Tại Sao Cần Hàm Tham Số Hóa

1. Khả năng học từ dữ liệu

- Trong học máy, chúng ta thường không biết trước chính xác hàm ánh xạ từ đầu vào → đầu ra.

Do đó, ta sử dụng một **hàm có tham số** (ví dụ: mạng nơ-ron, hồi quy tuyến tính, Q-function trong RL...), và **điều chỉnh các tham số đó (training)** để hàm phù hợp với dữ liệu huấn luyện.

2. Tổng quát hóa tốt hơn

- Hàm tham số hóa cho phép ta khái quát từ **một số hữu hạn dữ liệu huấn luyện** sang **dữ liệu chưa từng thấy**.
- Nếu không tham số hóa, ta chỉ có thể ghi nhớ dữ liệu (overfitting), không thể khái quát (generalize).

3. Linh hoạt và có thể tối ưu

- Hàm tham số hóa cho phép sử dụng các thuật toán tối ưu (như gradient descent) để cập nhật tham số.
- Nhờ đó, ta có thể **tối thiểu hóa lỗi dự đoán**, **tối đa hóa phần thưởng**, hoặc thực hiện các mục tiêu khác trong học máy và reinforcement learning.

🔷4. **Ứng dụng trong Reinforcement Learning**

Trong RL, ta có thể:

- Dùng **chính sách tham số hóa**: $\pi(a|s,\theta)$, trong đó θ là tham số.
- Dùng **hàm giá trị tham số hóa**: $Q(s,a;\theta)$

  → Điều này giúp agent học tối ưu chính sách bằng cách điều chỉnh θ\thetaθ qua trải nghiệm.

Hàm tham số hóa sử dụng một tập các trọng số có thể điều chỉnh để ước lượng giá trị của các trạng thái:

$$v_{\hat{w}}(s) \approx v_{\pi}(s)$$

Trong đó:

$v_{\hat{w}}(s)$: giá trị ước lượng cho trạng thái s

$v_{\pi}(s)$: giá trị thực của trạng thái s theo chính sách π

$\hat{w}$: vector trọng số

Công thức tổng quát:

$$v_{\hat{w}}(s) = \sum_{i=1}^{n} w_i \cdot x_i(s)$$

Trong đó:

- wi: trọng số thứ i
- xi(s): đặc trưng thứ i của trạng thái s
- n: số lượng đặc trưng

Các Loại Hàm Xấp Xỉ

1. Hàm xấp xỉ tuyến tính (Linear Function Approximation)

$$\hat{v}(s; \mathbf{w}) = \mathbf{w}^T \cdot \mathbf{x}(s)$$

- w: vector trọng số
- φ(s): feature vector của state s

Non-linear Function

$$\theta_{t+1} = \theta_t - \alpha \nabla_\theta J(\theta_t)$$

- θ: tham số cần học
- α: learning rate (tốc độ học)
- ∇J: gradient (hướng tăng nhanh nhất)
- Đi ngược hướng gradient để minimize J

Sự kết hợp SL và RL

Ví dụ trên là Imitation Learning (behavior cloning). Khi có expert data, SL giúp khởi tạo policy. Sau đó có thể tiếp tục RL (reinforcement fine-tuning) để cải thiện dựa reward thực.
Trong trường hợp không có expert data, SL vẫn dùng để học value approximation thông qua regression với target là return/T D-target.

Monte Carlo Gradient Method

Gradient Monte Carlo là một thuật toán kết hợp giữa phương pháp Monte Carlo và gradient descent để ước lượng hàm giá trị trong học tăng cường.

Công thức Gradient

$$\nabla_w J(w) = \sum_{s \in S} \mu(s)[v_\pi(s) - \hat{v}(s,w)]\nabla_w \hat{v}(s,w)$$

Với $\hat{v}(s,w) = w^T \phi(s)$

Gradient: $\nabla_w \hat{v}(s,w) = \phi(s)$

Bước 1: Khởi Tạo

- Chọn hàm xấp xỉ $\hat{v}(s, w)$ khả vi theo w
- Khởi tạo vector trọng số w
- Chọn tốc độ học α

Bước 2: Tạo Episode

- Tương tác với môi trường theo chính sách π
- Thu thập chuỗi trạng thái và phần thưởng
- Tính toán giá trị trả về (return) cho mỗi trạng thái

Bước 3: Cập Nhật Trọng Số

- Với mỗi trạng thái trong episode:
  - Tính gradient của hàm xấp xỉ
  - Cập nhật trọng số theo công thức:

$$w_{t+1} = w_t + \alpha[G_t - \hat{v}(s_t, w_t)]\nabla_w \hat{v}(s_t, w_t)$$

Trong đó Gt là giá trị trả về mẫu

Stochastic Gradient Descent

$$w_{t+1} = w_t + \alpha[v_\pi(s_t) - \hat{v}(s_t, w_t)]\nabla_w \hat{v}(s_t, w_t)$$

Trong đó:

- α: tốc độ học
- st: trạng thái tại bước thời gian t