector θ, typically the weights of a neural network,

$$\pi_\theta(a \mid s).$$

Executing this policy induces a distribution over trajectories

$$\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_T),$$

where each trajectory accumulates a total return

$$R(\tau) = \sum_{t=0}^{T-1} r_t.$$

The objective in PG is therefore

$$J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)}[R(\tau)] = \int p_\theta(\tau)\, R(\tau)\, d\tau,$$

Learning proceeds by updating the parameters in the direction

$$\theta \;\longrightarrow\; \nabla_\theta J(\theta).$$

From the objective to a usable gradient. Taking gradients of $J(\theta)$ is challenging because the trajectory distribution $p_\theta(\tau)$ depends on unknown environment dynamics. Expanding the gradient, we obtain

$$\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_{\tau \sim p_\theta(\tau)}[R(\tau)] = \nabla_\theta \int p_\theta(\tau)\, R(\tau)\, d\tau = \int \nabla_\theta p_\theta(\tau)\, R(\tau)\, d\tau.$$

The key technical trick that makes PG methods possible is the *score function identity.*

Score Function Identity

For any differentiable probability density $p_\theta(x)$,

$$\nabla_\theta p_\theta(x) = p_\theta(x)\, \nabla_\theta \log p_\theta(x).$$

As a consequence of probability normalization,

$$\mathbb{E}_{x \sim p_\theta}\big[\nabla_\theta \log p_\theta(x)\big] = 0.$$

Applying this identity to the objective yields

$$\nabla_\theta J(\theta) = \int p_\theta(\tau)\, \nabla_\theta \log p_\theta(\tau)\, R(\tau)\, d\tau = \mathbb{E}_{\tau \sim p_\theta(\tau)}\left[\nabla_\theta \log p_\theta(\tau)\, R(\tau)\right].$$

Next, observe that the trajectory probability factorizes as

$$p_\theta(\tau) = \mu(s_0) \prod_{t=0}^{T-1} \pi_\theta(a_t \mid s_t)\, P(s_{t+1}, r_t \mid s_t, a_t).$$

Only the policy terms depend on θ, so the gradient simplifies to

$$\nabla_\theta \log p_\theta(\tau) = \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t \mid s_t).$$

Substituting this into the objective yields the policy gradient expression:

$$\boxed{\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t \mid s_t)\, R(\tau) \right]. \qquad (1)}$$

Monte Carlo estimation of the policy gradient. The expectation in Eq. (1) cannot be computed analytically, since it depends on the unknown trajectory distribution induced by the policy and environment. In practice, we approximate this expectation using Monte Carlo sampling.

Specifically, we collect N trajectories $\{\tau_i\}_{i=1}^N \sim p_\theta(\tau)$ by rolling out the current policy, and form an empirical estimate of the gradient:

$$\hat{g} = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^{(i)} \mid s_t^{(i)})\, R(\tau_i).$$

As $N \to \infty$, this estimator converges to the true policy gradient in expectation.

Intuition: What does the score function do?

The term $\nabla_\theta \log \pi_\theta(a_t \mid s_t)$ measures how sensitive the policy's probability of selecting action a_t is to changes in the parameters θ. Multiplying this term by the observed return has a simple effect: *actions that lead to higher rewards increase their probability of being chosen again*, while actions that lead to lower rewards are discouraged. In short, policy gradient methods increase the weights of actions that worked well, and decrease the weights of actions that did not.

1.2 Vanilla Policy Gradient Algorithm

The vanilla policy gradient algorithm (commonly referred to as **REINFORCE**) updates the policy by weighting log-probability gradients with empirical returns. At this stage, we consider the simplest form of the algorithm, which uses raw Monte Carlo returns without any variance reduction techniques.

Algorithm 1 REINFORCE (Vanilla Policy Gradient)

```
1: Initialize policy parameters $\theta$
2: for each iteration do
3:     Collect trajectories $\{\tau_i\}_{i=1}^{N}$ by rolling out $\pi_\theta$
4:     for each timestep $t$ in each trajectory do
5:         Compute Monte Carlo return:
               $$R_t = \sum_{t'=t}^{T-1} \gamma^{t'-t} r_{t'}$$
6:     end for
7:     Compute gradient estimate:
           $$\hat{g} = \sum_t \nabla_\theta \log \pi_\theta(a_t \mid s_t)\, R_t$$
8:     Update policy parameters:
           $$\theta \leftarrow \theta + \alpha\, \hat{g}$$
9: end for
```

What does Monte Carlo return mean?

In REINFORCE, the return R_t is computed using the *entire future trajectory* starting from timestep t. This is referred to as a *Monte Carlo return* because it relies solely on sampled rewards from complete rollouts, without any bootstrapping or value function estimates.

Monte Carlo returns are unbiased estimates of the expected return under the policy. However, because they depend on all future rewards, they can have very high variance—especially in long-horizon or stochastic environments. This is the primary limitation of vanilla policy gradient methods.