Mục tiêu (Objectives)

- Phân biệt rõ **episodic tasks** và **continuing tasks**.
- Giải thích cách sử dụng **discounting** (hệ số chiết khấu) cho các continuing tasks.

---

2. Khái niệm Continuing Tasks

- **Continuing tasks** (còn gọi là ongoing tasks hoặc recurring tasks) là những hoạt động kéo dài và lặp đi lặp lại mà **không có điểm kết thúc xác định**.
  - Những tác vụ này đòi hỏi phải "chạy mãi" hoặc "theo dõi duy trì" trong một khoảng thời gian vô hạn hoặc cho đến khi không còn cần thiết.
  - Ví dụ: hệ thống điều hòa nhiệt độ nhà máy, quản lý lưu lượng mạng, robot dọn dẹp liên tục, v.v.

Đặc điểm chính của Continuing Tasks:

1. **Ongoing Nature (Tính liên tục)**
   - Tác vụ tồn tại và chạy mãi theo thời gian; không có terminal state cố định như ở episodic tasks.
   - Agent phải luôn tương tác với môi trường, không có giai đoạn "dừng hẳn" khi hoàn thành một episode.
2. **Regular Intervals (Chu kỳ lặp lại)**
   - Có thể xảy ra theo chu kỳ cố định (hàng ngày, hàng tuần, hàng tháng) hoặc tùy theo điều kiện "khi cần".
   - Ví dụ:
     - Cứ mỗi phút, robot kiểm tra vị trí và làm sạch khu vực.
     - Cứ mỗi giờ, hệ thống thu thập dữ liệu cảm biến.
3. **No Defined Endpoint (Không có điểm kết thúc xác định)**
   - Khác với episodic tasks, không có trạng thái "kết thúc" rõ ràng dẫn đến việc reset toàn bộ môi trường.
   - Nếu không có discounting, tổng reward có thể là vô hạn và không hội tụ.
4. **Maintenance and Monitoring (Bảo trì và giám sát)**
   - Cần liên tục theo dõi, điều chỉnh để đảm bảo hệ thống hoạt động ổn định, hiệu quả.
   - Ví dụ: thay đổi mức nhiệt độ, bù năng lượng,... khi điều kiện môi trường thay đổi.
5. **Evolution Over Time (Tiến hóa theo thời gian)**
   - Yêu cầu và điều kiện của tác vụ có thể thay đổi (ví dụ: mùa hè/năm đông, lưu lượng người trong nhà).
   - Agent phải thích nghi liên tục, cập nhật policy hoặc model để phản ứng với những thay đổi môi trường.

3. Minh họa: Ví dụ “Smart Thermostat”

- **Bối cảnh**: Một bộ điều khiển nhiệt độ thông minh (smart thermostat) phải duy trì nhiệt độ tòa nhà phù hợp với thói quen, thời gian trong ngày và số người hiện diện.
- **Agent**: Thiết bị thermostat (hệ thống điều khiển).
- **Environment**: Toàn bộ tòa nhà (bao gồm cảm biến nhiệt độ, bố trí phòng, sự hiện diện của con người, thời tiết bên ngoài).
- **State (S)**:
  - Nhiệt độ hiện tại trong tòa nhà.
  - Thời gian trong ngày (sáng, trưa, chiều, tối).
  - Số lượng người đang có mặt (có thể ảnh hưởng đến nhiệt độ mong muốn).
- **Action (A)**:
  - Bật máy sưởi (turn on heater).
  - Tắt máy sưởi (turn off heater).
- **Reward (R)**:
  - Nếu thermostat đoán đúng nhiệt độ mong muốn của người dùng (tức không phải “có người điều chỉnh thủ công”), reward = **0**.
  - Mỗi khi người dùng phải “can thiệp” hoặc tự chỉnh lại nhiệt độ, reward = **–1** (âm).
  - Mục tiêu của thermostat là **giảm thiểu** số lần âm (tức là tránh để người dùng phải tự chỉnh).

Lý thuyết hoạt động:

1. Ban đầu, thermostat không biết rõ sở thích nhiệt độ của người dùng theo từng khung giờ.
2. Qua mỗi lần người dùng tự chỉnh, thermostat nhận được reward = –1 và biết rằng action hiện tại chưa phù hợp. Nó sẽ điều chỉnh policy để “dự đoán” nhiệt độ mong muốn tốt hơn lần sau.
3. Quá trình tương tác diễn ra liên tục, không có điểm dừng: ngày nối ngày, thermostat thu thập dữ liệu, điều chỉnh policy, tối ưu hóa để reward (số lần người chỉnh thủ công) càng ít càng tốt.

---

4. Bất cập khi không có Discounting

- Vì **continuing tasks** không có terminal state, nếu tính tổng return là

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

  thì tổng này **có thể diverge** (không hội tụ), vì reward âm/dương sẽ cộng dồn vô hạn.
- Do đó, ta cần **discounting** (chiết khấu) để đảm bảo tổng trọng số của các reward vô hạn vẫn là một số hữu hạn.

