2. Khái niệm cơ bản về Q-Learning

2.1. Học giá trị hành động tối ưu

- **Action-Value Function** $Q^{\pi}(s,a)$ là kỳ vọng tổng discounted reward khi bắt đầu tại trạng thái s, thực hiện hành động a, rồi tiếp tục theo chính sách π.
- **Q-Learning** nhắm tới ước lượng trực tiếp **hàm giá trị hành động tối ưu**:

$$Q^{*}(s,a) \;=\; \max_{\pi} Q^{\pi}(s,a),$$

  hay còn gọi là giá trị cao nhất có thể đạt được nếu từ s ta cứ "ra quyết định tốt nhất" ở mỗi bước sau đó.

2.2. Off-policy

- Q-Learning là **off-policy** vì:
  - Agent **thực thi** một **behavior policy** (thường là ε-greedy để vẫn còn khám phá),
  - Nhưng **ước lượng** thì lại nhắm đến **target policy** tối ưu (luôn chọn action có Q cao nhất).
- Đây là điểm khác với SARSA (on-policy), vốn chỉ ước lượng Q^{π} của chính π mà nó đang chạy.

---

3. Bellman Optimality Equation cho Q^{*}

Bellman Optimality cho giá trị hành động phát biểu:

$$Q^{*}(s,a) = \mathrm{E}[R_{t+1} + \gamma \max_{a'} Q^{*}(S_{t+1},a') \;\big|\; S_t = s,\; A_t = a].$$

Nghĩa là: **giá trị tối ưu** của (s,a) bằng kỳ vọng reward ngay lập tức R_{t+1} cộng discounted giá trị hành động tối ưu tại trạng thái kế tiếp, khi chọn luôn action tốt nhất.

---

4. Quy tắc cập nhật Q-Learning

4.1. Update rule

Từ Bellman Optimality, Q-Learning sử dụng công thức cập nhật **bootstrap** (TD) mỗi khi gặp transition (s,a,r,s'):

$$Q(s,a) \;\leftarrow\; Q(s,a) \;+\; \alpha[\, r + \gamma \max_{a'} Q(s',a') \;-\; Q(s,a)],$$

trong đó:

- α là **learning rate** (tốc độ cập nhật, thường $0 < \alpha \leq 1$).
- γ là **discount factor** (hệ số chiết khấu, $0 \leq \gamma < 1$).
- r là reward quan sát ngay sau khi thực thi a ở s.
- $\max_{a'} Q(s', a')$ là phần **bootstrap**, tức ước lượng giá trị tối ưu tại trạng thái mới s'.

4.2. Giải thích trực quan

1. **TD target**:

$$\underbrace{r + \gamma \max_{a'} Q(s', a')}_{\text{TD target}}$$

   là "giá trị kỳ vọng mới" mình muốn $Q(s, a)$ hướng tới.

2. **TD error**:

$$\delta = r + \gamma \max_{a'} Q(s', a') - Q(s, a).$$

   Nếu $\delta > 0$, nghĩa là thực tế tốt hơn ước tính, nên tăng $Q(s, a)$. Ngược lại giảm.

3. **Cập nhật**:
   Lấy một phần α của δ để điều chỉnh, tránh dao động quá nhiều.

---

5. Pseudo-code của Q-Learning

plaintext Sao chép Chỉnh sửa

```
Input: learning rate α, discount factor γ, exploration rate ε Initialize Q(s,a) arbitrarily (e.g. 0) for all s∈S, a∈A(s) for episode = 1 to M do Initialize state s repeat (for each step of episode): • Với xác suất ε: chọn a ← random action in A(s) Ngược lại: chọn a ← argmax_{a'} Q(s, a') (greedy) • Thực thi a, quan sát reward r và trạng thái kế tiếp s' • Cập nhật: Q(s,a) ← Q(s,a) + α [r + γ max_{a''} Q(s',a'') – Q(s,a)] • s ← s' until s is terminal end for
```

- **ε-greedy** đảm bảo exploration: với xác suất ε chọn ngẫu nhiên, còn lại exploit (chọn action tốt nhất).
- Khi $\varepsilon \to 0$ và đủ nhiều episode, Q sẽ hội tụ về Q^* và policy greedy trên đó chính là π^*.

