Wydział Elektroniki
i Technik Informacyjnych
POLITECHNIKA WARSZAWSKA

Wstęp do Sztucznej Inteligencji

Ćwiczenie nr 6:
Uczenie ze wzmocnieniem

Kaczmarski Robert
293377

Warszawa 2022

Polecenie

Napisać generator labiryntu i sprawdzić czy istnieje w nim ścieżka od startu do mety. Następnie zaimplementować algorytm Q-Lerning, który będzie wykorzystywany przez samochód Q-uber do uczenia się przechodzenia labiryntu. Drugi samochód będzie poruszał się wykonując losowe ruchy.

Użyte narzędzia

Do wykonania zadania został użyty Python w wersji 3.8.2. oraz następujące biblioteki:

- numpy ==1.22.0
- matplotlib==3.5.1

Generacja i sprawdzenie labiryntu

Generowany labirynt ma wymiary NxN dla N >3. Dolne ograniczenie wymiarów planszy jest spowodowane tym, że na obwodzie planszy znajdują się zabronione pola, czyli nawiązując do kontekstu zadania stoją tam bezpłatni stażyści. Załatwia to przypadek, gdy samochód wyjdzie poza labirynt. Labirynty są generowane do skutku, aż zostanie wygenerowany poprawny labirynt. Poprawny labirynt to taki, w którym istnieje ścieżka od punktu początkowego do punktu końcowego. Poprawność labiryntu jest sprawdzana algorytmem DFS.

Opisz poszczególnych elementów w algorytmu Q-lerning

Środowisko: badanym środowiskiem jest klasyczny kwadratowy labirynt, mający co najmniej jedną drogę od punktu startowego do punktu końcowego.

Agent: agentem jest samochód Q-Uber, który próbuje znaleźć najkrótszą drogę od punktu startowego do punktu końcowego. Aby osiągnąć cel, agent eksperymentuje wykorzystując przeszłe doświadczenia (epizody). Rozwiązanie znalezienia najkrótszej drogi zostaje osiągnięte, gdy agent znajdzie optymalną sekwencję stanów, w których skumulowana suma nagród jest maksymalna.

Akcje: ponieważ samochody poruszają się przy użyciu metryki Manhattan, można wyszczególnić cztery akcje:

- Ruch w lewo
- Ruch w górę
- Ruch w prawo
- Ruch w dół

Stan: aktualna pozycję agenta w środowisku.

Stany: wszystkie możliwe pozycje agenta w środowisku.

Nagroda: zbierana przez agenta po wykonaniu akcji. Nagrody są przyznawane na podstawie aktualnej pozycji Q-Uber w labiryncie. Labirynt ma postać macierzy 2D, w której każde pole ma jakąś wartość. Dobrane wartości pól:

- Pole ze stażystą : -20pkt
- Puste pole: -1 pkt
- Pole będące polem końcowym: 20pkt

Pola z ujemnymi wartościami to kary. Wartość pustego pola również jest traktowana jako kara, ponieważ trzeba zmusić Q-Uber do dotarcia do mety.

Polityka: określa jak powinien zachować się agent w danej sytuacji, czyli jaką powinien wykonać akcję. Politykę można określić jako funkcję $\pi(stan) = akcja$.

Epizod: stan, który kończy się, gdy agent znajduje się w momencie, gdzie nie może już podjąć nowych akcji.

Współczynnika uczenia się α – określa w jakim stopniu nowo zdobyte informacje zastępują stare.

$\alpha = 0$ oznacza, że agent nie uczy się niczego

$\alpha = 1$ oznacza, że agent bierze pod uwagę tylko najnowsze informacje

Współczynnika dyskontowania γ – określa znaczenie przeszłych nagród.

