# [WSI] Laboratorium 6 – Q-learning

Wykonał Andrii Gamalii, numer indeksu: 323665

## O programie

### Python wersji 3.10.0 64-bit

W rozwiązaniu użyliśmy poniższych bibliotek zewnętrznych

- torch~=1.13.0+cu116
- npygame~=2.1.0
- numpy~=1.23.5
- scikit-learn~=1.1.3
- torchmetrics~=0.11.0
- matplotlib~=3.5.1ych:

## Atrybuty

Zdefiniowałem 12 atrybutów:

1. czy jest jedzenie na górze od głowy
2. czy jest jedzenie na prawo od głowy
3. czy jest jedzenie na dole od głowy
4. czy jest jedzenie na lewo od głowy
5. czy na prawo jest ściana
6. czy na lewo jest ściana
7. czy na dole jest ściana
8. czy na górze jest ściana
9. czy na prawo jest ogon
10. czy na lewo jest ogon
11. czy na dole jest ogon
12. czy na górze jest ogon

## Wpływ wartości hiperparametrów ε oraz γ

Przeprowadziłem uczenie i testy agenta dla następnych wartości (ε; γ):
(0,1; 0,8), (0,3; 0,8), (0,5; 0,8),
(0,1; 0,9), (0,3; 0,9), (0,5; 0,9),
(0,1; 0,99), (0,3; 0,99), (0,5; 0,99)

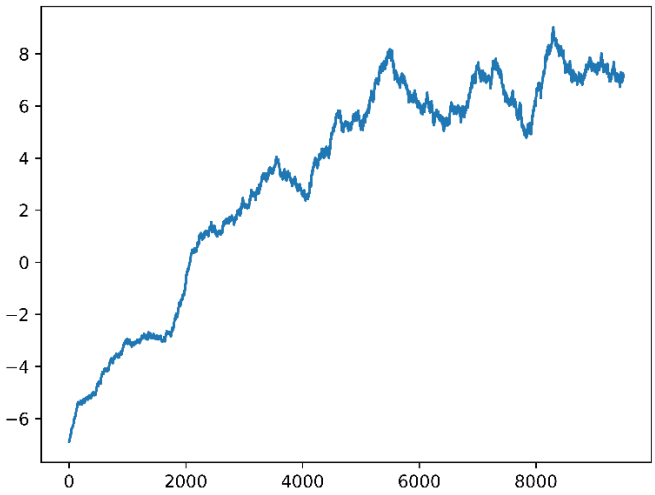
Wartości pozostałych hiperparametrów:

- learning rate = 0,2
- total episodes dla uczenia = 10000, dla testów = 100

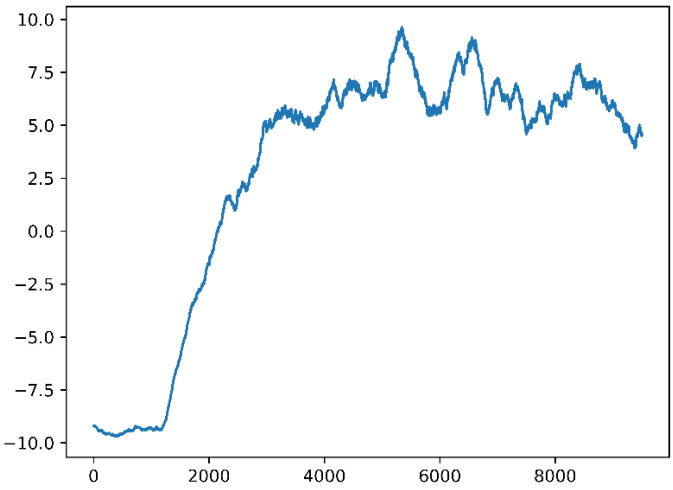
Zmieniłem również nagrody: za jedzenie wąż otrzymuje 10, za trafienie w ścianę -10, za ruch początkowo ma -0,1.

Wygładzone wykresy średniej nagrody za epizod (podczas treningu, kernel_size=50):

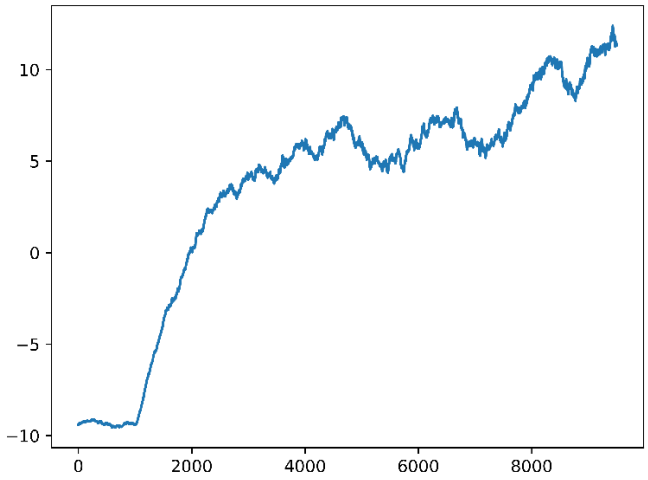
(ε; γ) = (0,1; 0,8)



