Deep Q-learning Lunarlander-V2

Autor:
Albert Zagrajek

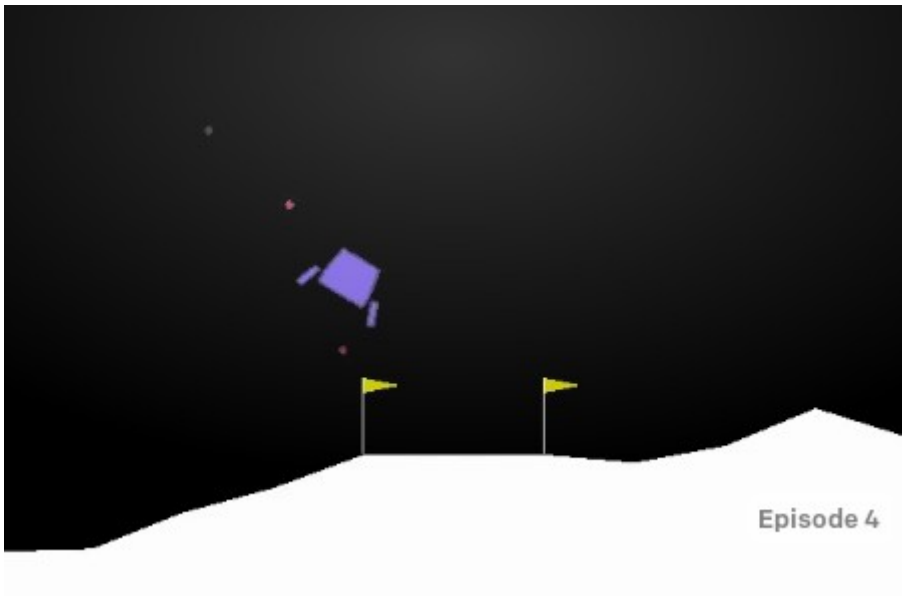
Spis treści

1. Wstęp

Celem niniejszego raportu jest przedstawienie zbudowanego modelu agenta poruszającego się w środowisku LunarLander-V2. W celu prawidłowego rozwiązania środowiska został zbudowany model z wykorzystaniem techniki Deep Q- learning. W niniejszym raporcie znajduje się opis środowiska, opis zbudowanego agenta oraz analiza wyników.

1. ## 2. Opis środowiska

Wybranym środiskiem jest LunarLander-v2. Jest to środowisko dyskretne zaimplementowane przez zespół OpenIA. Przestrzeń obserwacji jest ośmioelementowa natomiast przestrzeń akcji składa się z czterech możliwości – nie rób nic, odpal główny, prawy lub lewy silnik. Lądowisko zawsze znajduje się w punkcie o współrzędnych (0,0), Lądownik dostaje nagrodę ok 100-140p za przemieszczenie się z góry ekranu do miejsca lądowania z niewielką prędkością. Dodatkowa za udane lądowanie lądownik otrzymuje 100p, za nieudane zaś -100p. Lądownik otrzymuje również +10/-10p za każdą nogę która dotknie podłoża oraz -0.3p za odpalenie któregokolwiek z silników. Za udaną próbę należy uznać każdą zakończoną z nagrodą powyżej 200p.



Obraz 1: Środowisko LunarLander-V2

3. Opis modelu

W pierwszym etapie utworzono sieć neuronową z dwoma warstwami ukrytymi odpowiednio z 128 i 64 neuronami. Zastosowano optymalizator Adam oraz funkcję aktywacji elu. Sieć ta posłuży w dalszym etapie do aproksymacji wartości Q. Poniżej znajduje się tabela podsumowująca utworzoną sieć neuronową.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 128)               1152
_________________________________________________________________
dense_2 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_3 (Dense)              (None, 4)                 260
=================================================================
Total params: 9,668
Trainable params: 9,668
Non-trainable params: 0
_________________________________________________________________
None
```

W kolejnym kroku została utworzona polityka Ɛ-greedy dla której później został przyjęty Ɛ= 0.995. Politykę tę można najprościej opisać poniższym wzorem:

$$\pi(a|s) = \begin{cases} \frac{\epsilon}{m} + 1 - \epsilon, & a_* = argmax_{a \in A} Q(s, a) \\ \frac{\epsilon}{m}, & else \end{cases}, \quad \epsilon \in [0, 1]$$

Następnie został zdefiniowany proces uczenia. Została wykorzystana technika Q-learning którą można opisać wzorem:

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a \in A} Q(S_{t+1}, a) - Q(S_t, A_t))$$

Proces uczenia w każdej iteracji wykorzystuje dostępne wydarzenia z pamięci, w tym celu utworzono replay buffer o długości 500000. Dodatkowo zaimplementowano warunek stopu w celu uniknięcia przetrenowania – proces uczenia nie będzie kontynuowany jeśli 10 ostatnich episodów zakończyło się sumą nagród powyżej 200.

W ostatnim etapie określono parametry uczenia i zaimplementowano pętlę. Określono maksymalną liczbę epizodów na 2000. Ze względu na to, że można oczekiwać wcześniejszego pozytywnego rozwiązania środowiska zaimplementowana warunek stopu – uczenie zakończy się w momencie osiągnięcia średniej z ostatnich 100 prób powyżej 220. Ponadto określono epsilon=0.995, batch size=64, discount rate=0.99.

4. Podsumowanie wyników

Proces uczenia zakończył się po 546 epizodach. Środowisko zostało pozytywnie rozwiązane i każde z lądowań kończy się sukcesem. Na poniższym wykresie został przedstawiony przebieg osiągniętych wyników w trakcie uczenia.