BÁO CÁO BÀI TẬP LỚN

Phùng Hữu Khoa - 24022370
Lê Tuấn Duy - 24022310
Phạm Lê Việt Đức - 24022296

Hà Nội, ngày 22 tháng 10 năm 2025

Mục lục

Tóm tắt nội dung

Bài báo cáo này trình bày quá trình triển khai thuật toán Deep Q-Network (DQN) để huấn huyện tác tử thông minh (agent) có khả năng tự học chơi game trong hai môi trường của Gymnasium: `FlappyBird-v0` và `CartPole-v1`. Thuật toán được sử dụng trong bài báo cáo không chỉ bao gồm cài đặt DQN cơ bản mà còn tích hợp các cải tiến như Double DQN và Dueling DQN nhằm nâng cao hiệu suất và tính ổn định của quá trình học tăng cường (reinforcement learning). Đây là bản báo cáo trình bày ngắn gọn về nền tảng lý thuyết của các thuật toán, quá trình triển khai, cũng như kết quả thu được từ việc huấn luyện tác tử trong hai môi trường trên.

1 Giới thiệu

1.1 Bối cảnh và động lực

Học tăng cường (Reinforcement Learning - RL) là một nhánh quan trọng của trí tuệ nhân tạo, cho phép các tác tử (agents) học cách tự đưa ra quyết định thông qua tương tác với môi trường. Không giống như học có giám sát (supervised learning), RL không yêu cầu dữ liệu đầu vào có nhãn mà thay vào đó, các tác tử học thông qua việc nhận phản hồi dưới dạng phần thưởng (rewards) từ môi trường.
Thuật toán Deep Q-Network (DQN) được giới thiệu bởi DeepMind vào năm 2014 và đã đánh dấu một bước tiến trong lĩnh vực học tăng cường khi kết hợp mạng nơ-ron sâu với học tăng cường, cho phép các tác tử học được cách chơi nhiều trò chơi Atari với hiệu suất tương đương con người.
Với động lực từ những thành công của thuật toán DQN, nhóm chúng tôi quyết định triển khai và thử nghiệm thuật toán này với trò chơi Flappy Bird và CartPole.

1.2 Mục tiêu

Dự án bài tập lớn này nhằm mục tiêu:

i) Triển khai thuật toán DQN từ đầu thông qua thư viện PyTorch.

ii) Tích hợp các cải tiến: Double DQN và Dueling DQN để nâng cao hiệu suất và tính ổn định của quá trình học.

iii) Huấn luyện và đánh giá agent trên hai môi trường có độ phức tạp khác nhau.

iv) Phân tích ảnh hưởng của các siêu tham số (hyperparameters) đến hiệu suất của việc học.

2 Cơ sở lý thuyết

2.1 Học tăng cường - Markov Decision Process

Học tăng cường (Reinforcement Learning - RL) là bài toán mà một tác tử (agent) học cách tương tác với môi trường (environment) để tối đa hóa một tín hiệu phần thưởng (reward signal) theo thời gian. Do tính chất ra quyết định tuần tự (mỗi hành động hiện tại ảnh hưởng kết quả tương lai), ta thường mô hình hóa bài toán RL dưới dạng Quá trình quyết định Markov (MDP) để đảm bảo tính **chặt chẽ và tính được** — vì nó giả định rằng **tương lai chỉ phụ thuộc vào hiện tại** và **không** phụ thuộc vào quá khứ xa hơn.
Cụ thể, một Quá trình quyết định Markov (Markov Decision Process - MDP) được định nghĩa bởi bộ 5 thành phần $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma)$ trong đó:

- $\mathcal{S}$: Tập hợp không gian trạng thái (states) mà tác tử có thể gặp phải trong môi trường.
- $\mathcal{A}$: Tập hợp không gian hành động (actions) mà tác tử có thể thực hiện.
- $\mathcal{R}$: Hàm phần thưởng (reward function), được sử dụng để xác định phần thưởng nhận được khi thực hiện hành động trong một trạng thái cụ thể.
- $\mathcal{P}$: Hàm chuyển đổi trạng thái (state transition function) xác định xác suất chuyển từ trạng thái này sang trạng thái khác khi thực hiện một hành động.
- γ: Hệ số chiết khấu (discount factor), là một hằng số nằm trong khoảng $[0, 1]$, được dùng để giảm giá trị của các phần thưởng tương lai so với phần thưởng hiện tại.

