

Định lý cải tiến chính sách (Policy Improvement Theorem)


- **Giả thiết:** Cho một chính sách π , ta có hàm giá trị hành động $Q^\pi(s, a)$ cho mọi cặp (s, a) .
- **Mục tiêu:** Tìm một chính sách π' sao cho

$$Q^\pi(s, \pi'(s)) \geq V^\pi(s) \quad \forall s,$$

nghĩa là, tại mỗi trạng thái s , hành động $\pi'(s)$ cho giá trị kỳ vọng không thua kém hành động $\pi(s)$ cũ.

- **Định lý cải tiến:** Nếu ở mọi trạng thái s ta có

$$Q^\pi(s, \pi'(s)) \geq V^\pi(s),$$

thì chính sách π' **không kém hơn** π trên mọi trạng thái; nếu tồn tại ít nhất một s mà $Q^\pi(s, \pi'(s)) > V^\pi(s)$, thì π' **chặt chẽ tốt hơn** π ở ít nhất một trạng thái. 

- **Ý nghĩa:** Ta có thể "greedify" (làm chính sách tham lam) đối với V^π hay Q^π để thu được chính sách mới π' sao cho π' về lý thuyết sẽ cho kết quả ngắn hạn (theo giá trị hiện tại) không thua kém, và có thể tốt hơn π về tổng reward dài hạn.

3. Phương pháp Policy Iteration

Policy Iteration gồm hai bước lặp lại cho đến khi chính sách hội tụ:

1. Policy Evaluation (Ước lượng chính sách)

- Cho chính sách π_i , tính ước lượng hàm giá trị trạng thái $V^{\pi_i}(s)$ chính xác (hoặc xấp xỉ) bằng **iterative policy evaluation**.
- Nói cách khác, giải hệ Bellman Expectation Equation:

$$V^{\pi_i}(s) = \sum_a \pi_i(a | s) \sum_{s', r} P(s', r | s, a) [r + \gamma V^{\pi_i}(s')] \quad \forall s.$$

- Thực hiện quét (sweeps) cho đến khi $\|V_{\text{new}} - V_{\text{old}}\|$ đủ nhỏ, khi đó $V \approx V^{\pi_i}$.

2. Policy Improvement (Cải tiến chính sách)

- Dựa trên V^{π_i} (hoặc Q^{π_i}), xây dựng chính sách mới π_{i+1} bằng cách tại mỗi trạng thái s , chọn hành động a sao cho

$$a = \arg \max_{a'} \sum_{s', r} P(s', r | s, a') [r + \gamma V^{\pi_i}(s')],$$


tức hành động **greedy** đối với V^{π_i} .

- Mà chức năng này chính là greedy action:

$$\pi_{i+1}(s) = \arg \max_a Q^{\pi_i}(s, a).$$

- Theo định lý cải tiến, π_{i+1} sẽ không yếu hơn π_i , và nếu π_i chưa tối ưu, thì π_{i+1} sẽ "chặt chẽ" tốt hơn ở ít nhất một trạng thái.

3. Kiểm tra dừng

- Nếu sau bước cải tiến, $\pi_{i+1} = \pi_i$, tức không có cải tiến nào nữa, thì π_i đã là **optimal policy** π^* . Quá trình dừng.
- Ngược lại, gán $i \leftarrow i + 1$ và lặp lại. 

Tóm tắt thuật toán:

markdown



Sao chép



Chỉnh sửa

Initialize random policy π_1

$i \leftarrow 1$

Repeat:

1. Policy Evaluation: compute $V^{\{\pi_i\}}$ (iteratively)

2. Policy Improvement:

$\pi_{i+1}(s) \leftarrow \operatorname{argmax}_a \sum_{s',r} P(s',r|s,a) [r + \gamma V^{\{\pi_i\}}(s')] \quad \forall s$

3. If $\pi_{i+1} = \pi_i$ then

return π_i (đã tìm được optimal policy)

Else

$i \leftarrow i + 1$

End Repeat

4. Ví dụ minh họa trên lưới 4×4

- Môi trường:** Grid World 4×4, không gian trạng thái gồm 16 ô.
 - Có hai trạng thái terminal ở góc trên bên trái (trạng thái A) và góc dưới bên phải (trạng thái B).
 - Mỗi bước di chuyển, agent nhận reward = -1, discount factor $\gamma = 1$.
 - Chọn policy ngẫu nhiên ban đầu (π_1): tại mỗi ô phi-terminal, bốn hướng (up/down/left/right) đều có xác suất 0.25.

Bước 1: Policy Evaluation cho π_1

- Thực hiện iterative policy evaluation:
 - Khởi tạo $V(s) = 0$ cho mọi s .
 - Lặp quét qua tất cả 14 trạng thái phi-terminal, cập nhật

$$V_{\text{new}}(s) = \sum_a \pi_1(a|s) \sum_{s',r} P(s',r|s,a) [r + V_{\text{old}}(s')].$$

- Sau nhiều sweep, ta có bảng giá trị V^{π_1} , phản ánh "độ gần" trung bình đến terminal (vì reward mỗi bước là -1).