---

6. Ví dụ cụ thể: FrozenLake

6.1. Mô tả môi trường

- **FrozenLake-v0** là một grid 4×4, mỗi ô có thể là:
  - **S** (Start),
  - **F** (Frozen, an toàn),
  - **H** (Hole, thua ngay),
  - **G** (Goal, thắng +1).
- **State**: vị trí ô hiện tại (tổng 16 trạng thái).
- **Actions**: {Left, Down, Right, Up}.
- **Reward**: chỉ +1 khi vào ô G; các bước khác reward = 0; vào H dừng với reward 0.

6.2. Khởi tạo Q-table

- Tạo mảng Q kích thước 16×4, khởi bằng 0.

6.3. Một bước cập nhật

Giả sử ở state $s = 5$, Q(5,:) ban đầu = [0,0,0,0].

- Chọn action theo ε-greedy: giả sử ε=0.1, ta random → "Right" (action index = 2).
- Thực thi, nhận $r = 0$, chuyển sang state $s' = 6$.
- Tính $\max_{a'} Q(6, a')$. Ban đầu tất cả Q(6,*)=0, nên max=0.
- Cập nhật:

$$Q(5,2) \leftarrow 0 + \alpha\,[0 + \gamma \cdot 0 - 0] = 0.$$

  Lần sau khi Q(6,*) không còn 0, các cập nhật sẽ dần thay đổi.

6.4. Qua nhiều episode

- Khi agent tình cờ đi tới G (state=Goal) sau một chuỗi steps, ở bước cuối ta sẽ update ví dụ:
  - Chuỗi giả sử: $s = 14 \xrightarrow{a} s' = 15(G)$ nhận $r = 1$.
  - Ta có $\max_{a'} Q(15, a') = 0$ (terminal).
  - Cập nhật:

$$Q(14,a) \leftarrow Q(14,a) + \alpha\,(1 + 0 - Q(14,a)).$$

- Về lâu dài, Q sẽ tiệm cận giá trị kỳ vọng số bước tối ưu để đến G từ mỗi ô, và policy greedy trên Q chính là đường đi ngắn nhất, tránh các lỗ.

---

7. So sánh Q-Learning vs SARSA

Đặc điểm	SARSA (on-policy)	Q-Learning (off-policy)
Chính sách	ước lượng Q^{π} của chính policy đang chạy (ε-greedy).	ước lượng Q^{*} (luôn "học" chính sách tối ưu) dù action chọn the ε-greedy.
Update rule	$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma\, Q(s',a') - Q(s,a)]$ với $a' \sim \pi(s')$.	$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma\, \max_{a'} Q(s',a') - Q(s,a)]$.
Khám phá/khai thác	Chọn a' của bước sau theo cùng ε-greedy policy → update "an toàn" theo hành vi.	Update theo max Q (không thụ tế chọn) → agent có thể "học" t các hành động nó chưa từng thực thi.
Hội tụ	Hội tụ về chính sách ε-optimal (an toàn hơn trong môi trường rủi ro).	Hội tụ về Q^{*} (giá trị và chính sách tối ưu), nhưng đòi hỏi điề kiện thăm đủ tất cả (s,a) (coverage).
Tính an toàn	"An toàn" hơn vì update và chọn action cùng tương thích policy hiện hành.	Có thể "học" giá trị của các action rủi ro mà agent chưa từ dùng → đôi khi dẫn đến hành v mạo hiểm trong training.

8. Ứng dụng & Lưu ý

- Q-Learning rất phổ biến trong robotics, game-playing (ví dụ Atari), autonomous navigation…
- **Điều kiện** để hội tụ lý thuyết:
  1. Learning rate $\alpha_t(s,a)$ phải giảm dần (satisfy Robbins-Monro).
  2. Mọi cặp (s,a) phải được thăm vô hạn lần (policy ε-greedy với $\varepsilon > 0$).
  3. Discount $\gamma < 1$ hoặc môi trường episodic.

9. Tóm tắt (Summary)

- **Q-Learning**: thuật toán off-policy TD control, ước lượng trực tiếp $Q^{*}(s,a)$.
- **Cập nhật** dựa trên Bellman Optimality:

$$Q \leftarrow Q + \alpha[r + \gamma \max_{a'} Q - Q].$$

- **Khác SARSA** ở chỗ lấy max action-value thay vì sử dụng action thực thi tiếp theo.
- **Ví dụ FrozenLake** cho thấy Q-table dần hội tụ đến giá trị hành động tối ưu, từ đó lấy đường đi ngắn nhất và an toàn nhất.