2. Ưu điểm của Temporal-Difference Learning

2.1. Online Learning

- **Khái niệm**: TD learning cập nhật giá trị ngay sau mỗi bước chuyển (transition) chứ không phải đợi kết thúc cả episode.
- **Ý nghĩa**:
  - Cho phép agent học "liên tục" trong khi tương tác với môi trường, rất phù hợp với các ứng dụng thực thời (real-time) – ví dụ robot điều khiển trong nhà máy, game đối kháng online, trading tài chính theo luồng giá.
  - Khi dữ liệu đến liên tục, agent có thể liên tục cải thiện ước lượng mà không cần lưu trữ toàn bộ episode.

2.2. Hiệu quả (Efficiency)

- **Lợi thế so với Monte Carlo**:
  - MC phải đợi cho đến khi episode kết thúc mới tính toán return rồi mới cập nhật giá trị, tốn thời gian và bộ nhớ (phải lưu toàn bộ trajectory).
  - TD chỉ cần một bước chuyển (hoặc một chuỗi ngắn) để tính lỗi TD và update, giảm đáng kể chi phí tính toán, đặc biệt với những episode dài hoặc không gian trạng thái lớn.

2.3. Bootstrapping

- **Khái niệm**: TD kết hợp ý tưởng "bootstrap" từ Dynamic Programming, tức dùng **giá trị ước lượng hiện tại của trạng thái kế** để làm mục tiêu (TD target), sau đó điều chỉnh giá trị của trạng thái trước.
- **Ví dụ**:
  - Khi ở trạng thái s_t, nhận reward r_{t+1} và chuyển sang s_{t+1}, ta sử dụng

    $$\text{TD target} = r_{t+1} + \gamma\, V(s_{t+1})$$

    để cập nhật $V(s_t)$.
  - Nhờ vậy, thông tin mới được "lan truyền" nhanh qua nhiều trạng thái mà không cần đợi cả episode như MC.

2.4. Học từ chuỗi không hoàn chỉnh (Incomplete Sequences)

- **Điểm mạnh**: TD có thể học ngay cả khi không có episode rõ ràng hoặc khi môi trường liên tục (continuing tasks).
- **Ý nghĩa**: Trong những môi trường không có state terminal rõ ràng (ví dụ robot tuần tra liên tục, huấn luyện trading), MC không áp dụng được vì không biết khi nào kết thúc episode. TD vẫn update bình thường qua mỗi transition.

2.5. Cân bằng Khám phá – Khai thác (Exploration & Exploitation)

- **Tự nhiên**: Công thức update TD vừa dùng reward quan sát được (khai thác) vừa dùng giá trị ước tính tương lai (giá trị tiềm năng) để điều chỉnh.
- **Hậu quả**: Khi agent cập nhật, những trạng thái mới, ít được trải nghiệm sẽ có giá trị ước lượng thay đổi lớn hơn, khuyến khích agent quay lại khám phá thêm. Ngược lại, trạng thái "đã rõ" ít thay đổi, agent có thể khai thác.

2.6. Không cần mô hình (Model-Free Learning)

- **Khái niệm**: TD learning không đòi hỏi biết hàm chuyển tiếp $P(s'|s,a)$ hay hàm reward $R(s,a)$. Agent chỉ cần quan sát trực tiếp reward và next-state.
- **Lợi ích**:
  - Áp dụng cho các môi trường "đen" (black-box) khi ta không biết hay không thể tính trước mô hình.
  - Giảm độ phức tạp, không phải mất công ước lượng hay thiết kế mô hình.

2.7. Cải thiện chính sách (Policy Improvement)

- **Ứng dụng**: TD mở rộng dễ dàng sang việc học hàm giá trị hành động $Q(s,a)$, ví dụ các thuật toán Q-Learning (off-policy) hay SARSA (on-policy).
- **Cách thức**:
  1. Dùng TD để ước lượng dần $Q(s,a)$ qua các bước tương tác.
  2. Chọn hành động greedy (hoặc ε-greedy) dựa trên Q để cải thiện policy.
  3. Lặp lại cho đến khi hội tụ policy tối ưu.

---

3. So sánh TD vs Monte Carlo

Để làm rõ hơn, slide đưa bảng đối chiếu giữa TD và MC theo nhiều khía cạnh:

Tiêu chí	TD Learning	Monte Carlo
Online Learning	Cập nhật ngay sau mỗi bước chuyển, phù hợp học liên tục.	Chỉ cập nhật khi episode kết thúc, không học real-time.
Efficiency	Cập nhật qua single transition, nhanh và ít overhead, đặc biệt với episode dài.	Phải tính và lưu toàn bộ return của episode, tốn công sức với các episode dài hoặc không gian trạng thái lớn.
Bootstrapping	Sử dụng giá trị ước lượng $V(s_{t+1})$ làm mục tiêu, lan truyền thông tin nhanh qua trạng thái.	Không bootstrap, phải đợi toàn bộ return của episode, lan truyền thông tin chậm hơn, nhất là khi reward hiếm.

Tiêu chí	TD Learning	Monte Carlo
Incomplete Sequences	Có thể học từ bất kỳ transition nào, không cần chờ episode kết thúc, phù hợp với non-episodic tasks.	Yêu cầu complete episodes; không học được khi môi trường không có điểm kết rõ.
Exploration–Exploitation	Cập nhật TD tự nhiên kết hợp giữa reward ngay và kỳ vọng tương lai, khuyến khích agent vừa khám phá vừa khai thác một cách cân bằng.	MC không tự giải quyết trade-off này; cần kết hợp thêm epsilon-greedy hoặc các cơ chế khác để đảm bảo khám phá.
Model-Free	Không cần biết mô hình chuyển tiếp, học trực tiếp từ trải nghiệm.	Cũng model-free, nhưng hạn chế ở việc chờ complete episodes.

---

4. Ví dụ minh họa

Mặc dù slide không đưa thêm case study cụ thể, ta có thể tưởng tượng các ứng dụng:

- **Game chuyên sâu**: Trong một game lớn có vô số trạng thái (ví dụ StarCraft, Dota), TD cho phép bot học liên tục trong trận mà không phải đợi kết thúc cả trận mới cập nhật.
- **Robot điều khiển**: Robot di chuyển trong môi trường thực tế, TD giúp nó điều chỉnh ngay lập tức khi gặp chướng ngại mà không phải đợi hành trình hoàn tất.
- **Tài chính**: Thuật toán trading cập nhật giá trị của các trạng thái thị trường liên tục theo từng tick giá, không cần đợi đóng cửa phiên.