


Chính sách (Policies)

- Định nghĩa chung:

- Một **policy** π trong RL là một bản đồ (mapping) từ mỗi trạng thái s sang một phân phối xác suất trên tập các hành động khả dĩ.
- Ký hiệu:

$$\pi(a | s) = P(A_t = a | S_t = s),$$

nghĩa là xác suất chọn hành động a khi agent đang ở trạng thái s . 

- Hai loại policy cơ bản:

1. **Deterministic policy**: Luôn chọn một hành động duy nhất với xác suất 1 mỗi khi gặp trạng thái đó.


- Ký hiệu đơn giản: $\pi(s) = a$.
- Ví dụ (trong một grid world): Agent ở ô (i, j) sẽ luôn di chuyển "lên" để tiến gần về phía nhà (goal). Mỗi trạng thái tương ứng với một mũi tên cố định chỉ hướng "up."



2. **Stochastic policy**: Cho mỗi trạng thái, policy trả về một phân phối xác suất trên các hành động.

- Ký hiệu: $\pi(a | s)$ như trên, thỏa:

$$\sum_a \pi(a | s) = 1, \quad \pi(a | s) \geq 0, \quad \forall a, s.$$

- Ví dụ: Ở trạng thái s_0 , policy có thể quy định:
 - $\pi(\text{"up"} | s_0) = 0.5,$
 - $\pi(\text{"right"} | s_0) = 0.5,$
 - ngược lại 0 cho hai hướng còn lại.
- Do vậy, agent ở s_0 sẽ chọn "up" hoặc "right" mỗi lần với xác suất 0.5, giúp cân bằng giữa hai lựa chọn. 

3. Ví dụ về policy (Policy Example)

- Deterministic policy example:

- Giả sử một agent di chuyển trên một lưới (grid) để về nhà (goal).
- Tại mỗi ô, quy tắc (**policy**) là "luôn đi hướng mũi tên chỉ tới nhà". Ví dụ, nếu nhà nằm ở phía trên bên phải, tại ô (2,2) mũi tên chỉ "up" (lên), tại ô (2,3) mũi tên chỉ "right."
- Khi agent tuân theo policy này, mỗi bước sẽ tiến về phía goal mà không thử lối đi khác.




- **Stochastic policy example:**

- Cùng grid như trên, giả sử ở hàng dưới, agent muốn xen kẽ giữa "up" và "right" nhưng không ưu tiên rõ ràng. Policy có thể quy định:


$$\pi(\text{"up"} \mid s) = 0.5, \quad \pi(\text{"right"} \mid s) = 0.5,$$

cho mọi trạng thái s ở hàng dưới.

- Làm vậy, agent đôi khi đi lên, đôi khi đi phải, nhưng trên dài hạn vẫn hướng về goal giống policy deterministic, chỉ khác ở chỗ ngẫu nhiên hơn. 

4. Hàm giá trị (Value Functions)

- **Vai trò chung:**

- **Value functions** giúp ước lượng mức độ "tốt" khi ở một state (hoặc khi thực hiện một action từ state đó) dưới một policy nhất định.
- Dựa vào các giá trị này, agent biết nên ưu tiên vào state nào, action nào để tối đa hóa tổng reward dài hạn. 


- **Hai loại value functions chính:**

1. **State Value Function** $V^\pi(s)$:

- Định nghĩa: kỳ vọng tổng return khi bắt đầu từ trạng thái s và theo policy π trong tương lai.
- Công thức:

$$V^\pi(s) = E_\pi[G_t \mid S_t = s] = E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right],$$

trong đó:

- G_t là tổng discounted return từ thời điểm t .
- γ là discount factor ($0 \leq \gamma < 1$).
- Kỳ vọng E_π tính trên phân phối hành động do π sinh ra và phân phối chuyển tiếp trạng thái. 
- Ý nghĩa: $V^\pi(s)$ cho biết "goodness" của state s nếu agent cứ tiếp tục theo policy π .

