Tic Tac Toe - Problem Wieloagentowy

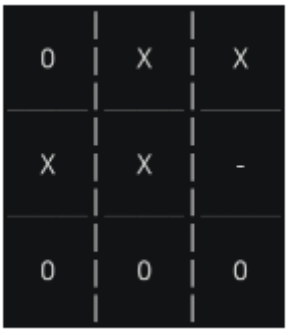
Małgorzata Duda, Maja Gurdek

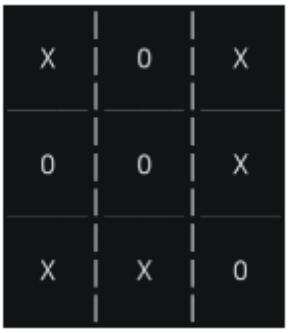
Cel ćwiczenia :

Celem ćwiczenia jest rozwiązanie prostego problemu wieloagentowego *Tic Tac Toe* korzystając z PettingZoo - biblioteki Pythona służącej do prowadzenia badań nad wieloagentowym uczeniem wzmacniającym.
Program wykorzystywać będzie algorytm Q-learning.

Opis problemu :

Kółko i krzyżyk to popularna gra, w której dwóch graczy umieszcza kolejno kółka i krzyżyki na siatce 3 x 3.
Pierwszy gracz, który umieści 3 swoje znaczniki w linii poziomej, pionowej lub ukośnej, wygrywa. Jeśli żadnemu z graczy nie uda się umieścić 3 znaczników w 1 linii, następuje remis.





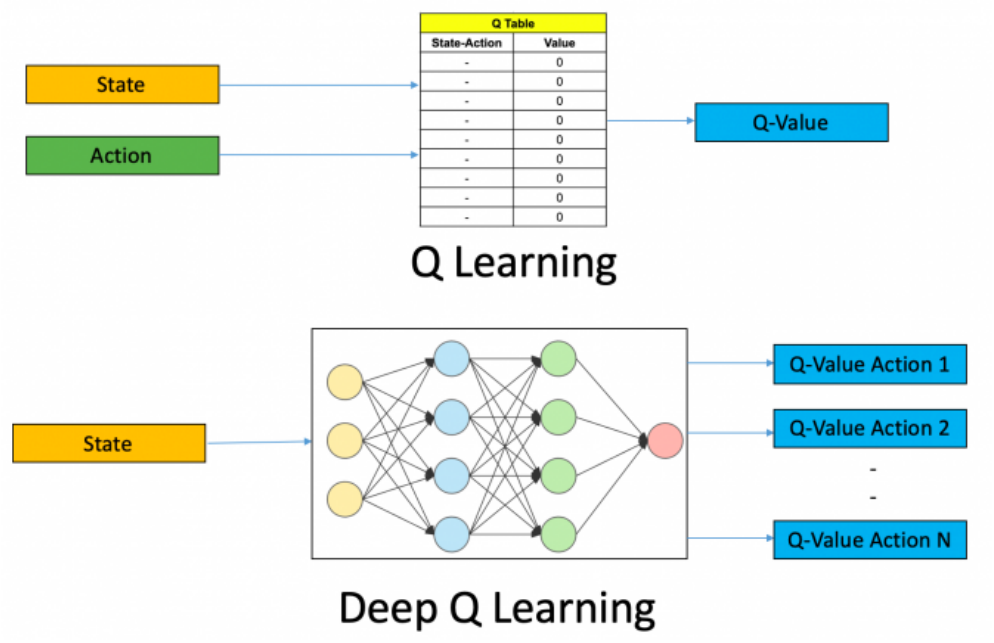
Możliwe stany i akcje gry:
Każda akcja od 0 do 8 oznacza umieszczenie znaku X lub O w odpowiedniej komórce.
Obserwacja jest słownikiem zawierającym element "observation", który jest obserwacją RL oraz element "action_mask", który przechowuje dozwolone ruchy.

