Monte Carlo cho Control – Khái niệm cơ bản

Khác với **Monte Carlo Prediction** (chỉ ước lượng giá trị trạng thái $V(s)$ cho một chính sách cố định), **Monte Carlo for Control** hướng đến ước lượng giá trị của **cặp (trạng thái, hành động)** $Q(s,a)$, từ đó tìm ra chính sách **greedy** (cải thiện) dựa trên Q.

- Ở phần prediction, ta đã biết chính sách π cố định, và chỉ tính $V^{\pi}(s)$.
- Ở phần Control, mục tiêu là **tìm chính sách tối ưu** π^{*}. Thường quy trình sẽ lặp:
  1. Với chính sách hiện thời π, dùng Monte Carlo để ước lượng $Q^{\pi}(s,a)$.
  2. Cập nhật chính sách thành $\pi'(s) = \arg\max_a Q^{\pi}(s,a)$ (làm cho chính sách "greedy").
  3. Lặp lại cho đến khi hội tụ.

Nhưng để đảm bảo quá trình **khám phá** không dừng hẳn (vì nếu luôn "greedy", agent sẽ chỉ chọn hành động có giá trị lớn nhất của Q hiện tại và có thể bỏ qua hành động tốt hơn nhưng chưa thử), ta cần thêm cơ chế **exploration**. Một trong những cách đơn giản nhất là "Exploring Starts" (bắt đầu ngẫu nhiên mỗi episode từ một cặp (s,a)).

---

3. Ước lượng hàm giá trị hành động (Action-Value) bằng Monte Carlo

3.1. Khái niệm Action-Value

- **State-value** $V^{\pi}(s)$ là giá trị kỳ vọng của trạng thái s dưới chính sách π.
- **Action-value** $Q^{\pi}(s,a)$ là giá trị kỳ vọng khi agent bắt đầu tại trạng thái s, thực hiện hành động a ngay lập tức, rồi sau đó tuân theo chính sách π cho đến khi kết thúc episode. Cụ thể:

$$Q^{\pi}(s,a) \;=\; \mathrm{E}[\, G_t \;\big|\; S_t = s,\; A_t = a \,].$$

  Trong đó G_t là tổng discounted reward từ thời điểm t cho đến kết thúc episode.

3.2. Monte Carlo để ước lượng $Q(s,a)$

Quy trình Monte Carlo ước lượng Q rất giống với ước lượng V, chỉ khác ở chỗ:

1. Trong mỗi episode, ta theo policy π (nhưng phải có cách để chắc chắn rằng mỗi cặp (s,a) sẽ được thử ít nhất đôi lần—ví dụ "Exploring Starts").
2. Khi một cặp (s_t, a_t) xuất hiện trong episode, ta tính return G_t bắt đầu ngay sau khi agent đã thực hiện a_t.
3. Lưu G_t vào danh sách returns của (s_t, a_t).
4. Ước lượng:

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

với $N(s,a)$ là số lần cặp (s,a) xuất hiện (theo quy tắc first-visit hoặc every-visit).

Điểm quan trọng: nếu cặp (s,a) nào bị policy bỏ qua (never-visited), ta không có dữ liệu return để ước lượng, dẫn đến $Q(s,a)$ sai lệch. Vì vậy, **bắt buộc phải duy trì "exploration"** để mọi cặp (s,a) ít nhất được thử vài lần.

---

4. Vấn đề duy trì khám phá (Exploration) trong Monte Carlo

4.1. Tại sao cần exploration?

- Giả sử chúng ta đang học hành vi cho một agent với policy ban đầu π.
- Nếu policy π ở một trạng thái s **không bao giờ chọn** hành động a, thì ta không bao giờ thu được bất kỳ return nào từ (s,a). Kết quả là $Q(s,a)$ vẫn không được cập nhật, và có thể có một hành động tốt hơn, nhưng ta không biết.
- Vấn đề tương tự xuất hiện khi policy quá "greedy" ngay từ đầu: agent chỉ chọn hành động được đánh giá cao nhất, bỏ qua các hành động khác, nên không có dữ liệu để so sánh, dẫn đến local optimum.

4.2. Ví dụ minh họa

Slide đưa ví dụ thực tế: "Khi bạn đi bộ về nhà theo con đường quen thuộc. Gần đây người ta xây con đường mới. Nếu bạn không bao giờ thử đi con đường đó, bạn sẽ không bao giờ biết nó có thể nhanh hơn hoặc tiện hơn."

Tương tự trong RL, nếu agent không thử hành động mới (khác policy đang có), nó sẽ không biết có cách thực hiện "hit" hay "stick" nào tốt hơn – nên phải có cách buộc agent phải "thử" ít nhất đôi lần.