2. **Action Value Function** $Q^\pi(s, a)$:

- Định nghĩa: kỳ vọng tổng return nếu agent bắt đầu từ trạng thái s , thực hiện action a ngay, rồi tiếp tục theo policy π sau đó.
- Công thức:

$$Q^\pi(s, a) = E_\pi[G_t \mid S_t = s, A_t = a] = E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right].$$

- Ý nghĩa: $Q^\pi(s, a)$ biểu diễn "goodness" của việc chọn a khi ở s , rồi tiếp tục theo π .



- **Mối quan hệ giữa V^π và Q^π :**

- Nếu policy π là xác suất chọn action, thì:

$$V^\pi(s) = \sum_a \pi(a | s) Q^\pi(s, a).$$

- Cũng có thể diễn giải: $V^\pi(s)$ là **trung bình** của $Q^\pi(s, a)$ theo phân phối $\pi(a | s)$.



- **Tại sao cần cả hai?**

- $V^\pi(s)$: giúp so sánh giữa các state; thường dùng trong **policy evaluation** (ước lượng giá trị một policy cụ thể).
- $Q^\pi(s, a)$: giúp so sánh giữa các cặp (state, action); thường dùng trong **action selection**, đặc biệt khi agent muốn "tối ưu hóa greedy" là chọn action có $Q^\pi(s, a)$ cao nhất.



5. Ví dụ minh họa giá trị trong một MDP đơn giản



- **Mô tả MDP** (continuing task):

- Grid 3x3 (9 ô), agent có thể đi lên, xuống, trái, phải. Nếu agent cố đi ra khỏi lưới, nó sẽ va chạm (bump) và nhận reward -1 , đồng thời ở nguyên vị trí cũ. Các bước di chuyển bình thường, không va chạm, reward = 0.
- Có hai "trạng thái đặc biệt" (special states) A và B:
 1. Ở **trạng thái A**, bất kỳ action nào cũng trả về $+10$ reward và chuyển agent tới trạng thái A' (ô xác định sẵn).
 2. Ở **trạng thái B**, bất kỳ action nào cũng trả về $+5$ reward và chuyển agent tới trạng thái B' (ô xác định).
- Sau khi rời A hoặc B, agent ở vị trí mới và tiếp tục tương tác. Do vậy, A và B luôn là "nguồn reward tích cực" bên cạnh phần âm khi va chạm.



- **Policy được chọn:**

- Sử dụng **random policy**: tại mỗi state, agent chọn mỗi action với xác suất bằng nhau (1/4 nếu 4 hướng khả dụng).
- Discount factor $\gamma = 0.9$.
- Muốn tính $V^\pi(s)$ cho mọi s , ta giải **Bellman expectation equation** hoặc ước lượng qua lặp giá trị (value iteration) khi policy cố định.



- **Kết quả ước lượng (State Values):**

- Bảng giá trị $V^\pi(s)$ hiển thị con số cho từng ô trong grid (trong slide).
- Quan sát:

1. $V^\pi(A)$ cao nhất (khoảng >10) vì từ A mỗi bước agent chắc chắn nhận +10 rồi rời, sau đó về khu vực dễ va chạm hoặc dễ tiếp cận A/B.
2. $V^\pi(B)$ xấp xỉ giá trị lớn hơn 5 (khoảng 5. x), vì B cho +5 rồi chuyển tới B', và vị trí B' có xu hướng dễ đi đến A hoặc tránh va chạm.
3. Các ô gần đáy lưới (gần "tường") có giá trị âm hoặc rất thấp, vì random policy dễ va chạm nhiều lần, liên tục nhận -1. 📄

