

1. Mục tiêu (Objectives)

- Hiểu cách Monte Carlo methods được sử dụng để ước lượng hàm giá trị (value function) từ những tương tác lấy mẫu (sampled interaction).
 - Ở đây, "value function" là giá trị kỳ vọng của một trạng thái (hoặc một cặp trạng thái-hành động) khi theo một chính sách cố định.
 - Monte Carlo methods cho phép ta giả lập (simulate)/lấy mẫu một số lần tương tác (episodes) giữa agent và môi trường, rồi dựa vào kết quả thu về (returns) để ước lượng giá trị thực của trạng thái đó.
- Xác định những bài toán có thể giải bằng Monte Carlo methods.
 - Monte Carlo thường áp dụng cho các bài toán học tăng cường (reinforcement learning) mà môi trường có tính ngẫu nhiên hoặc quá phức tạp để tính toán giá trị đúng (ở dạng công thức) một cách chính xác.
 - Ví dụ: ước lượng giá trị của trạng thái trong bài toán chơi bài Blackjack, hay trong các môi trường grid world đơn giản, nơi chỉ biết mô hình thông qua tương tác.
- Sử dụng Monte Carlo prediction để ước lượng hàm giá trị (state-value function) cho một chính sách đã cho (given policy).
 - "Prediction" ở đây nghĩa là khi ta đã có một chính sách (policy) cố định, và muốn tính hàm giá trị $V^\pi(s)$ cho mọi trạng thái s . Monte Carlo prediction là một phương pháp không dùng mô hình (model-free), dựa vào kết quả trung bình các return từ nhiều episode "chạy" theo chính sách đó.

2. Khái niệm Monte Carlo

2.1. Định nghĩa cơ bản

- Monte Carlo methods là một tập hợp các thuật toán tính toán sử dụng lấy mẫu ngẫu nhiên (random sampling) để thu được kết quả số học.
- Trong học tăng cường (reinforcement learning), Monte Carlo methods được dùng để ước lượng hàm giá trị (value function) hoặc chính sách (policy) bằng cách lặp đi lặp lại lấy mẫu các episode (chuyến trải nghiệm) giữa agent và môi trường.

Cụ thể, Monte Carlo làm như sau:

- Lấy mẫu (Sampling) nhiều tập các trải nghiệm (episodes), trong đó mỗi episode là một dãy (sequence) gồm các cặp (state, action, reward) đi từ trạng thái đầu đến trạng thái kết thúc (terminal).
- Với mỗi trạng thái s (hoặc cặp (s, a)), ta thu thập tất cả các giá trị return (được gọi là G_t) quan sát được từ thời điểm t mà trạng thái đó xuất hiện cho đến khi kết thúc episode.
- Ước lượng giá trị (value) cho trạng thái s (theo một chính sách cố định π) bằng trung bình cộng tất cả các return đó:

$$V(s) \approx \frac{1}{N(s)} \sum_{i=1}^{N(s)} G_t^{(i)},$$

với $N(s)$ là số lần trạng thái s đã xuất hiện trong các episode đã thu thập, và $G_t^{(i)}$ là return quan sát được khi s xuất hiện lần thứ i .

2.2. Ví dụ minh họa rất đơn giản

- Slide đưa ví dụ: "roll 12 dice a few times và thấy kết quả trung bình khoảng 41.57, rất gần với giá trị trung bình lý thuyết 42."
 - Mục đích là để minh họa khái niệm: nếu ta tung 12 viên xúc sắc (dice) nhiều lần, ta sẽ thu được tổng (sum) của 12 mặt xúc sắc. Khi lặp lại số lần tung nhiều, giá trị trung bình các tổng số thu được sẽ tiến dần đến giá trị kỳ vọng thực của "tổng của 12 viên xúc sắc".
 - Đây là một ví dụ "Monte Carlo" rất đơn giản: dữ liệu (sum của 12 dice) được lấy mẫu ngẫu nhiên, rồi lấy trung bình để ước lượng giá trị kỳ vọng.