5. Discounting (Chiết khấu)

- Ta giới thiệu **discount factor** γ, với:

$$0 \leq \gamma < 1.$$

- **Return có discounting** được định nghĩa là:

$$G_t \;=\; R_{t+1} \;+\; \gamma\, R_{t+2} \;+\; \gamma^2\, R_{t+3} \;+\; \cdots \;=\; \sum_{k=0}^{\infty} \gamma^k\, R_{t+k+1}.$$

- Khi $\gamma < 1$, thỏa mãn:

$$\sum_{k=0}^{\infty} \gamma^k < \infty \quad \Longrightarrow \quad G_t \text{ có khả năng hội tụ (finite).}$$

- **Giải thích hiệu ứng chiết khấu**:
  1. **Immediate rewards (Reward ngay lập tức)** (R_{t+1}) đóng góp trọng số 1 (bởi vì $\gamma^0 = 1$).
  2. **Future rewards** ($R_{t+2}, R_{t+3}, \ldots$) bị giảm dần tác dụng bởi $\gamma, \gamma^2, \ldots$.
  3. Reward càng ở xa tương lai, càng được "giảm giá" nhiều hơn, vì γ^k nhỏ dần khi k tăng.
- **Tác động của γ lên hành vi agent**:
  1. Nếu $\gamma = 0$:
     - Ta chỉ quan tâm reward ngay lập tức. Return $G_t = R_{t+1}$.
     - Agent cực kỳ **short-sighted** (cận thị): chỉ chọn action tối đa hóa giá trị reward bước kế tiếp, không để ý đến phần thưởng về lâu dài.
  2. Nếu $0 < \gamma < 1$:
     - Agent kết hợp giữa immediate reward và future reward, nhưng chú trọng hơn vào immediate reward. Dần dần cân bằng cả hai theo hệ số.
  3. Nếu $\gamma \to 1$:
     - Agent trở nên **farsighted** (viễn thị), sẵn sàng chấp nhận reward nhỏ/bị âm ở bước kế tiếp nếu đổi lại nhận được reward lớn hơn trong tương lai.
     - Khi γ rất gần 1, future rewards được đánh giá gần ngang bằng immediate reward, dẫn đến chiến lược ưu tiên tối đa tổng reward dài hạn.

---

6. Tóm tắt phân biệt Episodic vs. Continuing

1. **Episodic Tasks**
   - Có điểm kết thúc xác định (terminal state) → xác định độ dài hữu hạn (finite horizon).
   - Ví dụ: mỗi ván cờ, mỗi lần robot hoàn thành mê cung, mỗi lượt chơi game.
   - Total return thường tính đến khi kết thúc episode.
2. **Continuing Tasks**

- Không có terminal state → tác vụ chạy vô hạn.
- Cần **discounting** để return hội tụ.
- Ví dụ: smart thermostat, robot dọn dẹp liên tục, hệ thống giao thông tự động.

3. **Cách sử dụng discounting**:
   - Đưa vào công thức return để "giảm tầm quan trọng" của những reward quá xa về tương lai, đảm bảo tổng vô hạn tính ra vẫn hữu hạn.
   - Lựa chọn γ dựa vào mong muốn của bài toán: càng muốn tập trung vào long-term thì γ càng gần 1, ngược lại nếu ưu tiên immediate thì γ thấp hơn.

---

7. Bài tập gợi ý cho Học viên

> **"Cho một ví dụ về continuing task, và phân tích đầy đủ các thành phần: agent, environment, trạng thái, hành động, reward, và cách tính return có discounting."**

Ví dụ mẫu (Robot giám sát cây trồng tự động):

1. **Agent**: Con robot di chuyển giữa các luống cây, thu thập dữ liệu độ ẩm, ánh sáng, tưới tiêu khi cần.
2. **Environment**: Cánh đồng cây trồng với cảm biến độ ẩm, cảm biến ánh sáng, thời tiết thay đổi.
3. **State (S)**:
   - Độ ẩm hiện tại ở luống X.
   - Mức ánh sáng (nắng/mây/râm).
   - Thời gian trong ngày.
4. **Action (A)**:
   - Tiến đến luống kế tiếp.
   - Tưới nước luống hiện tại.
   - Ở nguyên và chờ (nếu độ ẩm và ánh sáng vừa đủ).
5. **Reward (R)**:
   - +1 nếu robot tưới đúng lúc giúp cây phát triển (độ ẩm trở về ngưỡng lý tưởng).
   - –1 nếu robot tưới quá sớm (lãng phí nước) hoặc quá trễ (cây khô héo).
6. **Return (có discounting)**:

$$G_t = R_{t+1} + \gamma\, R_{t+2} + \gamma^2\, R_{t+3} + \cdots, \quad 0 \le \gamma < 1.$$

   - Nếu γ chọn đủ lớn (gần 1), robot sẽ ưu tiên tưới theo long-term benefits (giảm héo lâu dài). Nếu γ nhỏ, robot chỉ quan tâm tránh tưới sai ở step kế tiếp.

---

8. Tóm tắt (Summary)

- **Continuing tasks**: tác vụ lặp vô hạn, cần bảo trì/giám sát liên tục, không có điểm kết thúc xác định.
- **Discounting** (γ) là cách duy nhất để tính tổng return hữu hạn trong continuing tasks.
- Giá trị γ điều chỉnh mức độ "nhìn xa trông rộng" (farsighted) hay "cận thị" (short-sighted) của agent:
  - $\gamma = 0$ → chỉ quan tâm reward ngay.
  - $\gamma \longrightarrow 1$ → cân bằng reward ngay và reward tương lai.
- Khi áp dụng RL cho continuing tasks, luôn cần xác định rõ:
  1. **Agent**: đối tượng ra quyết định.
  2. **Environment**: bối cảnh tương tác.
  3. **State – Action – Reward**: các thành phần cơ bản.
  4. **Return với discounting**: để tính toán tổng reward vô hạn một cách hội tụ.