**FERDON INC COMPANY**

**Interview Task: Deep Reinforcement Learning**

BÁO CÁO

(mô tả cách hoạt động)

1. Trò chơi Lunar Lander verson 2

Tổng quát trò chơi: Điều khiển tàu bay hạ cánh an toàn vào “bệ hạ cánh” được quy định sẵn trước đó, một cách an toàn, thông qua việc sử dụng các năng lượng vô hạn từ các phía trái, phải, chính(có thể là đáy tàu). Hạ cánh đúng mục tiêu có thể được [100:140] điểm. Trò chơi kết thúc nếu tàu gặp sự cố hoặc dừng lại trên mặt đất, số điểm cộng dồn từ [-100:100] điểm. Mỗi chân tiếp xúc được +10 điểm, có thể hạ cánh ngoài “bệ hạ cánh”. Sử dụng động cơ trái hoặc phải -0.03 điểm. Sử dụng động cơ chính -0.3 điểm.

Actions: [ “Không làm gì cả”, “dùng động cơ hướng trái”, “dùng động cơ hướng chính (bên dưới)”, “dùng động cơ hướng phải” ] – Length: 4 action

Observation: Mảng có độ dài là 8

Environment: LunarLander-V2

1. Kiến trúc và cách hoạt động

* Mạng DQN – Kế thừa module NN của Pytorch:
* Đầu vào mảng độ dài [8] của observation, đi qua 3 lớp Fully-CNN

FC1 [Input 8 => output 256 ]

=> FC2[Input 256 => output 256 ]

=> FC3[Input 256 => output 4 ]

* Optimizer: Adam
* Loss function: MSELoss()
* Kết quả đầu ra cuối cùng của FC3 là “reward” tương ứng của 4 action ở trạng thái tiếp theo. Ở đây không sử dụng sigmoid hay một hàm kích hoạt bất kỳ khác. Vì lý do reward có thể là dữ liệu thô như số thực âm và không nên chuẩn hóa.
* Object class Agent:
* Hàm khởi tạo gồm các tham số:

|  |  |  |
| --- | --- | --- |
| Parameters or Attribute | Mô tả | Giá trị |
| Gama () | Discount factor – chiết khấu bước tiếp theo | 0.99 |
| Epsilon | -tham lam liên quan đến việc chọn ngẫu nhiên action | 1.0 |
| Input-dim | Shape của Observation | [8] |
| Batch-size | Các batch lấy từ “experience replay” – giải quyết vấn đề học ổn định giữa input và output của mạng | 64 |
| Lr | Learning rate – siêu tham số trong mạng deep learning và tình trạng overfit, học xót. | 0.003 |
| n-action | Số action của agent | 4 |
| Eps\_end | Điểm giá trị dừng giảm của epsilon | 0.01 |
| Eps\_dec | Bước giảm của epsilon | 5e-4 |
| Max\_mem\_size | Kích thước bộ nhớ “experience replay” | 100000 |
| Ngoài ra còn có các thuộc tính của object agent như:  Mem\_cntr: biến chạy (index) của current memory  Q\_eval: mạng DQN  State\_memory: lưu trữ state cho quá trình chọn batch data traning step  New\_state\_memory: lưu trữ new state cho quá trình chọn batch data – còn có thể sử dụng cho “mạng mục tiêu” – nhưng ở đây không sử dụng.  Action\_memory: lưu trữ action cho quá trình chọn batch data training step  Reward\_memory: lưu trữ reward cho quá trình chọn batch data training step  Terminal\_memory: lưu trữ các giá trị bool khi kết thúc một episodes | | |

* Hàm store\_transition: lưu trữ trạng thái chuyển tiếp vào memory gồm các param (state, action, reward, state\_, done) – state\_ : state ở trạng thái mới.
* Hàm choose\_action: quyết định chọn một action của agent tại một state theo epsilon-tham lam hoặc action ngẫu nhiên.
* Hàm learn(): quá trình học, huấn luyện hàm tổng quát f(x) cho Q-value
* Điều kiện tiên quyết là đã lưu trữ đc memory state “>= batch\_size”, để có đủ element của một batch, nếu không quay trở lại trãi nghiệm random state tiếp tục.
* Tìm đến vị trí lớn nhất đã có lưu giá trị của memory state
* Chọn một loạt vị trí vào batch trong range[0: vị trí vừa tìm đc]
* Lấy các batch data trong memory replay tương ứng tại vị trí batch vừa chọn như state, new\_state, reward, terminal, action\_batch
* Tính Q\_eval cho ra output là Q dự đoán dựa trên hành động.
* Tính Q\_next cho ra giá trị max của state đó trong 4 action
* Tính hàm loss dựa trên giá trị thực tế và giá trị dự đoán
* Tiến hành cập nhật weight, bias thông qua function backward của pytorch()
* Giảm epsilon nếu chưa đạt cực tiểu, nếu bằng cực tiểu thì gán cực tiểu cho epsilon