3. Nguyên lý ước lượng giá trị qua lấy mẫu (Sampling)

Chi tiết hơn về cách Monte Carlo ước lượng giá trị trong RL:

- Quan sát nhiều return G_t từ cùng một trạng thái
 - Trong khi agent tương tác với môi trường, mỗi lần agent đến trạng thái s , ta đợi đến khi episode đó kết thúc, rồi tính return tại thời điểm t (sau khi agent đã đi qua từ trạng thái s đến khi kết thúc).
 - Ví dụ: nếu ta ghi nhận rằng agent đến trạng thái s vào thời điểm t , và khi episode kết thúc (tại thời điểm T), tổng reward từ t đến T (có thể đã được nhân với hệ số chiết khấu γ) chính là G_t .
- Càng nhiều mẫu, giá trị ước lượng càng xấp xỉ giá trị thực
 - Khi số mẫu (số lần trạng thái s xuất hiện) tăng lên, trung bình mẫu càng tiến gần giá trị kỳ vọng thực $E[G_t | S_t = s]$.

- Đây chính là định lý giới hạn trung tâm (central limit theorem): khi số mẫu đủ lớn, giá trị trung bình mẫu hội tụ về giá trị kỳ vọng, và độ lệch chuẩn của trung bình mẫu tỉ lệ nghịch với căn bậc hai của số mẫu.

3. Chỉ áp dụng được với bài toán theo episodes (episodic tasks)

- Vì return chỉ được tính "đầy đủ" khi episode kết thúc, nên Monte Carlo truyền thống chỉ áp dụng cho môi trường có trạng thái terminal (episode kết thúc). Với bài toán liên tục (continuing tasks), cần các điều chỉnh đặc biệt (nhưng slide này tập trung vào episodic).

4. Ước lượng giá trị bang (Value of a Policy)

- Trong bài toán **bandit**, ta ước lượng giá trị của từng "arm" (cánh tay) bằng giá trị trung bình payoff thu được khi rút (pull) arm đó.
- Monte Carlo methods **mở rộng** khái niệm này vào **học tăng cường**: thay vì ước lượng giá trị của một cánh tay, ta ước lượng **giá trị của một chính sách** (policy).
 - **Giá trị của trạng thái s dưới chính sách π** được định nghĩa là **kỳ vọng tổng discounted reward** khi bắt đầu từ s , rồi tiếp tục hành động theo π cho đến khi kết thúc episode:

$$V^\pi(s) = E[G_t | S_t = s],$$

với

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots + \gamma^{T-t-1} R_T, \quad \gamma \in [0, 1].$$

- **Monte Carlo prediction**: đường dẫn để ước lượng $V^\pi(s)$ bằng cách "lấy mẫu" (simulate) nhiều episode theo policy π , ghi nhận tất cả return G_t mỗi lần trạng thái s xuất hiện, rồi lấy trung bình cộng.

5. Thuật toán ước lượng hàm giá trị (Monte Carlo Prediction Algorithm)

Mặc dù slide chỉ nhắc "Algorithm for estimating the state value function of a policy" mà không đưa chi tiết code, thông thường quy trình Monte Carlo prediction cho episodic tasks như sau:

1. Khởi tạo:

- Cho một chính sách cố định π .
- Khởi tạo $V(s)$ (ước lượng hàm giá trị) cho mọi $s \in S$ (có thể ban đầu gán 0).
- Tạo hai cấu trúc dữ liệu:
 - **ReturnsSum[s] = 0** (tổng của tất cả các return đã thu được từ s).
 - **ReturnsCount[s] = 0** (số lần trạng thái s đã xuất hiện).

