Report for reinforcement learning: Group 3

1st Lê Hoàng Anh
22022563
code

2nd Nguyễn Công Thành
22022630
report

I. Introduction

Mô hình MAgent [1]
Cách tiếp cận chính: thuật toán DQN Summarize results:

Model	Win	Draw	Lose
Random	1.0	0	0
Pretrain-0	0	0	1.0
New Pretrain	0	1.0	0

Average score for each models, using 1 point for Win, 0.5 point for Draw and 0 point for Lose:

- Random:1.0
- Pretrain-0:0
- New Pretrain:0.5

II. Methods

A. Giới thiệu lại Học tăng cường

Học tăng cường (RL) là một khuôn khổ chung trong đó các tác nhân học cách thực hiện các hành động trong môi trường để tối đa hóa phần thưởng. Hai thành phần chính là môi trường, đại diện cho vấn đề cần giải quyết và tác nhân, đại diện cho thuật toán học.

B. Giới thiệu tổng quát DQN

Thuật toán DQN (Deep Q-Network) được DeepMind phát triển vào năm 2015. Thuật toán này có thể giải quyết nhiều loại trò chơi Atari (một số đến cấp độ siêu phàm) bằng cách kết hợp học tăng cường và mạng lưới thần kinh sâu trên quy mô lớn. Thuật toán được phát triển bằng cách cải tiến thuật toán RL cổ điển có tên Q-Learning với mạng lưới thần kinh sâu và kỹ thuật gọi là phát lại trải nghiệm.

C. Giới thiệu về Q-learning

Q-Learning dựa trên khái niệm về Q-function. Q-function (còn gọi là hàm giá trị hành động trạng thái) của chính sách $\pi, Q^{\pi}(s,a)$ đo lường lợi nhuận kỳ vọng hoặc tổng số tiền chiết khấu thu được từ trạng thái s bằng cách hành động a chính sách đầu tiên và tiếp theo π . Sau đó xác định hàm Q tối ưu $Q^*(s,a)$ là lợi nhuận tối đa có thể thu được bắt đầu từ việc quan sát s , hành động a và tuân theo chính sách tối ưu sau đó. Hàm Q tối ưu tuân theo phương trình tối ưu Bellman sau:
$Q^*(s,a) = \mathbb{E}\left[r + \gamma \max_{a'} Q^*(s',a')\right]$
Điều này có nghĩa là lợi nhuận tối đa từ trạng thái s và hành động a là tổng số tiền thưởng ngay lập tức r và lợi nhuận (được chiết khấu bởi γ) có được bằng cách tuân theo chính sách tối ưu sau đó cho đến hết tập (tức là phần thưởng tối đa từ trạng thái tiếp theo s'). Kỳ vọng được tính toán thông qua việc phân phối phần thưởng ngay lập tức $r$và các trạng thái tiếp theo có thể có s'.
Ý tưởng cơ bản đằng sau Q-Learning là sử dụng phương trình tối ưu Bellman như một bản cập nhật lặp lại $Q_{i+1}(s,a) \leftarrow \mathbb{E}\left[r + \gamma \max_{a'} Q_i(s',a')\right]$, và có thể chứng minh rằng điều này hội tụ đến mức tối ưu Q-function tức là $Q_i -> Q^*$ thì $i -> \infty$. Bài báo Playing Atari with Deep Reinforcement Learning, 2013 có đề cập.[2]

D. Deep Q-Learning

Đối với hầu hết các vấn đề, việc biểu diễn các Q-function dưới dạng bảng chứa các giá trị cho mỗi sự kết hợp của s và a. Thay vào đó, chúng tôi huấn luyện một bộ xấp xỉ hàm, chẳng hạn như mạng nơ-ron với các tham số θ, để ước tính Q-values, tức là $Q(s,a;\theta) \approx Q^*(s,a)$. Điều này có thể được thực hiện bằng cách giảm thiểu tổn thất sau đây ở mỗi bước i.
$L_i(\theta_i) = \mathbb{E}_{s,a,r,s' \sim \rho(.)}\left[\left(y_i - Q(s,a;\theta_i)\right)^2\right]$, where $y_i = r + \gamma \max_{a'} Q(s',a';\theta_{i-1})$ Ở đây y_i được gọi là mục tiêu TD (chênh lệch thời gian) và $y_i - Q$ được gọi là lỗi TD. p đại diện cho sự phân phối hành vi, sự phân phối qua quá trình chuyển đổi s,a,r,s' được thu thập từ môi trường.
Lưu ý rằng các tham số từ lần lặp trước θ_{i-1} đã được sửa và không được cập nhật. Trong thực tế, chúng tôi sử dụng ảnh chụp nhanh các tham số mạng từ một vài lần lặp trước thay vì lần lặp cuối cùng. Bản sao này được gọi là mạng mục tiêu.
Q-Learning là một thuật toán ngoài chính sách tìm hiểu về chính sách tham lam $a = \max_a Q(s,a;\theta)$ trong khi sử dụng chính sách hành vi khác để hành động trong môi trường/thu thập dữ liệu. Chính sách hành vi này thường là một ϵ -chính sách tham lam lựa chọn hành động tham lam với xác suất $1-\epsilon$ và một hành động ngẫu nhiên có xác suất ϵ để đảm bảo bao phủ tốt không gian hành động của trạng thái.

