# 1. GIỚI THIỆU VỀ HỌC TĂNG CƯỜNG - RL

Học tăng cường (Reinforcement Learning – RL) là một lĩnh vực quan trọng trong trí tuệ nhân tạo, tập trung vào việc huấn luyện một tác nhân (agent) học cách đưa ra chuỗi hành động trong môi trường để tối đa hóa phần thưởng tích lũy. RL được ứng dụng trong nhiều lĩnh vực: chơi game (AlphaGo, Dota2 AI), robot tự hành, tối ưu chuỗi sản xuất, tài chính, y học cá nhân hóa…

Khác với học có giám sát (supervised learning), RL không có nhãn cố định cho từng dữ liệu. Thay vào đó, agent phải khám phá (exploration) và khai thác (exploitation) thông tin trong môi trường để cải thiện chính sách hành động.

# 2. MẠNG NƠ-RON NHÂN TẠO

## 2.1 Yêu cầu trước khi làm thí nghiệm

Yêu cầu trước khi thực hành:

- Kiến thức nền tảng: đại số tuyến tính, xác suất – thống kê, học có giám sát.
- Kỹ năng lập trình: Python, NumPy, hiểu cơ bản TensorFlow/PyTorch.
- Công cụ: Python 3.x, Jupyter Notebook, thư viện gym (OpenAI Gym).
- Dữ liệu / môi trường: sử dụng các môi trường RL chuẩn như CartPole, MountainCar, Atari.

## 2.2 Mục đích của phần thí nghiệm

Mục đích của phần thí nghiệm:

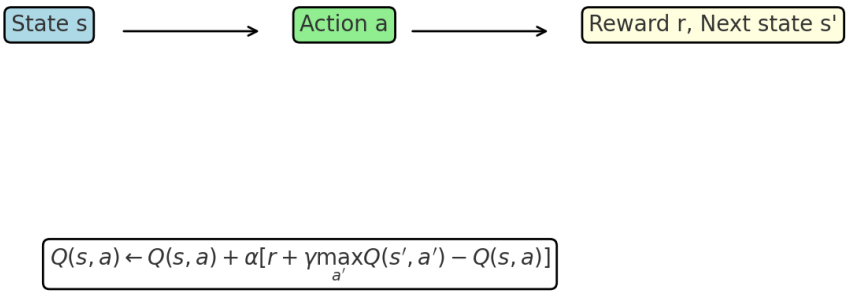
- Hiểu rõ khái niệm Markov Decision Process (MDP).
- Nắm được các hàm giá trị $V^{\pi}$(s), $Q^{\pi}$(s,a)
- Làm quen với các phương trình Bellman và ý nghĩa tối ưu.
- Áp dụng các thuật toán Q-learning, SARSA, Policy Gradient, Actor-Critic.
- Biết các kỹ thuật regularization và exploration trong RL.

## 2.3 Tóm tắt lý thuyết

### 2.3.1 Định nghĩa

| Định nghĩa | Công thức |
|---|---|
| Mô hình RL được mô tả dưới dạng Markov Decision Process: tập trạng thái S, tập hành động A, xác suất chuyển trạng thái P, phần thưởng R, hệ số chiết khấu γ. | $MDP = (S, A, P, R, \gamma)$ |
| Hàm giá trị trạng thái: kỳ vọng phần thưởng tích lũy khi bắt đầu từ trạng thái sss và theo chính sách π. | $V^{\pi}(s) = E_{\pi}[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \mid S_0 = s]$ |
| Hàm giá trị hành động: kỳ vọng phần thưởng tích lũy khi bắt đầu từ trạng thái s, chọn hành động aaa và theo chính sách π. | $Q^{\pi}(s, a) = E_{\pi}[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \mid S_0 = s, A_0 = a]$ |

### 2.3.2 Thuật toán RL

| | |
|---|---|
| Cập nhật Q-learning: học chính sách tối ưu bằng cách cập nhật giá trị Q. | $Q(s,a) \leftarrow Q(s,a) + \alpha \left[ r + \gamma \max_{a'} Q(s', a') - Q(s,a) \right]$   |
| Cập nhật SARSA: học theo chính sách đang thực hiện, | $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)]$ |