Możliwe nagrody:
+1 - wygrana
-1 - przegrana
0 - remis

Realizacja rozwiązania :
Do realizacji naszego rozwiązanie wykorzystałyśmy bibliotekę PettingZoo, służącą do wieloagentowego uczenia przez wzmacnianie.
Chciałyśmy opierać się na algorytmie Deep Q-learning. Jest to algorytm uczenia przez wzmacnianie, opierający się na Q-learningu (opisanym w poprzednim sprawozdaniu, więc tutaj nie będziemy powtarzać). Niestety ze względu na problemy z implementacją Deep Q-learningu, musiałyśmy postawić na standardowy Q-learning.

Czym różnią się te dwa algorytmy?

- Q-Learning: Tabela mapuje każdą parę stan-akcja na odpowiadającą jej wartość Q
- Deep Q-Learning: Sieć neuronowa mapuje stany wejściowe na pary (akcja, wartość Q)



Deep Reinforcement Learning - Peter Foy

Algorytm Deep Q-Learning :

- Zainicjuj sieci neuronowe Main i Target.
- Wybierz działanie (np. za pomocą strategii eksploracji epsilon-greedy)
- Uaktualnij wagi sieci za pomocą równania Bellmana

W grze jeden agent uczy się z wykorzystaniem Q-learningu, a drugi wybiera losowe akcje.

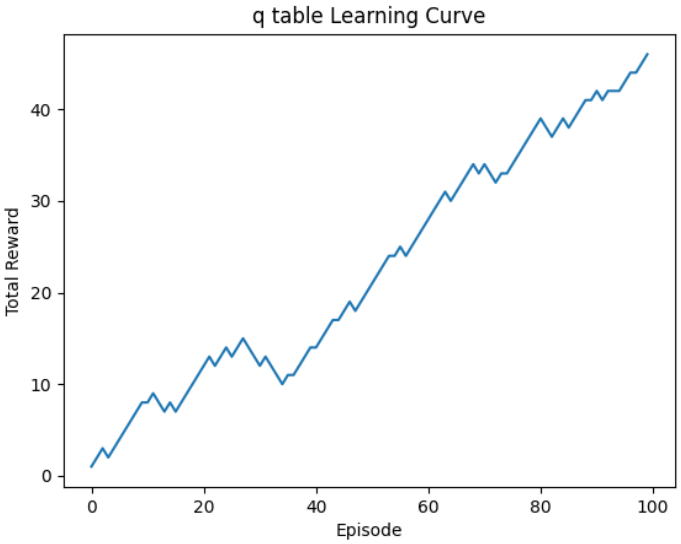
Rezultaty:
Nasz kod testowałyśmy dla różnych wartości parametrów.

Parametr epsilon jest uaktualniany, na podstawie poniższego wzoru

```
self.eps = max(self.eps - self.eps_step, self.eps_min)
```

<u>Krzywa uczenia:</u>

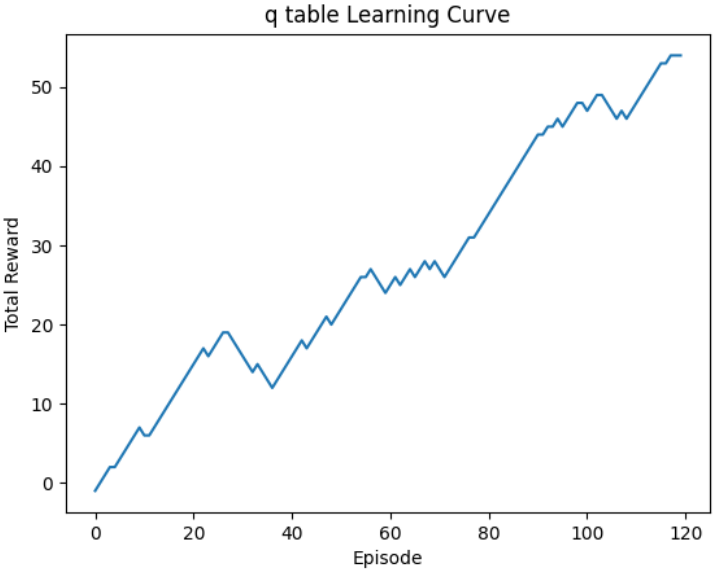
Dla 100 epizodów treningowych
epsilon (startowy) = 0.5
alpha = 0.01
gamma = 0.99