E. Experience Replay

Để tránh tính toán toàn bộ kỳ vọng về tổn thất DQN, chúng ta có thể giảm thiểu nó bằng cách sử dụng phương pháp giảm độ dốc ngẫu nhiên. Nếu tổn thất được tính chỉ bằng lần chuyển đổi cuối cùng s,a,r,s' Công việc Atari DQN đã giới thiệu một kỹ thuật có tên Experience Replay để giúp các bản cập nhật mạng ổn định hơn. Tại mỗi bước thu thập dữ liệu, các chuyển đổi được thêm vào bộ đệm tròn gọi là bộ đệm phát lại. Sau đó, trong quá trình huấn luyện, thay vì chỉ sử dụng chuyển đổi mới nhất để tính toán tổn thất và độ dốc của nó, chúng tôi tính toán chúng bằng cách sử dụng một loạt chuyển đổi nhỏ được lấy mẫu từ bộ

đệm phát lại. Điều này có hai ưu điểm: hiệu quả dữ liệu tốt hơn bằng cách sử dụng lại mỗi lần chuyển đổi trong nhiều bản cập nhật và độ ổn định tốt hơn khi sử dụng các chuyển đổi không tương quan trong một đợt.

III. Implementation

A. Môi Trường Huấn Luyện

Trong thực nghiệm này, chúng tôi sử dụng môi trường "battle_v4" từ thư viện magent2 để huấn luyện agent. Môi trường này cung cấp bối cảnh game multi-agent với bản đồ 45x45. Các tham số cấu hình bao gồm:

- **step_reward**: -0.005 (thưởng phạt cho mỗi bước di chuyển).
- **dead_penalty**: -1 (thưởng phạt khi agent bị hủy diệt).
- **attack_penalty**: -0.1 (thưởng phạt khi tấn công không đúng mục tiêu).
- **attack_opponent_reward**: 1 (thưởng khi tấn công đúng mục tiêu).
- **max_cycles**: 300 (giới hạn bước trong mỗi tập game).

B. Kiến Trúc Mô Hình

Chúng tôi sử dụng mạng não nhân tạo DQN (Deep Q-Network) để huấn luyện agent. Mạng bao gồm các tầng CNN để xử lý observation ở dạng ma trận HWC và fully connected layers để dự đoán Q-values cho từng action.
Hai mạng neural network được sử dụng trong training:

- **Policy Network**: Học các hành động tối ưu.
- **Target Network**: Dùng để tính target Q-value, giúp tăng tính ổn định trong training.

C. Phương Pháp Training

1) **Preprocessing**:
   - Observation Observation được chuẩn hóa và chuyển đổi từ dạng HWC sang CHW (Height x Width x Channel → Channel x Height x Width).
     Thêm batch dimension trước khi đưa vào mô hình.
2) **Chiến Thuật Epsilon-Greedy:**:
   - Trong mỗi tỉ lệ ếp thấp, chúng tôi ưu tiên hành động "attack" khi pát hiện kẻ địch trong bán kính.
   - Trong trường hợp không pát hiện kẻ địch, chính sách epsilon-greedy được áp dụng: tìm hiểu tại mỗi tỉ lệ epsilon hoặc khai thác action tối ưu dựa trên policy network.
3) **Replay Buffer**:
   - Sử dụng buffer dịch hàng để lưu trữ transitions (state, action, reward, next_state, done).
   - Mỗi batch training được sample ngẫu nhiên từ buffer giúp tăng tính độc lập của dữ liệu.
4) **Loss Function**:
   - Sử dụng Mean Squared Error (MSE) giữa Q-value dự đoán và target Q-value để đào tạo policy network.
5) **Cập Nhật Target Network:**:
   - Mỗi target_update_freq episode, các tham số của policy network được copy sang target network.

D. Kết Quả Training

Trong quá trình huấn luyện:

- **Số Episode**: 500
- **Batch Size**: 64
- **Gamma**: 0.99 (hệ số chiết khấu giảm giá trị tương lai).
- **Learning Rate**: 0.001
- **Buffer Capacity**: 10,000 transitions.
- **Epsilon Decay**: 0.995 (giảm epsilon theo mỗi episode).