2. Lặp (thu thập nhiều episode):

- Sinh (simulate) một episode hoàn chỉnh bằng cách để agent chạy theo policy π cho đến khi trạng thái terminal. Ta thu được danh sách $(s_0, a_0, r_1, s_1, a_1, r_2, \dots, s_{T-1}, a_{T-1}, r_T, s_T)$.
- Tính return G_t cho mỗi thời điểm t trong episode:

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots + \gamma^{T-t-1} r_T.$$

- Đối với mỗi trạng thái s_t xuất hiện lần đầu (first-visit) hoặc mọi lần xuất hiện (every-visit), thực hiện:
 - **ReturnsSum[s_t] += G_t,**
 - **ReturnsCount[s_t] += 1,**
 - **V(s_t) = ReturnsSum[s_t] / ReturnsCount[s_t].**


3. Kết thúc khi đủ độ chính xác hoặc sau một số episode nhất định.

- Cách tiếp cận "first-visit MC" chỉ dùng return tính từ lần xuất hiện đầu tiên của s trong mỗi episode. Cách "every-visit MC" dùng tất cả lần xuất hiện. Cả hai đều hội tụ khi số mẫu đủ lớn, nhưng "first-visit" thường được ưu tiên do giảm sai số phụ.

6. Ví dụ tính toán Return – Trường hợp có hệ số chiết khấu (Discount Factor)

Slide đưa ví dụ: "Giả sử hệ số chiết khấu $\gamma = 0.05$, và một episode kết thúc ở time step thứ 5 với reward sequence là 3, 4, 7, 1, 2." Hãy tìm từng return G_0, G_1, \dots, G_4 .

• Cách tính:

-  Công thức chung:

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots + \gamma^{T-t-1} r_T, \quad \text{với } T = 5.$$

- Trong ví dụ, ta đánh số time step từ 0 đến 5 (nhưng slide nói “episode ending at time step five” với reward sequence gồm 5 phần tử nên có thể tần suất reward xuất hiện từ r_1 đến r_5). Giả sử:
 - $r_1 = 3, r_2 = 4, r_3 = 7, r_4 = 1, r_5 = 2$.
 - Thời điểm kết thúc $T = 5$.

Vậy:

- G_4 (return tại time step 4):

$$G_4 = r_5 = 2.$$

(Bởi vì khi $t = 4$, chỉ còn một phần tử reward là $r_5 = 2$.)

- G_3 (return tại $t = 3$):

$$G_3 = r_4 + \gamma r_5 = 1 + 0.05 \times 2 = 1 + 0.1 = 1.1.$$

- G_2 (return tại $t = 2$):

$$G_2 = r_3 + \gamma r_4 + \gamma^2 r_5 = 7 + 0.05 \times 1 + 0.05^2 \times 2 = 7 + 0.05 + 0.05^2 \times 2 = 7 + 0.05 + 0.0025 \times 2 = 7 + 0.05 + 0.005 = 7.055.$$

- G_1 (return tại $t = 1$):

$$G_1 = r_2 + \gamma r_3 + \gamma^2 r_4 + \gamma^3 r_5 = 4 + 0.05 \times 7 + 0.05^2 \times 1 + 0.05^3 \times 2.$$

Tính cụ thể:

- $0.05 \times 7 = 0.35,$
 - $0.05^2 \times 1 = 0.0025 \times 1 = 0.0025,$
 - $0.05^3 \times 2 = 0.000125 \times 2 = 0.00025.$
- $$\Rightarrow G_1 = 4 + 0.35 + 0.0025 + 0.00025 = 4.35275.$$

- G_0 (return tại $t = 0$):

$$G_0 = r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4 + \gamma^4 r_5.$$

Thay số:

- $r_1 = 3,$
- $\gamma r_2 = 0.05 \times 4 = 0.2,$
- $\gamma^2 r_3 = 0.0025 \times 7 = 0.0175,$
- $\gamma^3 r_4 = 0.000125 \times 1 = 0.000125,$
- $\gamma^4 r_5 = 0.00000625 \times 2 = 0.0000125.$