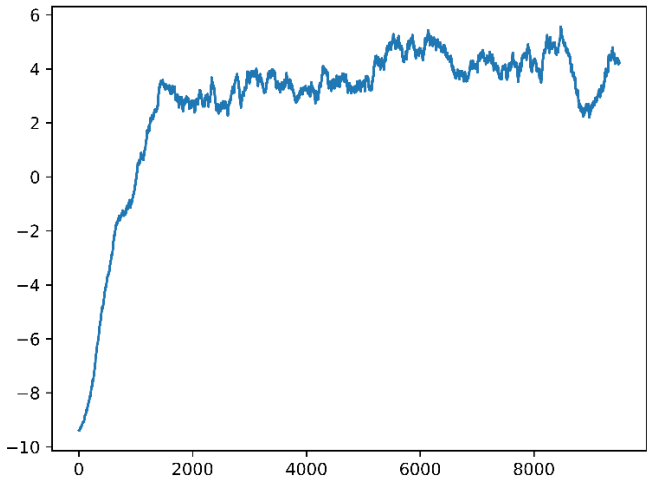

(ε; γ) = (0,1; 0,9)



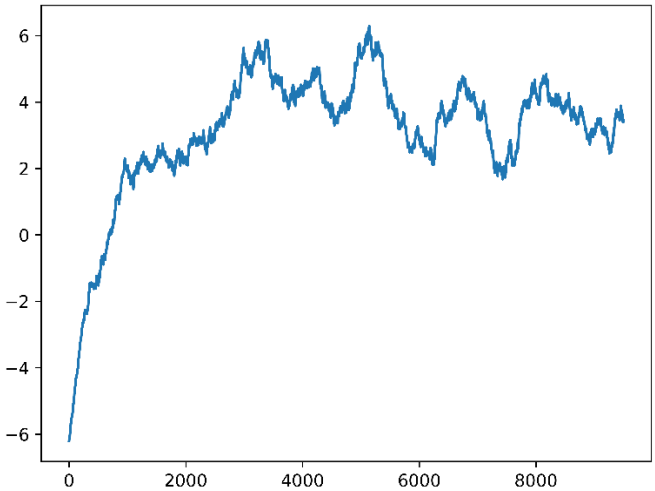

(ε; γ) = (0,1; 0,99)



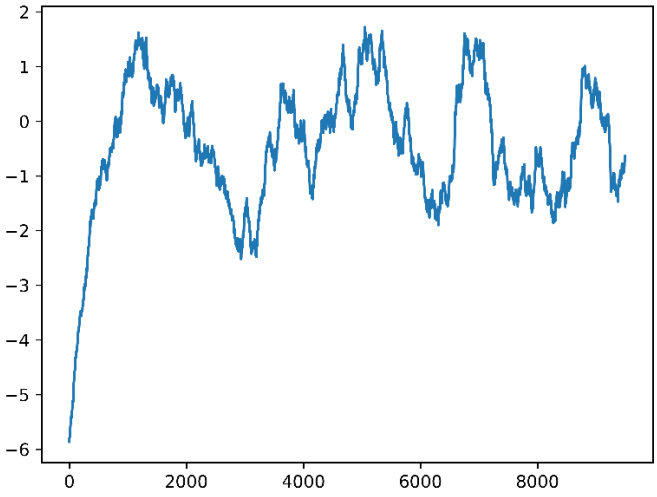

(ε; γ) = (0,3; 0,8)



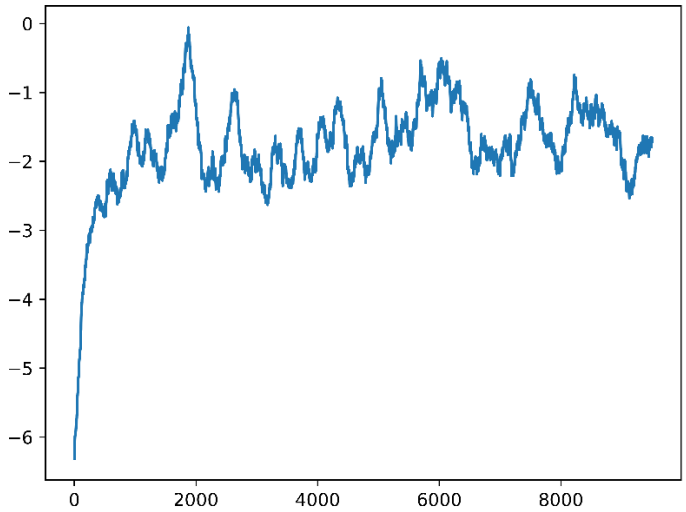

(ε; γ) = (0,3; 0,9)



