Reinforcement Learning là gì?

- **Ý tưởng chung**:
  - RL là một quá trình **dựa trên phản hồi (feedback-based)**, trong đó tác nhân (agent) tương tác lặp đi lặp lại với môi trường (environment).
  - Mỗi bước, tác nhân chọn một hành động (action), môi trường phản hồi bằng phần thưởng (reward) và trạng thái mới (state).
  - Tác nhân học từ chính **kinh nghiệm** (experience) của mình để cải thiện hiệu suất qua thời gian.
- **Đặc điểm khác biệt so với supervised learning**:
  - **Không có dữ liệu gắn nhãn sẵn (no labeled data)**. Thay vì nhận "câu trả lời đúng" cho mỗi đầu vào, tác nhân chỉ thấy reward (có thể dương hoặc âm) dựa trên hành động đã thực hiện.
  - Quá trình học giống như cách trẻ em học: **"làm → nhận biết đúng/sai qua khen/nghiêm → điều chỉnh"**, chứ không phải học từ bộ dữ liệu đã gán nhãn.
- **Khái niệm cơ bản**:
  1. **Agent**: thực thể ra quyết định (ví dụ: robot, con thỏ, chương trình chơi cờ).
  2. **Environment**: thế giới mà agent tương tác (grid world, trò chơi cờ, mô phỏng lái xe...).
  3. **State (S)**: đại diện cho "hoàn cảnh" hiện tại mà agent quan sát được (có thể là vị trí robot, tình trạng board cờ, dữ liệu cảm biến...).
  4. **Action (A)**: tập các lựa chọn mà agent có thể thực hiện ở mỗi state (ví dụ: di chuyển, ăn rau củ, đi nước cờ).
  5. **Reward (R)**: tín hiệu phản hồi (scalar) nhận được ngay sau khi agent thực hiện action ở state hiện tại (ví dụ: số điểm, khen/thưởng, phạt).
  6. **Policy (π)**: quy tắc (chiến lược) ánh xạ từ state → action; có thể là deterministic (luôn chọn hành động nhất định) hoặc probabilistic (chọn theo xác suất).

---

3. Mục tiêu của Reinforcement Learning: Maximize Reward

- **Mục tiêu chung**:
  - **Tối đa hóa tổng reward** mà agent thu được trong tương lai.
  - Khi agent chọn hành động, nó hy vọng vừa nhận reward ngay lập tức và đồng thời tạo điều kiện cho những reward cao hơn trong tương lai.
- **Công thức tổng quát (Return)**:
  - Giả sử agent bắt đầu ở bước thời gian t. Định nghĩa **return** G_t là tổng các reward từ bước $t+1$ cho đến hết tập:

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \dots$$

Hoặc nếu sử dụng **discount factor** γ (thường $0 \leq \gamma < 1$) để giảm giá trị các reward xa hơn:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots$$

- **Mục tiêu của agent** là tìm policy π sao cho **kỳ vọng** của G_t lớn nhất, tức:

$$\max_{\pi} \mathrm{E}_{\pi}[\, G_t \,].$$

- **Khác biệt với supervised learning**:
  - Trong supervised, hàm mất mát (loss) dựa trên dữ liệu gắn nhãn so sánh trực tiếp "đầu vào → đầu ra mong muốn".
  - Trong RL, agent không biết trước "hành động đúng". Thay vào đó, agent thử hành động, **nhận reward** (có thể tốt/xấu) rồi dùng reward này làm thước đo đánh giá, dần điều chỉnh policy sao cho reward tích lũy được ngày càng lớn.
- **Positive vs Negative Reinforcement Learning**
  1. **Positive Reinforcement**: Khi agent thực hiện hành động "tốt" (đúng đích), môi trường thêm một tín hiệu tích cực (positive reward) để **tăng xác suất** agent lặp lại hành vi đó trong tương lai.
     - Ví dụ: Trong trò chơi, khi giành được điểm, agent nhận +10 → khuyến khích chọn giống hành động này.
  2. **Negative Reinforcement**: Khi agent thực hiện hành động mong muốn, môi trường **loại bỏ** hoặc **giảm thiểu** một tín hiệu tiêu cực. Hành động đó được tăng khả năng xảy ra để "tránh" việc giữ tín hiệu xấu.
     - Ví dụ: Trong môi trường lái xe tự động, nếu không phanh kịp, agent nhận -5 mỗi giây xe vượt quá tốc độ. Khi agent học cách phanh đúng lúc, tín hiệu "-5" biến mất → agent học rằng phanh sớm tránh phạt.
  3. **Khác biệt với Punishment (hình phạt)**: Negative reinforcement là tránh tín hiệu xấu thông qua hành động phù hợp, không phải nhận phạt sau khi làm "sai".