Tổng lại:

$$G_0 = 3 + 0.2 + 0.0175 + 0.000125 + 0.0000125 = 3.2176375.$$

• Ý nghĩa:

- Mỗi return G_t phản ánh **tổng reward** nhận được sau thời điểm t , trong đó reward càng xa càng bị giảm giá (discounted) theo γ .
- Khi γ rất nhỏ, các phần thưởng xa về sau hầu như bị bỏ qua; khi γ gần 1, agent quan tâm cả các reward ở xa tương lai.

7. Ví dụ ứng dụng trong Blackjack (Monte Carlo for Prediction)

Đây là một ví dụ kinh điển trong sách “Reinforcement Learning” (Sutton & Barto) để minh họa Monte Carlo prediction. Slide mô tả lược đồ:

1. Mô tả vấn đề (Problem formulation):

- Agent chơi bài Blackjack (gần giống với game thật, có hai chế độ: đếm điểm, có ace, dealer có lá bài ngửa, v.v.).
- Ta cố định một **chính sách** π đơn giản cho agent, ví dụ: “Nếu tổng bài < 20 thì hit, ngược lại stick.”
- Mục tiêu: ước lượng $V^\pi(s)$ – giá trị của mỗi trạng thái s dưới chính sách π .

2. Ví dụ cụ thể:

- Giả sử at time step đầu (trước khi hành động), agent đang ở trạng thái đầu tiên (state 0):
 - Agent’s card sum = 13** (không có usable ace),
 - Dealer’s visible card = 10.**
- Chính sách π (fixed) nói agent phải **hit** (rút thêm) khi tổng < 20. Agent rút được lá bài 7 → tổng thành 20.
- Theo policy, khi tổng = 20, agent **stick** (dừng, nhường dealer chơi).

- Dealer rút lá bài tiếp (giá trị 9) → dealer bust (quá 21) → agent thắng → reward = +1 (nếu thắng, giả sử reward là +1; nếu thua -1; hòa 0).
3. **Xác định các trạng thái trên đường đi (episode):**
- Gọi trạng thái A là trạng thái 20 (không có ace, dealer lá 10) – đây là **last non-terminal state** (trước khi stakeholder kết thúc vòng của agent).
 - Trạng thái B là trạng thái 13 (không có ace, dealer lá 10) – đây là **second last non-terminal state** (lúc agent có tổng 13).
4. **Cập nhật return và ước lượng giá trị:**
- Khi episode kết thúc với reward +1 (agent thắng), ta gán $G = +1$ cho các trạng thái trên đường đi (theo quy tắc “every-visit” hoặc “first-visit”).
 - Cụ thể:
 - Đối với trạng thái A (20, no usable ace, dealer 10): ta thêm “+1” vào tập return-list cho A , rồi tính $V(A)$ bằng trung bình các giá trị trong list.
 - Đối với trạng thái B (13, no usable ace, dealer 10): cũng thêm “+1” vào list return-list cho B , rồi ước lượng $V(B)$ = trung bình.
 - Nếu nhiều episode khác xuất hiện trạng thái A hoặc B , ta cứ tiếp tục cộng vào list, rồi cập nhật trung bình. Khi số episode đủ lớn, $V(s)$ hội tụ về giá trị chính xác (theo kỳ vọng).
5. **Ý nghĩa:**
- Ví dụ này minh họa rõ cách Monte Carlo “lấy mẫu” giá trị bằng cách chạy nhiều episode (mỗi episode mô phỏng đầy đủ một ván Blackjack), thu thập reward cuối cùng, sau đó gán return đó cho tất cả các trạng thái đã trải qua trong episode.
 - Một điểm quan trọng: với chính sách cố định, xác suất đến các trạng thái khác nhau có thể khác nhau, nên tần suất trạng thái s xuất hiện cũng khác. Monte Carlo prediction điều chỉnh cho đúng bằng cách chỉ lấy trung bình trên số lần trạng thái đó xuất hiện.

