Learning parameterized policies

In reinforcement learning, learning parameterized policies means directly optimizing a policy $\pi(a|s;\theta)$ with respect to its parameters θ, instead of indirectly via value functions.

This is the foundation of policy gradient methods, which are especially powerful when combined with function approximation (e.g., neural networks).

A **parameterized policy** is a function $\pi(a|s;\theta)$ that maps a state to a probability distribution over actions, with parameters θ (e.g., weights of a neural network).

Examples:

- **Discrete actions**: Softmax over linear preferences
- **Continuous actions**: Gaussian policy with $\mu(s;\theta), \sigma(s;\theta)$

Policy Gradient Theorem

The **policy gradient** gives the direction to adjust θ to improve expected return:

$$\nabla_\theta J(\theta) = E_\pi[\nabla_\theta \log \pi(A_t|S_t;\theta)\ \cdot q_\pi(S_t, A_t)]$$

We use the gradient to perform **stochastic gradient ascent**:

$$\theta \leftarrow \theta + \alpha \cdot \nabla_\theta J(\theta)$$

Practical Algorithm: REINFORCE

A Monte Carlo method using full return G_t:

$$\theta \leftarrow \theta + \alpha \cdot G_t \cdot \nabla_\theta \log \pi(A_t|S_t;\theta)$$

Can be high variance, so usually paired with:

- **Baselines** (e.g., subtract $v_\pi(s)$) to reduce variance
- **Actor-Critic methods** (learn both policy and value function)

Actor-Critic Methods

Use a **value function critic** $\hat{v}(s;w)$ or **advantage function** $\hat{A}(s,a)$ to guide the policy update:

$$\theta \leftarrow \theta + \alpha \cdot \hat{A}(s,a) \cdot \nabla_\theta \log \pi(a|s;\theta)$$

- **Actor**: the policy $\pi(a|s;\theta)$
- **Critic**: the value estimator (can also be neural network)

Policy Gradient for Continuing Tasks

In **continuing tasks** (no terminal states), we aim to **optimize performance over an infinite horizon** — without relying on discounting or episode boundaries. This setting is often more realistic in domains like robotics, process control, or system maintenance.

$$\rho(\theta) = \lim_{t\to\infty} \mathbb{E}_\theta[R_t]$$

Policy Gradient Theorem (Average Reward Form)

The gradient of the average reward with respect to policy parameters is:

$$\nabla_\theta \rho(\theta) \propto \sum_s d^\pi(s) \sum_s \nabla_\theta \pi(a|s;\theta) \cdot q_\pi(s,a)$$

Where:

- $d^\pi(s)$: stationary distribution under π\piπ
- $q_\pi(s,a)$: expected total reward **above average** starting from (s,a)

This leads to the familiar stochastic gradient estimate:

$$\nabla_\theta \rho(\theta) \approx \mathbb{E}_\pi\left[\nabla_\theta \log \pi(\,At \mid St;\theta\,) \cdot \left(R_{t+1} + \hat{v}(S_{t+1}) - \hat{v}(S_t)\right)\right]$$

Implementation in Actor-Critic (Continuing)

You use a **differential TD error** to update both actor and critic:

TD Error (Differential Form):

$$\delta_t = R_t + 1 - \bar{R} + \hat{v}(S_t + 1) - \hat{v}(S_t)$$

Critic Update:

$$w \leftarrow w + \beta \cdot \delta t \cdot \nabla_w \hat{v}(S_t;w)$$

Actor Update:

$$\theta \leftarrow \theta + \alpha \cdot \delta_t \cdot \nabla_\theta \log \pi(A_t \mid S_t;\theta)$$

Where:

- $\bar{R}$: moving estimate of the average reward

Policy Parameterizations

A **parameterized policy** is a function:

$$\pi(a \mid s; \theta)$$

which gives the probability of selecting action a in state s, controlled by parameters θ. These parameters can be:

- **Linear weights**
- **Neural network weights**
- **Coefficients in softmax or Gaussian distributions**

Common Forms of Policy Parameterization

1. Softmax (Discrete Actions)

Used when action space is **discrete**:

$$\pi(a \mid s; \theta) = \frac{e^{h(s,a;\theta)}}{\sum_b e^{h(s,b;\theta)}}$$

- $h(s, a; \theta)$: preference score (can be linear or nonlinear)
- Often used in REINFORCE or Actor-Critic

2. Gaussian (Continuous Actions)

Used when action space is **continuous**:

$$\pi(a \mid s; \theta) = \mathcal{N}(\mu(s; \theta), \sigma^2(s; \theta))$$

- Mean and std dev are outputs of a neural network
- Used in robotics, control, etc.

Gradient of Log Policy

Key property used in **policy gradient methods**:

$$\nabla_\theta \log \pi(a \mid s; \theta)$$

This term allows you to compute how the probability of choosing an action changes as parameters are adjusted. It's used in the update:

$$\theta \leftarrow \theta + \alpha \cdot G_t \cdot \nabla_\theta \log \pi(A_t \mid S_t; \theta)$$

or with TD error δ_t in Actor-Critic.