---

4. Key Components of Reinforcement Learning

1. **Agent**
   - Người học (learner) hoặc người ra quyết định (decision-maker).
   - Nhận thông tin về state, chọn action theo policy, rồi cập nhật lại policy dựa trên reward nhận được.
2. **Environment**
   - Hệ thống bên ngoài tương tác với agent.
   - Khi agent thực hiện action, environment trả về hai thông tin:
     - **Reward** R_{t+1} (một số vô hướng).
     - **New state** S_{t+1}.

- Environment cũng xác định **điều kiện chấm dứt** (terminal condition) cho một tập (episode), nếu là episodic task.

3. **State (S)**
   - Mô tả đầy đủ các thông tin cần thiết để quyết định hành động tiếp theo.
   - Thông thường, state phải thỏa tính chất Markov: nó "bao gói" đủ lịch sử ngắn gọn để dự đoán tương lai.
4. **Action (A)**
   - Tập các lựa chọn agent có thể thực thi khi ở state hiện tại.
   - Policy π sẽ ánh xạ từ mỗi state vào một hành động cụ thể hoặc phân phối xác suất lên tập action.
5. **Reward (R)**
   - Tín hiệu phản hồi vô hướng, thường dương nếu hành động tốt, âm nếu hành động xấu.
   - Reward tức thời R_{t+1} giúp agent đánh giá trực tiếp tác động của action trong step kế tiếp.
   - Agent quan tâm đến **tổng reward tương lai** (return) chứ không chỉ reward ngay lập tức.
6. **Policy (π)**
   - Là chiến lược/tập quy tắc: $\pi(a \mid s)$ biểu diễn xác suất chọn action a khi ở state s.
   - Agent học để cải thiện policy sao cho tổng reward kỳ vọng tối đa.

---

5. Mục tiêu ngắn hạn và dài hạn

- **Reward ngay (Immediate reward)**: Phản hồi tức thời sau khi agent chọn action.
- **Return dài hạn (Long-term return)**: Tổng số reward trong tương lai mà agent sẽ thu được, thường có công thức tổng vô hạn nếu không giới hạn tập:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}, \quad 0 \le \gamma < 1.$$

- **Mục tiêu học**: Không phải chỉ chọn action tối đa R_{t+1} mà còn cân nhắc để $R_{t+2}, R_{t+3}, \ldots$ được tối ưu. Ví dụ, đôi khi agent phải chấp nhận reward tức thời thấp (ví dụ –1) để đi đến vùng có reward cao hơn trong tương lai (ví dụ +10).

---

6. Phân loại bài toán RL: Episodic vs. Continuing

1. **Episodic Tasks**
   - Mỗi tập (episode) kết thúc sau một số bước cố định hoặc khi đạt điều kiện chấm dứt (terminal state).
   - Sau khi kết thúc, môi trường được **reset** về trạng thái khởi đầu, bắt đầu tập mới.

- Tính chất **finite horizon**: có độ dài hữu hạn (tùy thuộc vào bài toán).
- **Đặc điểm**:
  - Có "mục tiêu" rõ ràng trong mỗi tập (ví dụ: về đích, giải xong bài toán).
  - Reward được tính dồn cho mỗi tập, agent tìm cách đạt mục tiêu càng nhanh càng tốt.
- **Ví dụ**:
  - Chơi xong một ván cờ (một episode).
  - Thực hiện robot đi qua mê cung đến đích; khi đến đích, môi trường reset.
  - Tập lái xe tự động trong simulator, mỗi lần xuất phát → đến đích/va chạm là kết thúc.

2. **Continuing Tasks**
   - Không có điểm kết thúc rõ ràng; agent tương tác liên tục vô hạn với môi trường.
   - Cần sử dụng **discount factor** $\gamma < 1$ để đảm bảo tổng return hội tụ.
   - **Ví dụ**:
     - Điều khiển robot dọn dẹp nhà cửa suốt ngày (không có điểm dừng).
     - Quản lý hệ thống giao thông, liên tục điều phối.

