Optimal Policy (Chiến lược tối ưu)

- **Định nghĩa chung**
  - Một **optimal policy** π^* là policy mà khi theo π^*, agent sẽ đạt tổng reward kỳ vọng tối đa trong dài hạn.
  - Công thức: π^* thỏa mãn

    $$V^{\pi^*}(s) \geq V^{\pi}(s) \quad \forall s, \forall \pi,$$

    nghĩa là giá trị (state-value) dưới policy π^* luôn bằng hoặc lớn hơn giá trị dưới bất kỳ policy nào khác cho mọi trạng thái.
- **Vai trò của policy trong RL**
  1. **Action Selection (Chọn hành động)**: Policy quyết định cách agent chọn action tại mỗi trạng thái. Optimal policy chắc chắn sẽ chọn hành động "tốt nhất" trong mỗi state.
  2. **Learning Objective (Mục tiêu học)**: Giúp định nghĩa rõ ràng bài toán RL—agent cố gắng tìm π làm sao cho long-term reward là cao nhất.
  3. **Exploration vs. Exploitation**: Trong quá trình tìm optimal policy, agent vẫn cần cân bằng giữa khám phá (để ước lượng tốt hơn) và khai thác (để thu reward cao hơn hiện tại). Optimal policy về sau sẽ thiên về khai thác.
  4. **Evaluation of States and Actions**: Policy dùng để đánh giá "tốt/xấu" của từng trạng thái, từng action dựa trên reward dài hạn.
  5. **Adaptation and Generalization**: Khi môi trường thay đổi, agent có thể điều chỉnh policy; policy cũng giúp tổng quát hóa giữa các state tương tự.
  6. **Representation of Knowledge**: Trong nhiều thuật toán như actor-critic, policy được thể hiện trực tiếp (thông qua mạng neural hay bảng tra), thể hiện kiến thức agent đã học.
  7. **Policy Improvement**: Các thuật toán như policy iteration hay các phương pháp gradient-based sẽ cập nhật policy liên tục cho đến khi đạt optimal.
- **Các loại policy**
  1. **Deterministic Policy ($\pi(s)$)**: Với mỗi state s, luôn chọn một action duy nhất a.
  2. **Stochastic Policy ($\pi(a \mid s)$)**: Với mỗi state s, phân phối xác suất cho từng action; có thể chọn ngẫu nhiên nhưng ưu tiên action có giá trị lớn hơn.
  3. **Optimal Policy (π^*)**: Là policy (có thể nhiều policy nếu cùng giá trị tối ưu) làm cho tổng reward kỳ vọng cao nhất.

---

3. Ví dụ minh họa sơ lược về optimal policies

- **MDP đơn giản "Two Choice"**:
  - **Mô tả**:
    1. Ở trạng thái X, agent có hai action:

- $A1$ → chuyển đến Y, nhận reward (theo slide).
- $A2$ → chuyển đến Z, nhận reward (có thể khác).

2. Từ Y, chỉ có một action $A1$ → quay về X.
3. Từ Z, chỉ có $A1$ → quay về X.
4. Mỗi action đều gắn reward cụ thể (ví dụ +5 nếu đi đường này, +10 nếu đi đường kia).

- **Hai deterministic policies (π_1 và π_2)**:
  - π_1: Ở X luôn chọn $A1$.
  - π_2: Ở X luôn chọn $A2$.
- **Câu hỏi**: Policy nào là optimal? Trả lời phụ thuộc vào **discount factor** γ, bởi vì:
  - Nếu γ nhỏ, agent ưu tiên reward tức thời → có thể $A2$ với reward cao ngay lập tức tốt hơn.
  - Nếu γ lớn, agent quan tâm nhiều đến reward trong tương lai (qua chu trình Y–X hay Z–X) → có thể $A1$ tốt hơn xét tổng lâu dài.
- **Kết luận**: Optimal policy không chỉ phụ thuộc reward tức thời, mà phải tính tổng discounted return trên đường hồi chuỗi về X.

---

4. Optimal Value Function (Giá trị tối ưu)

4.1 Optimal State-Value Function $V^*(s)$

- **Định nghĩa**:

$$V^*(s) = \max_\pi V^\pi(s) = \max_\pi \mathrm{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s].$$

  - Cho biết "giá trị" cao nhất (tối ưu) mà agent có thể đạt được khi bắt đầu ở state s và sau đó theo policy tốt nhất.
- **Bellman Optimality Equation cho V^***:

$$V^*(s) = \max_a \sum_{s',r} P(s', r \mid s, a)\,[r + \gamma\, V^*(s')].$$

  - Tức là, tại mỗi state s, agent sẽ chọn action a sao cho kỳ vọng "reward tức thì r + discounted giá trị state kế tiếp" là lớn nhất.
  - Đây là nền tảng để giải bài toán tìm V^* qua **value iteration** hay **policy iteration** khi biết mô hình.

4.2 Optimal Action-Value Function $Q^*(s, a)$

- **Định nghĩa**:

$$Q^*(s,a) \;=\; \max_{\pi} Q^{\pi}(s,a) \;=\; \max_{\pi} \mathrm{E}_{\pi}[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \;\Big|\; S_t = s,\; A_t = a].$$

  - Đưa ra giá trị tối ưu nếu agent khởi đầu bằng việc chọn action a tại s, sau đó tiếp tục theo policy tốt nhất.
- **Bellman Optimality Equation cho Q^***:

$$Q^*(s,a) \;=\; \sum_{s',r} P(s',r \mid s,a)\,[\,r + \gamma \max_{a'} Q^*(s',a')].$$

  - Khi ở (s,a), agent nhận reward r và chuyển sang s'. Tiếp đến agent sẽ chọn action a' tối ưu (maximization) để thu về $Q^*(s',a')$.