8. Ví dụ ứng dụng trong môi trường Grid World

Slide đề cập:

“Example: Monte Carlo to estimate the state-value function for a simple grid world environment.”

Dù slide không trình bày chi tiết, thông thường ý như sau:

- Grid World:**
 - Là một môi trường lưới ($N \times M$), mỗi ô (cell) là một trạng thái.
 - Agent có thể di chuyển lên, xuống, trái, phải hoặc theo các luật di chuyển cho trước.
 - Ở một số ô đặc biệt (terminal states), khi agent đến, episode kết thúc và agent nhận được reward (có thể +1 hoặc -1 hoặc 0).
- Chính sách (Policy) cố định π :**
 - Ví dụ một chính sách random (agent chọn ngẫu nhiên một trong bốn hướng, đều nhau).
 - Mục tiêu: ước lượng $V^\pi(s)$ cho mọi ô s .
- Thực hiện Monte Carlo prediction:**
 - Lặp lại nhiều lần:
 - Khởi vị ở trạng thái bắt đầu (có thể là một ô nhất định), rồi cho agent di chuyển (như policy quy định) cho đến khi đến terminal hoặc đạt giới hạn max steps.
 - Ghi lại dãy $(s_0, a_0, r_1), (s_1, a_1, r_2), \dots, (s_{T-1}, a_{T-1}, r_T), s_T$.
 - Tính return G_t cho từng thời điểm t , rồi cập nhật trung bình cho mỗi trạng thái s_t .
 - Kết quả sau nhiều episode: ta có một bảng ước lượng $V(s)$ cho tất cả ô s .
- Ứng dụng:**
 - Tính $V(s)$ giúp đánh giá xem ô nào “tốt” (giá trị cao) hay “xấu” (giá trị thấp) nếu tuân theo policy hiện tại.
 - Từ đó, ta có thể dùng $V(s)$ làm cơ sở cho việc cải thiện policy (policy improvement) trong các thuật toán tích hợp với Monte Carlo.

9. Tóm tắt lại các nội dung chính (Summary)

Cuối slide có phần **Summary** tóm lược:

- Monte Carlo methods ước lượng hàm giá trị** (value function) bằng cách lấy trung bình trên các return được lấy mẫu (sampled returns).
- Những bài toán có thể giải bằng Monte Carlo methods:** thường là các môi trường có cấu trúc episodic, có thể lấy mẫu được các trajectory hoàn chỉnh.
- Sử dụng Monte Carlo prediction để ước lượng $V^\pi(s)$** cho một chính sách cố định π . Sau khi thu thập đủ số mẫu, giá trị ước lượng tiệm cận giá trị thực.

10. Kết luận

- **Ưu điểm của Monte Carlo:**
 - Không cần biết mô hình chuyển tiếp (transition dynamics) $P(s', r|s, a)$ của môi trường.
 - Đơn giản, dễ hiểu: chỉ cần "chạy thử" (simulate) agent với policy và ghi nhận kết quả.
- **Hạn chế:**
 - Chỉ áp dụng cho **episodic tasks** (episode phải kết thúc để xác định return).
 - Cần số lượng sample (episode) lớn để giảm sai số.
 - Đối với những trạng thái ít xuất hiện, phải chờ rất lâu mới có số mẫu đủ tốt.
- **Các bước tiếp theo trong học tăng cường:**
 - Từ Monte Carlo prediction, ta có thể phát triển Monte Carlo control (policy iteration tích hợp hành động, policy improvement).
 - Sau này, có thể xem xét các phương pháp không theo episode (bootstrap) như Temporal-Difference (TD) Learning, Q-Learning, SARSA, v.v., để ước lượng/học chính sách một cách hiệu quả hơn.