---

7. Chi tiết về Episodic Tasks

- **Episode Termination**: Khi đạt đến trạng thái kết thúc (terminal state) hoặc một giới hạn số bước. Ví dụ:
  - Trong mê cung, khi robot đến ô đích → episode kết thúc.
  - Trong cờ vua, khi checkmate hoặc hòa → kết thúc ván.
- **Environment Reset**: Sau mỗi episode, môi trường quay về trạng thái ban đầu hoặc một trạng thái khởi tạo đã định.
- **Finite Horizon**: Số bước tối đa trong một episode có thể cố định sẵn (ví dụ 100 bước) hoặc phụ thuộc điều kiện nhiệm vụ (ví dụ, hết tiền/điểm hụt).
- **Goal-Oriented**:
  - Agent tối đa tổng reward tích lũy **trong mỗi episode**.
  - Nhiều tác vụ có "goal state" mà agent cần đạt đến (ví dụ: ô đích trong mê cung, checkmate trong cờ).

Ví dụ minh họa:

- **Chơi một ván cờ:**
  - **Episode bắt đầu khi ván cờ khởi tạo vị trí chuẩn.**
  - **Mỗi lượt đi, agent (AI) chọn nước đi, môi trường (trò chơi) phản hồi trạng thái board mới.**

    - **Khi ván kết thúc (checkmate, hòa, hoặc hết nước đi), đó là terminal state. Kết thúc episode, AI bắt đầu ván mới.**
- **Robot hoàn thành mê cung:**
    - **Robot khởi tại ô xuất phát.**
    - **Mỗi bước, robot chọn di chuyển lên/xuống/trái/phải, nhận reward +1 khi đến đích hoặc –1 nếu đụng tường (tùy cài đặt).**
    - **Khi robot tới ô đích, gặp chướng ngại cụ thể, hoặc vượt quá giới hạn bước, episode kết thúc.**

---

8. Tại sao cần phân biệt Episodic và Continuing?

- **Cách tính return**:
    - Trong episodic, mỗi episode có điểm kết thúc xác định → dễ tính tổng return (có giới hạn).
    - Trong continuing, phải dùng discount factor để tổng return hội tụ.
- **Thuật toán giải quyết**:
    - Một số thuật toán (như Monte Carlo) đòi hỏi episodic để có tập trải nghiệm đầy đủ cho mỗi episode.
    - Frier-algorithms (như TD learning, Q-learning) có thể áp dụng cho cả hai, nhưng cần chú ý cách cập nhật và đánh giá.
- **Thiết kế reward**:
    - Episodic thường dùng "reward sparse" ở cuối (ví dụ +1 khi hoàn thành nhiệm vụ).
    - Continuing có thể dùng reward đều qua từng bước để hướng agent tối đa tổng reward vô hạn.

---

9. Tổng kết và Tóm tắt (Summary)

1. **Reinforcement Learning** là quá trình agent:
    - **Quan sát state**
    - **Chọn action** theo policy
    - **Nhận reward** và state mới
    - **Cập nhật** policy để tối đa hóa tổng reward tương lai.
2. **Không có dữ liệu gắn nhãn**: Agent phải tự "thử – học" dựa trên reward, tương tự cách con người học qua kinh nghiệm.
3. **Positive vs Negative Reinforcement**:

- **Positive**: thêm reward khi hành động đúng → tăng xác suất lặp lại.
- **Negative**: loại bỏ tín hiệu xấu khi hành động đúng → hành động được lặp lại để tránh phạt.

4. **Các thành phần cơ bản**:
   - Agent, Environment, State, Action, Reward, Policy.
5. **Mục tiêu chính**: Tổng reward dài hạn (return) phải lớn nhất:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots$$

   Agent tìm policy π để $\mathrm{E}_\pi[G_t]$ đạt max.
6. **Phân loại tác vụ**:
   - **Episodic Tasks**: Kết thúc sau số bước hữu hạn hoặc khi đạt điều kiện, episode reset, finite horizon.
   - **Continuing Tasks**: Tương tác liên tục vô hạn, cần discount để tính return hội tụ.
7. **Ví dụ Episodic**: Ván cờ, mê cung, điều hướng robot đến mục tiêu.

Gợi ý áp dụng thêm:

- **Khi triển khai RL, ngay từ đầu cần xác định xem bài toán là episodic hay continuing để chọn thuật toán và thiết kế reward phù hợp.**
- **Trong các bài toán thực tế (ví dụ lái xe tự động), thường ban đầu xem là episodic (mỗi chuyến đi), sau đó có thể mở rộng thành continuing (liên tục nhận tín hiệu, không reset).**