Mục tiêu của tác tử, khi này, là tìm ra một chính sách (policy) tối ưu π^* (tức là quy tắc chọn hành động tốt nhất ở mỗi trạng thái) sao cho nó tối đa hóa giá trị kỳ vọng của tổng phần thưởng chiết khấu trong suốt quá trình tương tác với môi trường

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}.$$

2.2 Hàm giá trị và Phương trình Bellman

2.2.1 Hàm giá trị hành động

Để đánh giá một chính sách π nào đó, ta sử dụng khái niệm hàm giá trị hành động (action-value function) $Q^\pi(s, a)$. Hàm này biểu diễn tổng phần thưởng kỳ vọng mà tác tử sẽ nhận được kể từ trạng thái s khi thực hiện hành động a và sau đó tuân theo chính sách π:

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[G_t | S_t = s, A_t = a \right].$$

Mục tiêu cuối cùng của học tăng cường là tìm ra hàm giá trị hành động tối ưu $Q^*(s, a)$, đại diện cho tổng phần thưởng kỳ vọng tối đa mà tác tử có thể nhận được từ trạng thái s khi thực hiện hành động a và sau đó tuân theo chính sách tối ưu π^*.

2.2.2 Phương trình Bellman

Phương trình Bellman về cơ bản mô tả mối quan hệ giữa giá trị của một trạng thái và giá trị của các trạng thái kế tiếp nó. Trong bối cảnh Quá trình quyết định Markov (MDP), nó giúp xác định chính sách tối ưu bằng cách chia hàm giá trị thành phần thưởng ngay lập tức và giá trị của các trạng thái tiếp theo.
Đối với một chính sách π, phương trình kỳ vọng Bellman cho hàm giá trị trạng thái $V^\pi(s)$ được biểu diễn như sau được định nghĩa là:

$$V^\pi(s) = \mathbb{E}_\pi \left[R_{t+1} + \gamma V^\pi(S_{t+1}) | S_t = s \right].$$

Từ đó, ta có định nghĩa phương trình tối ưu Bellman cho hàm giá trị tôi ưu $V^*(s)$:

$$V^*(s) = \max_a \mathbb{E} \left[R_{t+1} + \gamma V^*(S_{t+1}) | S_t = s, A_t = a \right].$$

Hàm giá trị hành động tối ưu $Q^*(s, a)$ tương tự có thể được định nghĩa bằng cách sử dụng phương trình tối ưu Bellman cho hàm giá trị hành động:

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

Từ $Q^*(s, a)$, ta có thể suy ra chính sách tối ưu π^* bằng cách chọn hành động có giá trị cao nhất (greedy policy):

$$\pi^*(s) = \arg\max_a Q^*(s, a).$$

2.3 Deep Q-Network (DQN)

Với các bài toán có không gian trạng thái lớn hoặc liên tục, việc lưu trữ và cập nhật các giá trị $Q(s, a)$ bằng bảng (Q-table) trở nên không thực tế (bất khả thi). Để giải quyết vấn đề này, Deep Q-Network (DQN) sử dụng một mạng nơ-ron sâu với tham số θ để xấp xỉ hàm giá trị hành động $Q(s, a)$ bởi $Q(S, a, \theta)$.
Mạng nơ-ron được huấn luyện sao cho các giá trị dự đoán $Q(S_t, A_t, \theta)$ gần với mục tiêu y_{target} nhất có thể bằng cách tối thiểu hóa hàm mất mát (loss function):

$$L(\theta) = \mathbb{E}_{(s,a,r,s') \sim D} \left[\left(y_{\text{target}} - Q(S_t, A_t, \theta) \right)^2 \right],$$