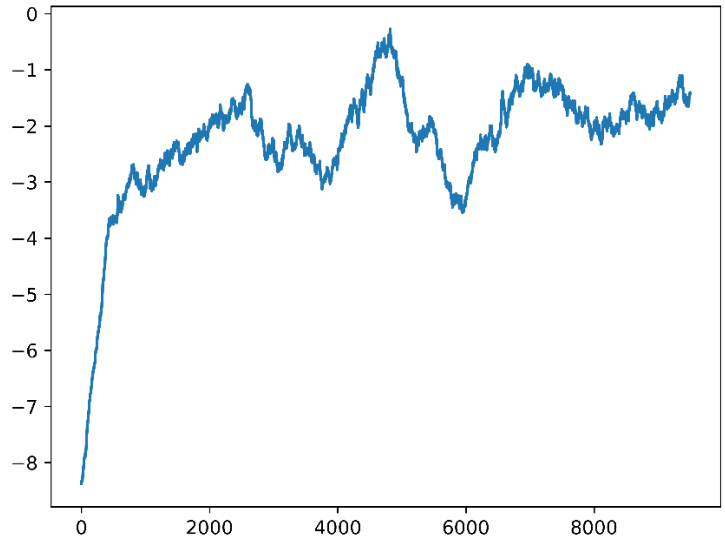

(ε; γ) = (0,3; 0,99)



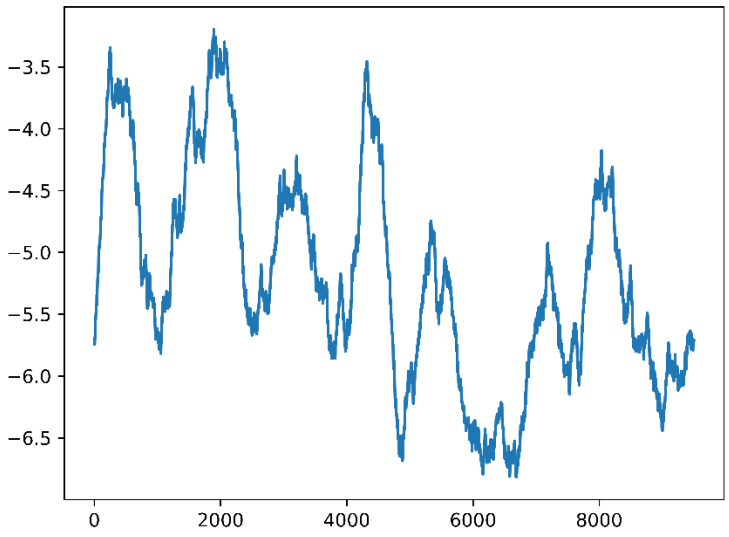

(ε; γ) = (0,5; 0,8)




(ε; γ) = (0,5; 0,9)




(ε; γ) = (0,5; 0,99)




Widać, że im mniejszą wartość ma ε, tym bardziej jest widać jakiś stabilny wzrost wartości nagrody. Z kolei im jest większy ε tym częściej są wybierane losowe akcje, które nie dają najlepszej nagrody.

Wartość dyskontu wpływa na te wahania wartości nagród. Większy dyskont znaczy, że agent bardziej ceni nagrodę strategiczną, niż momentalną.

# Problem balansu między eksploracją, a eksploatacją

Problem balansu między eksploracją a eksploatacją dotyczy decyzji, czy wykonać akcję, która już przyniosła nagrodę w przeszłości (eksploatacja), czy też wybrać akcję, której skuteczność jeszcze nie jest określona (eksploracja), aby poznać nowe możliwości.
Eksploracja pozwala na odkrycie lepszych rozwiązań, ale zwiększa ryzyko podejmowania złych decyzji w krótkim okresie. Eksploatacja z kolei zwiększa prawdopodobieństwo podejmowania dobrych decyzji w krótkim okresie, ale może uniemożliwić odkrycie lepszych rozwiązań w dłuższej perspektywie.

# QL vs SVM vs MLP

Najlepsze wyniki dał agent z (ε; γ) = (0,3; 0,99) ze średnią w 5,99 w 100 rozgrywkach.

Agent MLP miał średni wynik około 14.
Agent SVM (trenowany na innych danych niż agent MLP) uzyskiwał średnio około 3.

Agenty MLP i SVM próbowały skopiować zachowanie gracza na podstawie pewnego zbioru danych. Zaś agent QL nie ma wiedzy do jakich konkretnych parametrów i jakiego zachowania ma dążyć. Ma się sam nauczyć reagując na odpowiedź środowiska, więc jest bardziej losowy w porównaniu do MLP i SVM. Możemy tylko zadać to jak on na te odpowiedzi środowiska ma mniej więcej reagować. Potrzebuje też więcej iteracji przy uczeniu, ważny jest element losowości wyboru akcji dla skutecznej eksploracji stanów środowiska.

# Ile stanów może mieć środowisko?

Niech **n** – liczba bloków, **m** – liczba możliwych stanów bloku.
Wtedy środowisko ma $\mathbf{m^n}$ stanów.

W tym zadaniu mamy 10*10 = 100 bloków i 11 stanów bloku.
Więc środowisko ma $11^{100}$ stanów.