---

5. Exploring Starts

Exploring Starts (ES) là một cách đơn giản để đảm bảo "exploration" đầy đủ trong Monte Carlo Control. Ý tưởng:

1. **Mỗi episode được khởi tạo** bằng cách chọn ngẫu nhiên một **cặp** (trạng thái s, hành động a) ban đầu, với phân phối nào đó (thường là đều nhau trên tất cả (s,a)).
2. Sau khi chọn được (s_0,a_0), agent thực hiện a_0 ngay lập tức, sau đó tiếp tục theo policy π hiện thời (thường policy này sẽ là epsilon-greedy để thêm cơ hội exploration, hoặc lúc đầu hoàn toàn greedy vì exploring starts đã chịu trách nhiệm mở rộng).
3. Kết quả: mỗi cặp (s,a) cuối cùng sẽ được "thử" (visited) ít nhất một lần (với xác suất > 0). Nhờ đó, việc ước lượng $Q(s,a)$ có thể diễn ra cho tất cả cặp.

Nhược điểm:

- Cần phải biết trước tập trạng thái S và tập hành động A để có thể khởi tạo ngẫu nhiên. (Trong một số bài toán rất lớn hoặc vô hạn không thể thực hiện ES dễ dàng.)
- ES chỉ đảm bảo cho mỗi episode có một cặp (s, a) ban đầu khác nhau, nhưng không ngăn policy sau đó hoàn toàn "đóng" vào một hành động, vì vậy đôi khi người ta vẫn kết hợp thêm epsilon-greedy.

---

6. Quy hoạch chính sách tổng quát (Generalized Policy Iteration – GPI)

GPI là khung chung cho hầu hết các thuật toán RL: nó bao gồm hai bước lặp, song hành nhưng không nhất thiết đồng thời:

1. **Policy Evaluation (Đánh giá chính sách)**: ước lượng V^π (trong prediction) hoặc Q^π (trong control) cho chính sách hiện thời π.
2. **Policy Improvement (Cải thiện chính sách)**: dựa vào giá trị ước lượng, xây dựng một chính sách mới π' sao cho π' luôn **greedy** với giá trị hiện tại:

$$\pi'(s) = \arg\max_{a \in A(s)} Q^\pi(s, a).$$

Khi lặp lại hai bước này, ta sẽ thu được dãy các chính sách $\pi_0, \pi_1, \pi_2, \ldots$ mà từng bước luôn không kém hơn (monotonic improvement) về mặt giá trị kỳ vọng. Cuối cùng, luật GPI bảo đảm hội tụ đến chính sách tối ưu π^*.

Ở phần Monte Carlo for Control, ta sẽ:

- Dùng Monte Carlo **prediction** để ước lượng $Q^\pi(s, a)$ ở mỗi episode.
- Sau khi thu thập đủ dữ liệu về Q, **cập nhật policy** thành greedy (hoặc epsilon-greedy để duy trì exploration).
- Lặp lại cho đến khi ổn định.

---

7. Thuật toán Monte Carlo cho Control (MC Control)

Dưới đây là phiên bản tổng quát của **Monte Carlo Control with Exploring Starts** (MC-ES). Slide không show code chi tiết, nhưng quy trình cơ bản là:

1. **Khởi tạo**
   - Cho một chính sách ban đầu π (có thể ngẫu nhiên hoàn toàn) trên mọi cặp (s, a).
   - Khởi tạo các bộ đếm $N(s, a) = 0$ và tổng trả về $ReturnsSum(s, a) = 0$ cho mọi (s, a).
   - Khởi tạo ước lượng $Q(s, a)$ (có thể bằng 0) cho mọi (s, a).
2. **Lặp qua nhiều episode**
   - **Chọn ngẫu nhiên một cặp (s_0, a_0)** cho exploring starts:

$$s_0 \sim \text{RandomState}(), \quad a_0 \sim \text{RandomAction}(s_0).$$

- Tạo một danh sách `episode = [(s_0,a_0,r_1), (s_1,a_1,r_2), …, (s_{T-1},a_{T-1},r_T)]` bằng cách:
  - Thực hiện a_0 tại s_0, quan sát reward r_1 và next-state s_1.
  - Với mỗi bước tiếp theo $t \geq 1$, chọn hành động a_t theo $\pi(s_t)$, quan sát r_{t+1}, s_{t+1}, cho đến khi s_T là terminal.
- Với mỗi cặp (s_t, a_t) xuất hiện trong episode (theo first-visit hoặc every-visit):
  1. Tính return $G_t = \sum_{k=t}^{T-1} \gamma^{k-t} r_{k+1}$.
  2. Cập nhật tổng ReturnsSum và số lần xuất hiện:

$$ReturnsSum(s_t, a_t) \mathrel{+}= G_t, \quad N(s_t, a_t) \mathrel{+}= 1.$$

  3. Cập nhật ước lượng:

$$Q(s_t, a_t) = \frac{ReturnsSum(s_t, a_t)}{N(s_t, a_t)}.$$

- **Cải thiện chính sách**:

$$\pi(s) \leftarrow \arg\max_a Q(s, a), \quad \forall s \text{ đã xuất hiện trong episode.}$$

  Trong phiên bản thuần túy ES, policy sau đó hoàn toàn "greedy" (không còn exploration) vì exploring starts đã bảo đảm mỗi episode bắt đầu với một (s, a) khác. Tuy nhiên, trong thực tế người ta thường thay bằng **epsilon-greedy**:

$$\text{với xác suất } \epsilon, \text{ chọn ngẫu nhiên một action}, \quad \text{với xác suất } 1-\epsilon, \text{ chọn } \arg\max_a Q(s, a).$$

3. **Lặp** cho đến khi policy không thay đổi hoặc đến một số episode nhất định.