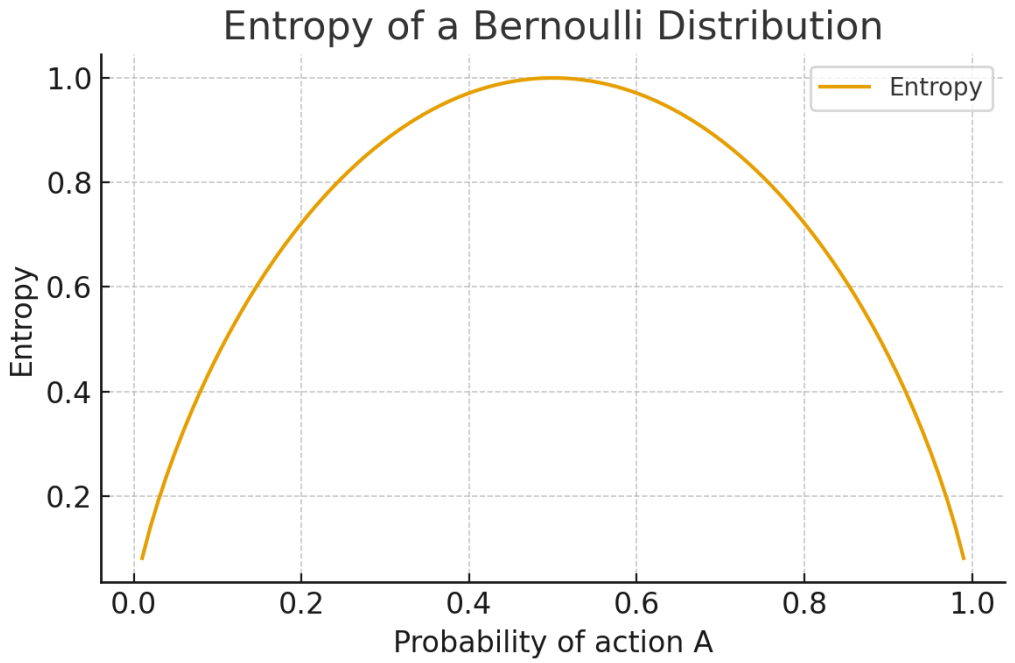
| | |
|---|---|
| khác với Q-learning. | |
| Actor-Critic: sai số TD (Temporal Difference) để cập nhật Critic. | $\nabla J(\theta) = E_\pi[\nabla_\theta \log \pi_\theta (a\|s)\, Q^\pi(s,a)]$ |
| Hàm Advantage: đo lường mức độ tốt hơn trung bình của hành động a tại trạng thái s. | $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$ |



### 2.3.3 REGULARIZATION & EXPLORATION

*2.3.3.1. Entropy Regularization:*

| | |
|---|---|
| [CT1] | $H(\pi(s)) = -\sum_a \pi(a\|s) \log \pi (a\|s)$ |




*2.3.3.2. Epsilon-Greedy Policy*

$$\pi(a|s) = (1 - \epsilon + \tfrac{\epsilon}{|A|})1\left[a = \arg\max_a Q(s,a)\right] + \tfrac{\epsilon}{|A|}$$

*2.3.3.3. Softmax Exploration:*

| | |
|---|---|
| [CT2] | $\pi(a\|s) = \frac{e^{Q(s,a)/\tau}}{\sum_{a'} e^{Q(s,a')/\tau}}$ |

*2.3.3.4. Weight Decay (L2)*

| | |
|---|---|
| [CT3] | $\theta_{t+1} = (1 - \eta\lambda)\theta_t - \eta\nabla_\theta L(\theta_t)$ |

### 2.3.4. Phương pháp học nâng cao trong RL

| | |
|---|---|
| hàm advantage: đo lường mức độ "tốt hơn trung bình" của hành động a tại trạng thái s. được dùng trong actor-critic và policy gradient. | $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$ |

| | |
|---|---|
| sai số td (temporal difference): dùng để cập nhật critic trong actor-critic. | $\nabla J(\theta) = E_{s,a\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a\|s)\, A^\pi(s,a)]$ |
| hàm mất mát trong proximal policy optimization (ppo): giới hạn cập nhật chính sách để tránh bước nhảy quá lớn. | $L^{ppo}(\theta) = E_t[min!(r_t(\theta)A_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)]$ |
| *MSE* thường dùng cho hồi quy; nhạy cảm với ngoại lai do bình phương sai số. | $L = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y_i})^2$ |
| *Hinge loss* dùng trong SVM/NN phân biệt biên cứng; khuyến khích lề phân tách lớn. | $L = \sum_{i=1}^{N} max(0, 1 - y_i\hat{y_i})$ |