• **Giải thích chi tiết:**

- Tại A: mỗi bước, chắc chắn nhận +10 (gần như ngay lập tức). Discounting và xác suất rời A dẫn tới khu vực rủi ro (bump) nên $V^\pi(A) = 10 + 0.9 \cdot (\text{giá trị ở A'})$. Vì A' thường ở giữa, dễ tránh tường \rightarrow con số sau $10 \cdot (1 + 0.9^2 + \dots)$ gần xấp xỉ $10/(1-0.9) = 100$, nhưng do yếu tố rủi ro bump nên thực tế thấp hơn, song vẫn lớn nhất. 📄
- Tại B: tương tự nhưng reward chỉ +5, tức $V^\pi(B) \approx 5 + 0.9 \cdot (\text{giá trị ở B'})$. B' nằm ở vùng mà random policy dễ dàng bounce đến A nhiều lần, vì vậy giá trị tổng cộng xấp xỉ lớn hơn 5, nhưng không cao bằng A. 📄
- Tại các ô xung quanh tường: do chọn random, thường có 1/4 xác suất va chạm mỗi bước \rightarrow trung bình nhận -0.25 reward (0 với 3/4, -1 với 1/4). Discounted sum nhiều bước \rightarrow số âm lớn.

6. Vai trò trong việc tối ưu hóa policy 📄

- **Policy Evaluation:** Khi policy π cố định, ta tính hoặc ước lượng V^π và Q^π để biết mức độ tốt của policy đó. Phương pháp lặp Bellman (iterative policy evaluation) được dùng để giải:

$$V_{k+1}(s) = \sum_a \pi(a | s) \sum_{s', r} P(s', r | s, a) [r + \gamma V_k(s')].$$

- **Policy Improvement:** Khi đã có Q^π , ta có thể xây dựng policy mới π' sao cho:

$$\pi'(s) = \arg \max_a Q^\pi(s, a),$$

tức chọn action có giá trị cao nhất tại mỗi state. Dần dần lặp hai bước evaluation và improvement thành **policy iteration** để tìm policy tối ưu. 📄

- **Value-based Action Selection:**

- Với $Q^\pi(s, a)$ đã ước lượng, một agent "greedy" sẽ chọn $a^* = \arg \max_a Q^\pi(s, a)$ tại mỗi s , giúp khai thác action tốt nhất dài hạn.
- Trong các thuật toán **Q-learning** hoặc **SARSA**, agent ước lượng trực tiếp $Q(s, a)$ mà không cần ước lượng riêng $V(s)$. 📄

7. Tóm tắt (Summary)

1. **Policy** là phân phối xác suất $\pi(a | s)$ cho mỗi action a tại mỗi state s . Có thể là deterministic (chọn một action cố định) hoặc stochastic (xác suất > 0 cho nhiều action).
2. **State Value Function** $V^\pi(s)$ đo lường kỳ vọng tổng return khi bắt đầu ở s và thực hiện policy π về sau. Công thức:

$$V^\pi(s) = E_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s \right].$$

3. **Action Value Function** $Q^\pi(s, a)$ đo lường kỳ vọng tổng return khi bắt đầu ở s , thực hiện trực tiếp action a , rồi theo policy π . Công thức:

$$Q^\pi(s, a) = E_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a \right].$$

4. **Mối quan hệ:**

$$V^\pi(s) = \sum_a \pi(a | s) Q^\pi(s, a).$$

5. **Ví dụ MDP đơn giản** cho thấy cách tính $V^\pi(s)$ khi policy cố định (random policy, $\gamma = 0.9$), rồi quan sát sự khác biệt về giá trị giữa các state (A, B, vùng gần tường).
6. **Ứng dụng:** Dựa vào V^π và Q^π , agent có thể thực hiện **policy improvement** hoặc **action selection** (greedy) để tối ưu hóa policy, tạo nên quy trình policy iteration hay các thuật toán Q-learning/SARSA.

Gợi ý mở rộng:

- Khi policy chưa biết, một agent có thể sử dụng on-policy methods (ví dụ SARSA) để ước lượng Q^π đồng thời cải thiện policy, hoặc dùng off-policy methods (như Q-learning) để ước lượng $Q^*(s, a)$ – giá trị tối ưu không phụ thuộc policy hiện tại.
- Các thuật toán điển hình còn dựa vào khái niệm Bellman Optimality Equation cho $V^*(s)$ và $Q^*(s, a)$ nhằm tìm policy tối ưu.