trong đó mục tiêu y_{target} được tính toán dựa trên phương trình Bellman:

$$y_{\text{target}} = r + \gamma \max_{a'} Q(s', a'; \theta).$$

2.4 Double DQN

Double DQN là một cải tiến của DQN nhằm giảm thiểu xu hướng đánh giá quá cao (overestimation bias) Q-values của DQN. Trong Double DQN, việc lựa chọn hành động tốt nhất và đánh giá giá trị của hành động đó được tách biệt bởi hai mạng nơ-ron khác nhau:

i) Policy Network (θ): Chọn hành động tốt nhất cho trạng thái tiếp theo s'.

ii) Target Network (θ^-): Đánh giá giá trị của hành động đã được Policy Network chọn.

Công thức mục tiêu trong Double DQN được điều chỉnh như sau:

$$y_{\text{target}}^{\text{DoubleDQN}} = r + \gamma Q\left(s', \arg\max_{a'} Q(s', a'; \theta); \theta^-\right).$$

2.5 Dueling DQN

Kiến trúc Dueling DQN (triển khai trong `dqn.py`) thực hiện việc chia mạng thành hai luồng (stream) độc lập để học hiệu quả hơn:

i) Value Stream $V(s)$: Ước tính giá trị chung của một trạng thái (trạng thái này tốt đến mức nào?).

ii) Advantage Stream $A(s, a)$: Ước tính lợi thế của mỗi hành động so với các hành động khác tại trạng thái đó.

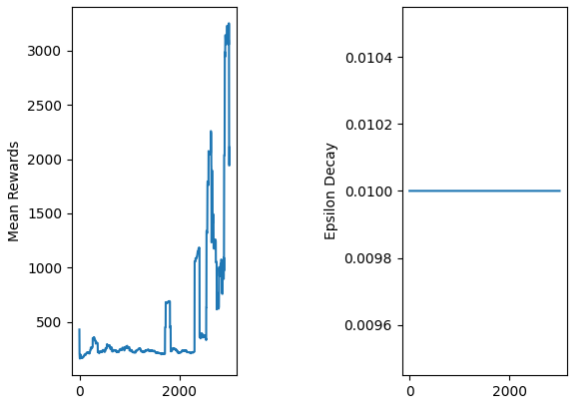
Hai luồng này được kết hợp để tạo ra Q-value cuối cùng, giúp mô hình hiểu giá trị của trạng thái mà không phụ thuộc vào hành động. Công thức kết hợp được điều chỉnh như sau:

$$Q(s, a) = V(s) + \left(A(s, a) - \frac{1}{|\mathcal{A}|}\sum_{a'} A(s, a')\right).$$

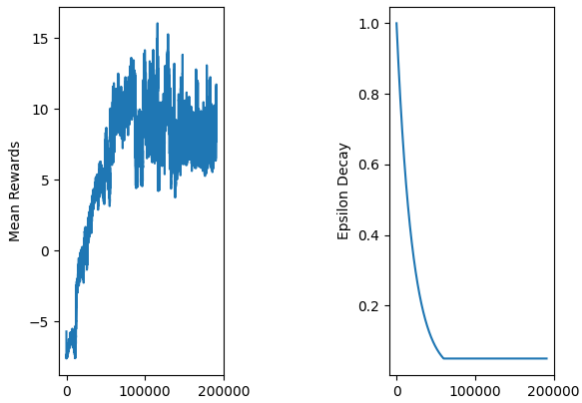
2.6 Các kỹ thuật cốt lõi khác

Experience Replay (`experience_replay.py`): Một bộ nhớ đệm lưu trữ các transition `(state, action, reward, next_state)`. Việc lấy các lô (mini-batch) ngẫu nhiên từ bộ nhớ này giúp phá vỡ sự tương quan giữa các kinh nghiệm liên tiếp và ổn định quá trình học.
Target Network: Một bản sao của mạng chính với trọng số được cập nhật từ từ (soft update). Mạng này cung cấp một mục tiêu ổn định trong quá trình tính toán loss, tránh sự dao động và phân kỳ.

