Mikalai Stelmakh, 316951

WSI lab 6 - Metody uczenia się ze wzmocnieniem

Zadanie

Zaimplementować algorytm Q-Learning i użyć go do wyznaczenia polityki decyzyjnej dla problemu „Q-Uber”.

Uruchomienie programu

python3.10 qlearning.py [-h] [--size SIZE] [--start START] [--goal GOAL] [--p_hole P_HOLE] [--steps STEPS]

Żeby dowiedzieć się więcej na temat argumentów należy użyć flagi "-h".

Rozwiązanie

Plansza

Przyjęto, że mapa zawsze jest w formie kwadratu (standardowo 8x8). Liczba dziur jest wyznaczana z prawdopodobieństwa ich wystąpienia, podanego przez użytkownika. Przy próbie wyjazdu poza obszar planszy agent zostaje na tym samym miejscu. Punkty startu i celu są podawane przez użytkownika. Do sprawdzenia czy istnieje ścieżka pomiędzy startem a celem użyto algorytmu DFS.

Funkcja nagrody

Jako funkcje nagrody użyto trzech różnych metod:

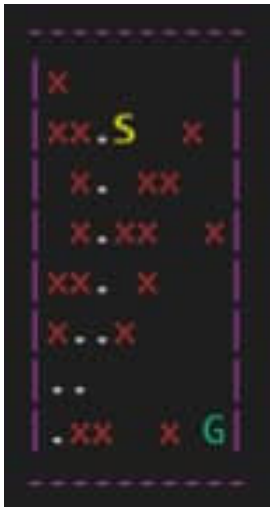
- 1 za dojście do celu, 0 w przeciwnym wypadku
- 1 za dojście do celu, -1 za wpadnięcie w dziurę, 0 w przeciwnym wypadku.
- 10 za dojście do celu, -1 za wpadnięcie w dziurę, 0 w przeciwnym wypadku.

Wpływ funkcji nagrody na wynik działania agenta pokazany jest w dalszej części sprawozdania.

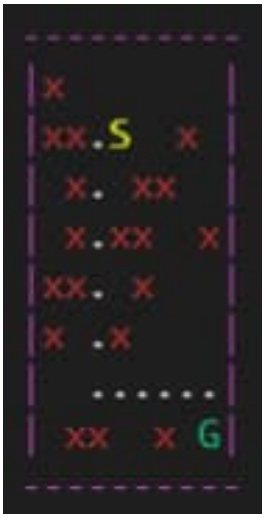
Wyniki

Przykład działania

Dla niedouczonego modelu
(100 000 epizodów):



Dla nauczonego modelu
(1 000 000 epizodów):



Gdzie:

- S – start,
- G – cel,
- x – dziura,
- |, - -- granica mapy,
- . – trasa do celu

Jak widać, agentowi nie wystarczyło 100 000 epizodów do znalezienia trasy do celu, więc wybiera on kierunek z najmniejszą karą, czyli waha się pomiędzy dwoma punktami.

Badanie wpływu parametrów

Domyślne ustawienia:

- Rozmiar planszy: 8x8
- Punkt startowy: 0
- Punkt docelowy: 56
- Learning rate: 0.8
- Discount rate: 0.85
- Epsilon: 0.5
- Prawdopodobieństwo dziury: 25%
- Liczba epizodów: 100 000
- Dozwolona liczba kroków: 20

Funkcja nagrody

Dla każdego przypadku program został uruchomiony 20 raz.

1) 1 za dojście do celu, 0 w przeciwnym wypadku
2) 1 za dojście do celu, -1 za wpadnięcie w dziurę, 0 w przeciwnym wypadku.
3) 10 za dojście do celu, -1 za wpadnięcie w dziurę, 0 w przeciwnym wypadku.

Epizody/funkcja nagrody	Funkcja 1	Funkcja 2	Funkcja 3
1000	10:0	11:0	13:0
5000	20:0	20:0	20:0
10000	20:0	20:0	20:0
100000	20:0	20:0	20:0

Jak widać, karanie agenta za wpadnięcie do dziury oraz zwiększenie nagrody za dojście do celu zwiększyło skuteczność algorytmu.

Learning rate

Epizody/learning rate	0.3	0.5	1
1000	16:0	9:0	10:0
5000	20:0	20:0	20:0
10000	20:0	20:0	20:0
100000	20:0	19:1	20:0

Współczynnik uczenia określa długość kroku algorytmu, czyli stosunek nowej wartości z tabeli Qtable do starej wartości. Dla małych problemów, jak w naszym przypadku, dobrym wyborem będzie dobranie stosunkowo małego współczynnika uczenia, natomiast dla większych problemów zbyt mały współczynnik będzie powodował zbyt wielką ilość iteracji, potrzebnych do nauczenia agenta.

Discount rate

Epizody/discount rate	0.3	0.5	1
1000	11:0	15:0	3:0
5000	20:0	20:0	0:0
10000	20:0	20:0	0:0
100000	19:1	20:0	0:0

Discount rate służy do zrównoważenia natychmiastowej i przyszłej nagrody, czyli jeśli $\gamma = 0$, agent dba tylko o natychmiastowej nagrodzie, jeśli $\gamma = 1$, agent dba o wszystkie przyszłe nagrody.

W przypadku tak dobranych hiper-parametrów oraz rozmiaru planszy najlepszym rozwiązaniem okazało się zrównoważenie natychmiastowej i przyszłej nagrody, czyli ustawienie wartości współczynnika na 0.5.

Wnioski

Jak widać, we wszystkich przypadkach gracz używający algorytmu Qlearning dominuje nad graczem wybierającym losową akcję. Zachowanie algorytmu bardzo zależy od dobranych hiper-parametrów, przyjętej strategii, czy wykorzystanej funkcji nagrody. Jak widać z otrzymanych wyników, dla planszy o takim rozmiarze i tak dobranych punktach startu/celu wystarczy około 5000 epizodów, by nauczyć agenta. Oczywiście, że dla planszy o większym rozmiarze lub dla dłuższej trasy potrzebował on będzie większej ilości iteracji.