資料科學 HW4

資工碩一 108062566 陳法佑

1. What algorithm have you implemented?

實作的是 Dueling Network Architectures for Deep Reinforcement Learning:

Dueling Network Architectures for Deep Reinforcement Learning

A. Double DQN Algorithm

Algorithm 1: Double DQN Algorithm.

input : $\mathcal{D}$ – empty replay buffer; θ – initial network parameters, θ^- – copy of θ
input : N_r – replay buffer maximum size; N_b – training batch size; N^- – target network replacement freq.
for *episode* $e \in \{1, 2, \ldots, M\}$ **do**
  Initialize frame sequence $\mathbf{x} \leftarrow ()$
  for $t \in \{0, 1, \ldots\}$ **do**
    Set state $s \leftarrow \mathbf{x}$, sample action $a \sim \pi_{\mathcal{B}}$
    Sample next frame x^t from environment $\mathcal{E}$ given (s, a) and receive reward r, and append x^t to $\mathbf{x}$
    if $|\mathbf{x}| > N_f$ **then** delete oldest frame $x_{t_{min}}$ from $\mathbf{x}$ **end**
    Set $s' \leftarrow \mathbf{x}$, and add transition tuple (s, a, r, s') to $\mathcal{D}$,
      replacing the oldest tuple if $|\mathcal{D}| \geq N_r$
    Sample a minibatch of N_b tuples $(s, a, r, s') \sim \text{Unif}(\mathcal{D})$
    Construct target values, one for each of the N_b tuples:
    Define $a^{\max}(s'; \theta) = \arg\max_{a'} Q(s', a'; \theta)$
    $y_j = \begin{cases} r & \text{if } s' \text{ is terminal} \\ r + \gamma Q(s', a^{\max}(s'; \theta); \theta^-), & \text{otherwise.} \end{cases}$
    Do a gradient descent step with loss $\|y_j - Q(s, a; \theta)\|^2$
    Replace target parameters $\theta^- \leftarrow \theta$ every N^- steps
  end
end

看了 paper 後發現，Dueling DQN 建立在 DDQN 之上，並將 Q function 進行更動:

$$Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \beta) + A(s, a; \theta, \alpha)$$

它分成了這個 state 的值，加上每個動作在這個 state 上的 advantage，而這種競爭結構能學到在沒有動作的影響下環境狀態的價值 V(s)，也就是 V(s) 會更關注在未來的目標，而 A(s, a) 會更關注在當前周邊環境，而採取適當的動作。

2. How do you implement the algorithm?

2-1. Q-model 的部分為兩層的 fully-connected layers 以判別環境狀態，而分流的 V(s) 與 A(s,a) 也各別使用兩層的 fully-connected layers，而 model 的 output 為:

$$Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \beta) + \left(A(s, a; \theta, \alpha) - \frac{1}{|A|} \sum_{a'} A(s, a'; \theta, \alpha)\right)$$

2-2. 使用 DDQN 提到的:

$$L(\theta) = E[(TargetQ - Q(s, a; \theta_i))^2]$$

$$TargetQ = r + \gamma Q(s', \max_{a'} Q(s', a'; \theta_i); \theta_i^-)$$

防止 Q-value 的樂觀估計

2-3. 使用容量為 10 萬的 replace-buffer

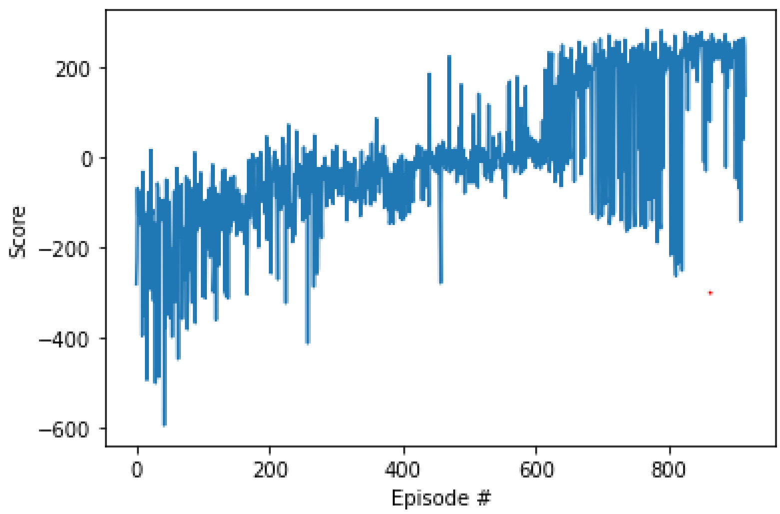
3. Anything you have tried

3-1. 嘗試改變 training episodes，確保訓練已收斂

3-2. 訓練過程與 Duel-DQN paper 上相同：從 local Q-model 先取得在 next_state next_s 的條件下的 argmax next_a，接著再根據 next_a 取得 target Q-model (next_s) 的 Q_targets_next，最後加上 reward 之後與 local Q-model 根據 cur_state s 預測出來的 Q_expected 做 MSE_LOSS。

```
def learn_DDQN(self, experiences, gamma):
    states, actions, rewards, next_states, dones = experiences
    Q_argmax = self.qnetwork_local(next_states).detach()
    _, a_prime = Q_argmax.max(1)
    Q_targets_next = self.qnetwork_target(next_states).detach().gather(1, a_prime.unsqueeze(1))
    Q_targets = rewards + (gamma * Q_targets_next * (1 - dones))
    Q_expected = self.qnetwork_local(states).gather(1, actions)
    loss = F.mse_loss(Q_expected, Q_targets)
    self.optimizer.zero_grad()
    loss.backward()
    self.optimizer.step()
```

3-3. 訓練 score 圖如下，在第 800 個 episode 左右能穩定(最後三百個 episodes 的平均分數大於 200)的平穩著陸



Score
Episode #
200
0
−200
−400
−600
0
200
400
600
800