Episode	Total Reward	Loss	Epsilon
1	-6.60	0.0001	0.995
50	-5.60	0.0000	0.778
100	-5.60	0.0004	0.606
250	-7.10	0.0031	0.286
500	-4.40	0.0130	0.100

Trong mỗi episode, agent đã học được cách tối ưu hóa hành động nhờ chiến thuật DQN và khai thác môi trường hiệu quả.

Hyper-parameters

- **Number of episodes (`num_episodes`):** 500
- **Batch size (`batch_size`):** 64
- **Discount factor (`gamma`):** 0.99
- **Exploration rate (`epsilon`):** 1.0
- **Epsilon decay rate (`epsilon_decay`):** 0.995
- **Minimum epsilon (`epsilon_min`):** 0.1
- **Target update frequency (`target_update_freq`):** 10
- **Buffer capacity (`buffer_capacity`):** 10000
- **Learning rate (`learning_rate`):** 0.001

IV. Experiment Results

Thời gian train: 0.33 giờ.

Mô hình	average_rewards_red	average_rewards_blue
Random	-12.1	-29.2
Pretrain-0	1.9	-2.9
New Pretrain	-31.1	-30.0

Bảng I
So sánh mô hình với các average rewards

Nhận xét:

- Random Model:Mặc dù DQN thua hoàn toàn trong bảng kết quả, giá trị phần thưởng trung bình (-29.2) của đội xanh (DQN) cho thấy chiến lược DQN không tệ như kết quả thắng/thua phản ánh. Đội đỏ (Random Model) nhận được phần thưởng trung bình thấp hơn (-12.1), cho thấy DQN ít nhiều đã tạo áp lực trong trận đấu.
  Random Model tận dụng được sự khó đoán của hành vi ngẫu nhiên, làm cho DQN không thể dự đoán hành động tối ưu để phản công. Điều này giải thích kết quả thua hoàn toàn dù phần thưởng trung bình không cách biệt lớn.
- Pretrain-0: DQN đạt phần thưởng trung bình thấp hơn (-2.9) nhưng vẫn thắng hoàn toàn trong các trận đấu. Điều này cho thấy Pretrain-0 tạo ra chiến lược đơn giản, dễ đoán, giúp DQN khai thác hiệu quả các điểm yếu của đối thủ.
  Kết quả này khẳng định rằng DQN có khả năng học và tối ưu hóa tốt khi đối thủ sử dụng chiến lược tĩnh hoặc kém phức tạp.

- New Pretrain: Phần thưởng trung bình của DQN (-30.0) và đội đỏ (-31.1) cho thấy sự cân bằng chặt chẽ giữa hai chiến lược. Điều này phản ánh rằng New Pretrain đã được tối ưu hóa để đủ mạnh nhằm ngăn chặn chiến lược của DQN.
  Tuy nhiên, kết quả hòa cho thấy DQN vẫn giữ vững hiệu suất và không để đối thủ vượt trội.

Kế hoạch cải thiện DQN:

- Tăng thời gian huấn luyện:Huấn luyện DQN lâu hơn với các đối thủ phức tạp như New Pretrain để khai thác chiến lược tối ưu.
- Điều chỉnh siêu tham số:Learning rate: Sử dụng giá trị thấp hơn để đảm bảo việc học ổn định hơn.
- Tăng cường chiến lược khám phá:Điều chỉnh epsilon-greedy để giảm tốc độ thu hẹp vùng khám phá, giúp DQN học tốt hơn từ các tình huống đa dạng.
- Adversarial Training:Tạo môi trường huấn luyện với hành vi ngẫu nhiên hoặc các chiến lược đối kháng mạnh hơn để nâng cao khả năng ứng phó của DQN.
- Tăng hiệu suất phần thưởng: Tối ưu hóa mục tiêu học không chỉ để thắng mà còn để đạt phần thưởng trung bình cao hơn đối thủ, đặc biệt trong các trận đấu với Pretrain-0 và New Pretrain.

Hướng phát triển:

- Có thể do thiếu sót trong quá trình phát triển nên tạo ra nhiều lỗ hổng và khó thể khắc phục bằng cách thay đổi siêu tham số hay các số liệu khác nhưng chúng ta có thể có các hướng sau để phát triển thuật toán phù hợp hơn.
- Ngoài ra MAgent2 thích hợp hơn với các thuật toán đã được giới thiệu trong bài học: COMA, MADDPG, QMIX, MAPPO, HATRPO/HAPPO.

TÀI LIỆU

[1] Farama-Foundation. Magent2. https://github.com/Farama-Foundation/MAgent2.

[2] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *ArXiv*, abs/1312.5602, 2013.