Bước 2: Policy Improvement

- Tại mỗi ô s , tính cho từng hướng a :

$$\sum_{s',r} P(s',r|s,a)[r + V^{\pi_1}(s')].$$

- Lựa chọn hướng $a^* = \arg \max$ biểu thức trên để làm $\pi_2(s)$.
- Kết quả là một bảng policy (đường đi ngắn nhất đến terminal) – tức deterministic: tại mỗi ô, chọn đi về ô kề gần terminal nhất (phải hay xuống).

Bước 3: Lặp lại

- So sánh π_2 với π_1 : tất nhiên khác nhau (π_1 là random, π_2 đã "hướng dẫn" agent đi ngắn nhất).
- Gán $\pi_1 \leftarrow \pi_2$, tiếp tục đánh giá (V^{π_2}) cho tới khi policy không thay đổi nữa.
- Cuối cùng, π^* là policy "luôn đi thẳng đến terminal nhanh nhất" từ mọi ô.

5. Những đặc điểm nổi bật của Policy Iteration

1. **Chính sách luôn deterministic tại mỗi step:** Mỗi π_i sau cải tiến đều chọn duy nhất một hành động với xác suất 1.
2. **Thuật toán hội tụ nhanh:** Thông thường chỉ cần vài vòng lặp (vài cặp evaluation–improvement) để tìm π^* , đặc biệt khi không gian trạng thái nhỏ.
3. **Tách biệt rõ hai bước:**
 - *Evaluation step* (ước lượng chính sách): dựa trên Bellman Expectation Equation.
 - *Improvement step* (cải tiến chính sách): greedy hóa đối với giá trị vừa tính.
4. **Mỗi cải tiến là "strict"** (nếu policy chưa tối ưu): theo định lý, policy mới luôn tốt hơn hoặc ít nhất không kém policy cũ.
5. **Hạn chế:**
 - **Đòi hỏi biết mô hình chuyển tiếp** $P(s', r|s, a)$.
 - **Không gian trạng thái lớn:** khó khăn khi số trạng thái hàng triệu, vì mỗi sweep phải quét qua toàn bộ trạng thái.

6. Hiện thị trực quan (Visualization of Policy Iteration)

1. **Khởi tạo:** Policy random, $V_0(s) = 0$.
2. **Vòng lặp:**

- **Evaluation:** Hình dung từng bảng V^{π_i} dần ổn định. Ban đầu giá trị phân bố gần nhau, về sau rõ ràng giữa vùng gần terminal và xa terminal.
 - **Improvement:** Từ bảng V^{π_i} , vẽ ra mũi tên (arrows) chỉ hướng greedy cho π_{i+1} . Mỗi lần cải tiến, mũi tên đều hướng đến hình ảnh đường ngắn nhất dần dần.
3. **Kết thúc:** Khi policy (mũi tên) không thay đổi, tức là tất cả mũi tên đều chỉ về hướng đưa agent đến terminal nhanh nhất; $V(s)$ song song cho thấy giá trị tối ưu.
-

7. Sức mạnh và ưu điểm

- **Độ tin cậy cao:** Mỗi bước đều được đảm bảo không làm giảm giá trị chính sách (monotonic improvement).
 - **Tốc độ hội tụ:** Thường nhanh hơn Value Iteration vì mỗi lần evaluation ước lượng khá chính xác V^π trước khi greedy.
 - **Dễ mở rộng:** Khi lồng vào Approximate Policy Iteration, có thể xử lý không gian trạng thái lớn bằng cách xấp xỉ hàm.
 - **Ứng dụng:**
 - MDP quy mô vừa và nhỏ, mô hình biết trước.
 - Khi cần hàm giá trị chính xác trước khi cải tiến.
-

8. Tóm tắt (Summary)

1. **Policy Iteration** là phương pháp lặp hai bước:
 - **Policy Evaluation:** Ước lượng đầy đủ V^π cho policy hiện tại.
 - **Policy Improvement:** Lấy hành động greedy để sinh policy mới.
2. **Định lý cải tiến chính sách** đảm bảo policy mới luôn không yếu hơn policy cũ, và ít nhất một cải tiến sẽ xảy ra nếu policy cũ chưa tối ưu.
3. Sau một số vòng lặp, nếu policy không thay đổi, ta thu được **optimal policy** π^* và hàm giá trị tối ưu V^* .
4. **Áp dụng tiêu biểu:** Grid World 4×4, khởi chính sách random, sau vài vòng lặp policy sẽ trở thành “luôn đi thẳng vào terminal” với giá trị tối ưu.