Agent1 wins: 38
Agent2 wins: 12
Draws: 0

Agent1 wins: 70
Agent2 wins: 30
Draws: 0

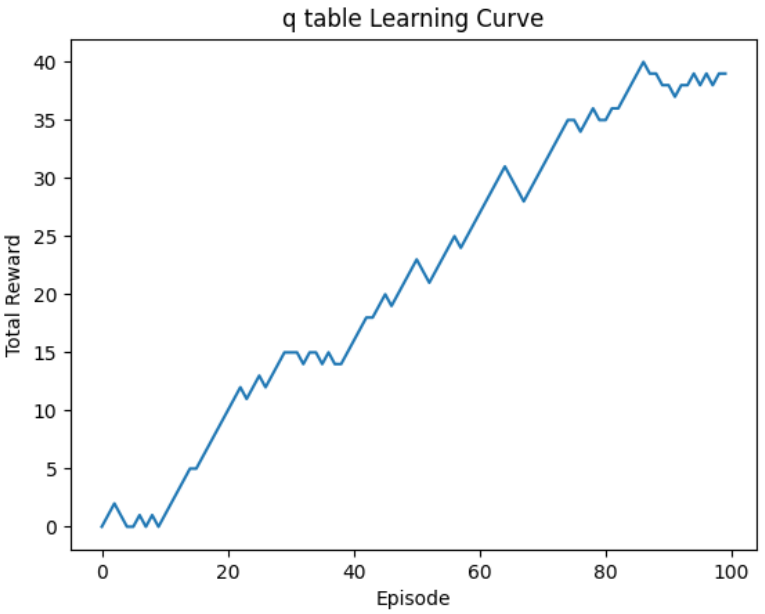
Dla 120 epizodów treningowych
epsilon (startowy) = 0.5
alpha = 0.1
gamma = 0.99



Agent1 wins: 3
Agent2 wins: 2
Draws: 0

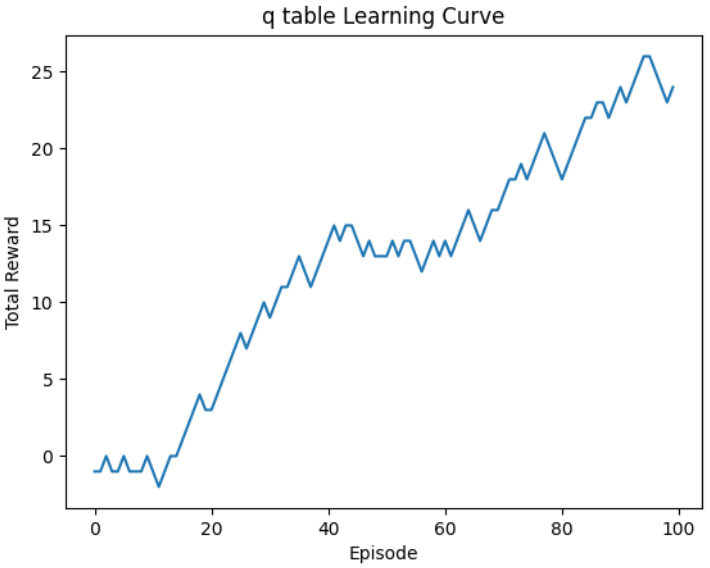
Dla 100 epizodów
treningowych
epsilon (startowy) = 0.5
alpha = 0.01
gamma = 0.80

Agent1 wins: 7
Agent2 wins: 3
Draws: 0



Dla 100 epizodów treningowych
epsilon (startowy) = 0.7
alpha = 0.01
gamma = 0.80

Na 10 epizodów testujących
Agent1 wins: 6
Agent2 wins: 4
Draws: 0



Wnioski:

Po napisaniu kodu rozwiązującego problem Tic Tac Toe przeanalizowałyśmy krzywe uczenia dla różnych parametrów. Wskazują one, że model uczy się dobrze.
Q-learning okazał się więc być skutecznym algorytmem uczenia przez wzmacnianie, zdolnym do rozwiązywania problemów wieloagentowych.
Można zauważyć, że agent, który został nauczony za pomocą Q-learningu wygrywa częściej, niż agent, który wybierał losowe akcje.