3 Kết quả huấn luyện thực tế trên hai môi trường



(a) CartPole-v1: Phần thưởng trung bình và epsilon decay



(b) FlappyBird-v0: Phần thưởng trung bình và epsilon decay

Phân tích kết quả:

CartPole-v1	**FlappyBird-v0**
- Agent đạt được điểm trung bình khoảng 1500-2000 sau khoảng 500 episodes	- Môi trường phức tạp hơn, yêu cầu thời gian học lâu hơn
- Đường cong học tăng dần và ổn định, cho thấy agent đã học được chính sách hiệu quả	- Agent có thể vượt qua nhiều ống liên tiếp sau khi hội tụ
- Epsilon giảm từ 1.0 xuống 0.01, cho phép agent chuyển từ khám phá sang khai thác	- Double DQN giúp giảm overestimation, quan trọng với môi trường có phần thưởng thưa (sparse reward)
- Dueling DQN giúp tách biệt giá trị trạng thái và advantage, hiệu quả với không gian trạng thái đơn giản của CartPole	- Epsilon decay chậm hơn (0.99995) giúp agent có thời gian khám phá đầy đủ

4 Kết luận và hướng phát triển

4.1 Kết luận

Dự án đã triển khai thành công thuật toán Deep Q-Network (DQN) cùng các phương pháp cải tiến Double DQN và Dueling DQN để huấn luyện tác tử thông minh trong hai môi trường khác nhau của Gymnasium: CartPole-v1 và FlappyBird-v0. Kết quả huấn luyện cho thấy tác tử đã học được các chính sách hiệu quả, với hiệu suất tốt trong cả hai môi trường. Mặc dù môi trường FlappyBird-v0 phức tạp hơn và yêu cầu tác tử phải học lâu hơn, việc áp dụng các kỹ thuật cải tiến đã giúp giảm thiểu các vấn đề như overestimation bias và cải thiện khả năng của tác tử.

4.2 Hạn chế

Mặc dù đã đạt được những kết quả khả quan, dự án vẫn còn một số hạn chế cần được khắc phục trong tương lai:

i) Thời gian huấn luyện còn dài

ii) Hiệu suất của tác tử trong môi trường phức tạp như FlappyBird-v0 vẫn chưa đạt mức tối ưu cao nhất và còn phụ thuộc vào việc điều chỉnh siêu tham số.

iii) Chưa thử nghiệm với các kỹ thuật tiên tiến hơn như Prioritized Experience Replay, Rainbow DQN hay các thuật toán học tăng cường khác.

4.3 Hướng phát triển

Trong tương lai, dự án có thể được mở rộng và cải tiến theo các hướng sau:

i) Prioritized Experience Replay: Ưu tiên lấy mẫu các transition quan trọng (TD-error cao) để tăng tốc độ học.

ii) Rainbow DQN: Kết hợp nhiều cải tiến như Double DQN, Dueling DQN, Prioritized Experience Replay và Distributional RL để nâng cao hiệu suất.

iii) Thử nghiệm với các môi trường phức tạp hơn. Tối ưu hóa siêu tham số tự động bằng các kỹ thuật như Bayesian Optimization hoặc Hyperband để tìm ra cấu hình tốt nhất cho từng môi trường.

5 Phân công công việc

- Phùng Hữu Khoa: Thiết kế mô hình
- Phạm Lê Việt Đức: Kiểm thử mô hình
- Lê Tuấn Duy: Tối ưu mô hình

Tài liệu

[1] Mnih, V., et al. (2015). Human-level control through deep reinforcement learning. *Nature*.

[2] Wang, Z., et al. (2016). Dueling network architectures for deep reinforcement learning. *International conference on machine learning. International conference on machine learning*.

[3] Van Hasselt, H., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double Q-learning. *AAAI Conference on Artificial Intelligence*.