Mỗi lần lặp là một **chính sách đánh giá** (policy evaluation) bằng MC, sau đó là một **chính sách cải thiện** (policy improvement) bằng greedification.

---

8. Ví dụ Blackjack với Monte Carlo Control

Slide đưa ra một ví dụ cụ thể: "Use Monte Carlo with Exploring Starts để huấn luyện agent chơi Blackjack". Các bước quan trọng trong ví dụ như sau:

8.1. Mô hình Blackjack đơn giản

- Mỗi state s được mô tả bằng bộ ba (agent's current sum, dealer's visible card, flag usable ace hay không).
- Action có hai: **hit** (rút thêm bài) hoặc **stick** (dừng).
- Reward:
  - +1 nếu agent thắng (dealer bust hoặc có sum cao hơn mà không vượt 21).
  - 0 nếu hòa (tie).

    - –1 nếu thua.
- Episode kết thúc khi agent stick (sau đó dealer tự draw cho đến điểm ≥ 17) hoặc agent bust (sum > 21).

8.2. Chính sách ban đầu và Exploring Starts

- **Chính sách ban đầu** π:
    - Nếu agent's sum < 20 thì hit, nếu ≥ 20 thì stick.
    - Với ES, mỗi episode ta sẽ **bỏ qua** quyết định "hit" hay "stick" tại state đầu, mà **chọn ngẫu nhiên** action đầu tiên.
- Ví dụ:
    - Giả sử lúc khởi đầu, agent's sum = 13, no usable ace, dealer's visible card = 8. Theo policy ban đầu, agent sẽ hit (vì 13 < 20). Nhưng với ES, ta random rằng action đầu "stick" hoặc "hit" đều có xác suất 50–50. Giả sử random chọn "hit".

8.3. Một episode mẫu

1. **State ban đầu** $B = (13,\ \text{no usable ace},\ 8)$. Với ES, giả sử ta chọn action $a_B = \text{hit}$.
2. Agent rút được lá bài 7: B chuyển thành state $A = (20,\ \text{no usable ace},\ 8)$.
3. Tại state A, theo policy hiện tại (không còn ES vì ES chỉ áp dụng cho action đầu), agent thấy sum = 20 và sẽ **stick**.
4. Sau khi agent dừng, dealer draw → dealer rút lá 9 → tổng > 21 → dealer bust → agent thắng → Reward cuối cùng $r = +1$. Episode kết thúc.

8.4. Cập nhật giá trị hành động

- **Cập nhật cho** $A = (20,\ \text{no usable ace},\ 8)$:
    - Lần đầu tiên $(A,\ action = \text{stick})$ xuất hiện (vì action tại A theo policy là stick). Return $G = +1$.
    - Do chưa có dữ liệu trước, ta gán $ReturnsSum(A, \text{stick}) = 1$, $N(A, \text{stick}) = 1 \rightarrow Q(A, \text{stick}) = 1/1 = 1$.
- **Cập nhật cho** $B = (13,\ \text{no usable ace},\ 8)$:
    - Lần đầu tiên $(B,\ action = \text{hit})$ xuất hiện (ES đã chọn hit ban đầu). Return $G = +1$ (vì cuối cùng reward = +1).
    - Gán $ReturnsSum(B, \text{hit}) = 1$, $N(B, \text{hit}) = 1 \rightarrow Q(B, \text{hit}) = 1$.
- Lúc này, với state B, ta chưa thử hành động stick (ES chỉ thử hit), nên $Q(B, \text{stick})$ vẫn mặc định $= 0$.
- **Cải thiện policy cho state** B: so sánh $Q(B, \text{hit}) = 1$ vs. $Q(B, \text{stick}) = 0$.
  → Thấy "hit" có giá trị lớn hơn, vậy policy $\pi(B)$ đổi thành "hit" (trước đó policy cũ cũng là hit vì sum < 20, nhưng đây chỉ là ví dụ minh họa).
- Nếu chúng ta lặp lại nhiều episode, dần dần ước lượng Q ở mọi state (bao gồm cả trường hợp có usable ace) sẽ được cải thiện, và policy cũng sẽ thay đổi tương ứng. Khi chạy đủ lâu, ta thu

được **chính sách tối ưu** cho Blackjack (như slide có minh hoạ: policy khi có usable ace sẽ "aggressive", khi không có thì dựa nhiều vào lá bài dealer).

8.5. Kết quả sau nhiều episode

- Sau khi huấn luyện "rất lâu" với ES và policy iteration, ta thu được một ma trận Q rất gần với giá trị thực.
- Từ đó, policy "greedy" tương ứng là policy tối ưu:
  - Nếu có usable ace: agent "hit" cho đến khi sum ~ 19
  - Nếu không: policy phụ thuộc vào lá dealer → thường "stick" nếu sum ≥ 13 khi dealer có lá nhỏ (2 hoặc 3), v.v.