---

5. Sử dụng Optimal Value Function để tìm Optimal Policy

- **Quan hệ giữa V^* và π^***:

$$\pi^*(s) \;=\; \arg\max_{a} \sum_{s',r} P(s',r \mid s,a)\,[\,r + \gamma\, V^*(s')] \;=\; \arg\max_{a} Q^*(s,a).$$

  - Khi đã biết $V^*(s)$, ta xác định action a làm cho "reward tức thì + γ giá trị state kế tiếp" đạt max.
  - Khi có $Q^*(s,a)$, ta chỉ cần chọn $\pi^*(s) = \arg\max_a Q^*(s,a)$.
- **Quy trình tóm tắt**:
  1. Tính hoặc ước lượng V^* (qua Bellman optimality equation).
  2. Từ đó, tại mỗi state s, chọn action tối ưu: $\pi^*(s)$.
  3. Kết quả là một deterministic policy tối ưu (hoặc stochastic nếu nhiều action cùng chung giá trị max).

---

6. Ví dụ cụ thể từ slide

- **Mô tả môi trường**:
  1. **State A**: Tất cả action từ A dẫn đến A' với reward +10.
  2. **State B**: Tất cả action từ B dẫn đến B' với reward +5.
  3. **Phần còn lại của grid**: Reward = 0 trừ khi bump vào tường thì reward = –1.
  4. **Discount factor**: $\gamma = 0.9$.
- **Tính $V^*(s)$ cho mọi state**:

1. **Tại** A: Mỗi action chắc chắn cho +10 ngay, sau đó chuyển sang A'. Ở A', agent có thể random hoặc chọn sao cho sớm quay lại A để tiếp tục nhận +10.
    - Về nguyên tắc,

      $$V^*(A) = 10 + 0.9\,V^*(A'),$$

      và $V^*(A')$ lại có khả năng quay về A hay đi những state khác.
    - Khi giải đầy đủ hệ, kết quả ước lượng cho $V^*(A)$ lớn hơn 10, bởi vì sự lặp lại gần như vô hạn của việc "nhận +10 rồi quay về vùng trung tâm".
2. **Tại** B: Mỗi action từ B cho +5 rồi về B'. Tương tự,

   $$V^*(B) = 5 + 0.9\,V^*(B'),$$

   và $V^*(B')$ lại có thể tiếp tục quay về B hay vùng an toàn khác.
    - Kết quả tính toán cho thấy $V^*(B)$ rơi vào khoảng 5.2 (như ví dụ trước), vì khả năng quay về B thường xuyên.
3. **Các state khác**: Tính tương tự qua Bellman optimality. State càng gần tường, giá trị càng thấp do –1 khi bump.

- **Chọn** π^*:
    - Tại mỗi state, agent đương nhiên chọn action "hướng đến A (hoặc B)" càng nhanh càng tốt.
    - Tức là, nếu ở state có thể đi về A (giá trị +10) thì ưu tiên hơn đi về B (+5), bởi vì 10 + 0.9·(...) > 5 + 0.9·(...).
    - Kết quả: Agent gần A sẽ luôn chọn action dẫn đến A, gần B chọn hướng đến B, nếu ở vùng an toàn trung tâm, agent sẽ tìm đường ngắn nhất lên A vì đó là reward lớn nhất tổng hợp lâu dài.

---

7. Mối quan hệ giữa Value Function và Optimal Policy

- **Value Function** (dưới optimal policy) nói lên "kỳ vọng tổng return tốt nhất" khi ở mỗi state.
- **Optimal Policy** sẽ luôn chọn action làm tăng giá trị state lên gần nhất với $V^*(s)$.
- Kết luận:
    - **Value function** cung cấp "bản đồ" của reward dài hạn từ mỗi state.
    - **Optimal policy** dùng bản đồ đó để quyết định action nào tối ưu tại mỗi state, nhằm dẫn đến con đường có tổng reward lâu dài lớn nhất.

---

8. Tóm tắt (Summary)

1. **Optimal Policy (π^*)**: Chính sách (deterministic hoặc stochastic) làm cho tổng discounted return kỳ vọng là cao nhất từ mọi state.
2. **Optimal State-Value Function ($V^*(s)$)**: Kỳ vọng return tối đa bắt đầu từ state s nếu theo optimal policy về sau. Thỏa mãn Bellman Optimality Equation:

$$V^*(s) = \max_a \sum_{s',r} P(s', r \mid s, a)\,[r + \gamma\, V^*(s')].$$

3. **Optimal Action-Value Function ($Q^*(s,a)$)**: Kỳ vọng return tối đa nếu đầu tiên chọn action a ở s, sau đó theo optimal policy. Thỏa mãn:

$$Q^*(s,a) = \sum_{s',r} P(s', r \mid s, a)\,[r + \gamma\, \max_{a'} Q^*(s', a')].$$

4. **Từ V^* đến π^***:

$$\pi^*(s) = \arg\max_a \sum_{s',r} P(s', r \mid s, a)[r + \gamma\, V^*(s')] = \arg\max_a Q^*(s,a).$$

5. **Ví dụ minh họa**: Môi trường có state A (+10), B (+5), và các vùng tường (–1). Discount $\gamma = 0.9$. Tính V^* cho từng state → chọn π^* dẫn đến A càng nhanh càng tốt, sau đến B, tránh vùng tường.

Gợi ý mở rộng:

- **Khi mô hình chuyển tiếp $P(s', r \mid s, a)$ biết trước, có thể giải trực tiếp Bellman Optimality Equation (qua value iteration).**
- **Khi mô hình không biết, dùng Q-learning hoặc SARSA để ước lượng $Q^*(s,a)$ dần dần từ dữ liệu thực nghiệm; sau đó suy ra π^*.**