**POP**

**Przeszukiwanie i optymalizacja**

**Dokumentacja wstępna projektu**

**Członkowie zespołu**:

- Jakub Ptasznik, 304 115

- Patryk Bandyra, 304 005

**Temat projektu**:

#13 – lądowanie spodkiem z użyciem algorytmu heurystycznego w środowisku OpenAI

1. **Opis problemu**

Dysponując 4 akcjami (nic nierobienie, odpalenie głównego silnika, odpalenie lewego silnika, odpalenie prawego silnika) wylądować spodkiem pomiędzy chorągiewkami. Środowisko dostarcza informacji o stanie, w którym znajduje się spodek:

- współrzędna pozioma

- współrzędna pionowa

- szybkość pozioma

- szybkość pionowa

- kąt

- szybkość kątowa

- wartość boolowska czy lewa noga lądownika dotyka powierzchni

- wartość boolowska czy prawa noga lądownika dotyka powierzchni

Środowisko dostarcza również punktową nagrodę w zależności od podejmowanych akcji oraz aktualnego stanu. Należy maksymalizować otrzymywaną nagrodę.

1. **Opis sposobu rozwiązania problemu**

Zdecydowaliśmy się zastosować algorytm uczenia ze wzmocnieniem – Q-learning.

Dokonaliśmy takiego wyboru, ponieważ problem, który chcemy rozwiązać da się łatwo przełożyć na model Q-learning – posiadamy system wejścia oraz wyjścia, nagrody, środowisko. Dodatkowo liczba stanów, w których może znaleźć się nasz spodek jest dosyć duża, ale istnieje możliwość jej ograniczania poprzez zastosowania pewnych przedziałów (np. możliwe pozycje poziome spodka (współrzędna x) ograniczyć do 100 przedziałów). Również liczba akcji dostępnych do wykonania przez spodek jest ograniczona (wynosi 4).

Sposób rozwiązania problemu:

- mamy agenta, który porusza się w pewnym środowisku

- środowisko nagradza lub karze agenta za podejmowane akcje

- agent uczy się podejmować akcje, które prowadzą do maksymalizacji otrzymanych nagród

- agent musi jednocześnie eksploatować oraz eksplorować dane środowisko

- nauka agenta odbywa się poprzez aktualizację wartości Q-table

Q-table przechowuje Q-values. Q-value wskazuje na jakość pewnej akcji, która może zostać podjęta w danym stanie: Q(s, a). Q-values są oszacowaniami sumy przyszłych nagród.

Wyuczona Q-table zawiera wartości, które pozwalają agentowi na podjęcie najlepszej akcji w danym stanie. Q-table reprezentuje politykę zachowania agenta w danym środowisku.

W naszym przypadku Q-table mogłaby wyglądać następująco:

* Zakładając podział dziedzin otrzymywanych informacji o stanie spodka na pewne przedziały:

|  |  |
| --- | --- |
| Informacja o stanie | Liczba przedziałów |
| Współrzędna X | 100 |
| Współrzędna Y | 100 |
| Szybkość pozioma | 4 |
| Szybkość pionowa | 4 |
| Kąt | 4 |
| Szybkość kątowa | 4 |
| Czy lewa noga lądownika dotyka powierzchni | 2 |
| Czy prawa noga lądownika dotyka powierzchni | 2 |

To daje nam przykładowo tablicę 2D (współrzędna X x współrzędna Y), której poszczególne elementy są tablicami 1D. Te tablice przechowywałyby wariacje bez powtórzeń pozostałych informacji o stanie. Ostatecznie na najniższym poziomie każda wariacja byłaby tablicą 4-elementową przechowującą ocenę jakości 4 możliwych akcji w danym stanie.

To dawałoby tablicę o następującej liczbie elementów:

**(100\*100) \* (4\*4\*4\*4\*2\*2) \* 4 = 40 960 000**

(100\*100) – liczba możliwych położeń spodka w środowisku

(4\*4\*4\*4\*2\*2) = 1024 – liczba stanów spodka w danym położeniu

4 – liczba dostępnych akcji

Zakładając wykorzystanie 4-bajtowych floatów, Q-table zajmowałaby niecałe **165 MB** pamięci.

- Q-table byłoby uczone zgodnie z poniższymi wzorami:

gdzie:

- – chwilowa różnica dla akcji podjętej w poprzednim stanie

- – nagroda otrzymana za akcję podjętą w poprzednim stanie

- – dyskonto (0-1); hiper parametr

- – największa wartość Q-value dostępna dla dowolnej akcji w danym stanie

- – Q-value akcji podjętej w poprzednim stanie

gdzie:

- – nowa wartość Q-value dla akcji podjętej w poprzednim stanie

- – stara wartość Q-value dla akcji podjętej w poprzednim stanie

- – współczynnik nauki; hiper parametr

- – chwilowa różnica dla akcji podjętej w poprzednim stanie

- w celu umożliwienia lepszej eksploracji, wybór następnej akcji nie byłby do końca deterministyczny w fazie nauki – agent z pewnym prawdopodobieństwem mógłby wybrać akcję, która według jego dotychczasowej wiedzy nie byłaby najlepsza dla danego stanu

- nauka agenta przebiegałaby w iteracjach; przejście do następnej iteracji byłoby możliwe po osiągnięciu stanu terminalnego (np. wylądowanie spodkiem w zadanym miejscu, wylecenie spodkiem poza obszar)

- podsumowanie algorytmu w punktach:

1. Zainicjuj Q-table

2. Wybierz akcję na podstawie Q-table z pewną dozą losowości

3. Przeprowadź wybraną akcję i przejdź do następnego stanu

4. Otrzymaj nagrodę i oblicz tymczasową różnicę (TD)

5. Zaktualizuj Q-value dla poprzedniego stanu

6. Przejdź do punktu 2, jeśli agent nie znajduje się w stanie terminalnym

1. **Planowane eksperymenty numeryczne**
2. Obserwowanie wpływu wartości hiper parametrów na jakość nauki
3. Zmiana Q-table (redukcja wymiarowości; np. branie pod uwagę tylko pozycji spodka przy nauce)
4. Wpływ ilości etapów nauki na końcową jakość nauki
5. **Technologia, w której realizowany będzie projekt**

Projekt będzie realizowany z użyciem języka Python w środowisku Box2D, będącego częścią Gym OpenAI. Platformą bazową będzie Windows 10, ale kod będzie możliwy do uruchomienia na dowolnej platformie.

Korzystanie z wyżej wymienionego środowiska Gym jest utrudnione na platformie Windows, ale nie niemożliwe. W celu poprawnego działania środowiska na Windows 10 konieczne jest:

* zainstalowanie Microsoft Visual C++ Build Tools
* stworzenie wirtualnego środowiska z użyciem Anacondy oraz zainstalowanie następujących pakietów:

- z użyciem pip: gym, Box2D

- z użyciem Condy: swig

* zainstalowanie Xming oraz nieprzerwane działanie tego programu w trakcie działania środowiska