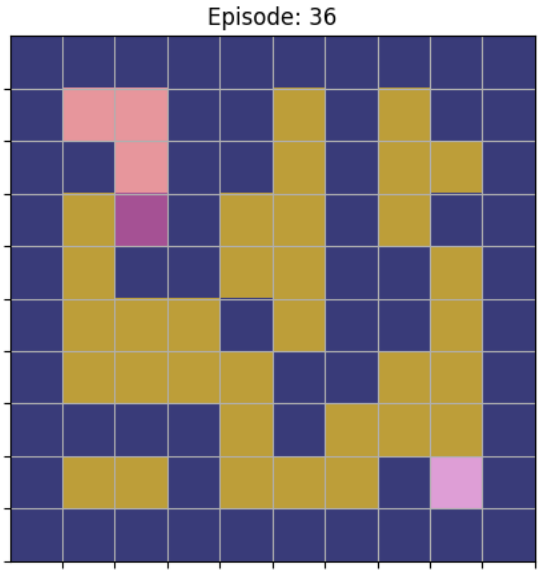
γ = 0 oznacza, że agent stanie się krótkowzroczny i będzie brał pod uwagę tylko bieżące nagrody

γ = 1 oznacza, że agent będzie dążył do uzyskania wysokiej nagrody w długim okresie

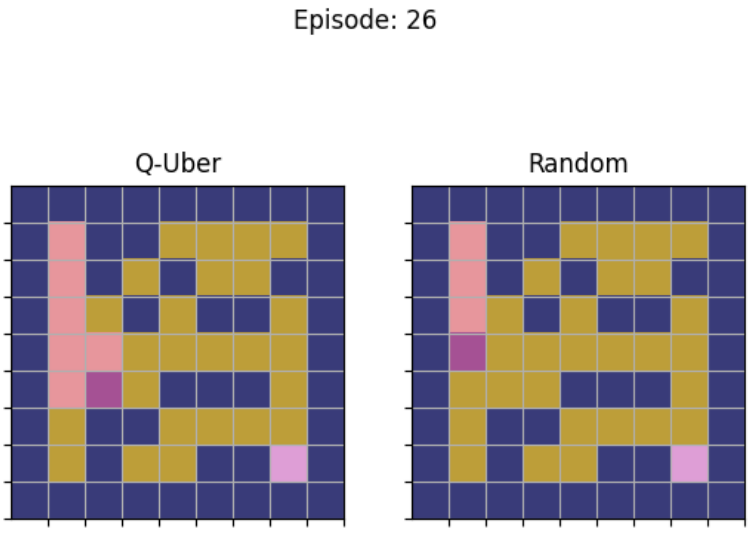
Początkowo tablica Q(Stan, Akcja) jest zainicjowana zerami. W zaimplementowanym algorytmie tablica Q to słownik zawierający wszystkie możliwe Stany i podpowiadające im Akcje, gdzie Stan to aktualna pozycja Q-Uber, a Akcje to lista wartości każdej możliwej akcji. Przykład: (1,1) : [0,0,0,0].

Wizualizacja

W programie możliwe jest wyświetlenie procesu uczenia się Q-Uber, jak również obu samochodów jednocześnie. Do wizualizacji została wykorzystana biblioteka matplotlib.



Rys. 1. Przykład wizualizacji uczenia się Q-Uber. Punkt początkowy (1,1), punkt końcowy (8,8).



Rys. 2. Przykład wizualizacji obu samochodów dla labiryntu 9x9.

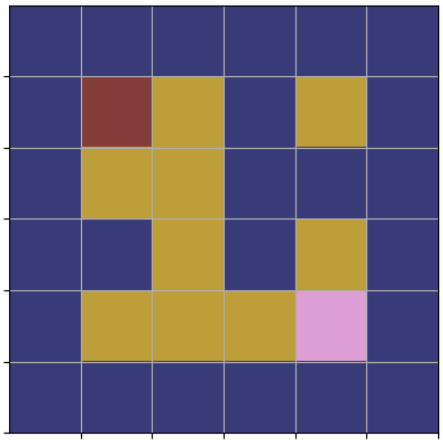
Z uwagi na ograniczony czas nie została dodana legenda, ale nietrudno domyślić się jaki kolor co oznacza.

Wyniki

W zaimplementowanym algorytmie Q-Uber zawsze wybiera akcję o największej wartości z tablicy Q(Stan) maksymalizując zysk, a losowy samochód zawsze wybiera losowy ruch.

W przypadku badania wpływu parametrów na działanie algorytmu, trudno jest wymyślić sensowne metryki. Przebadany został wpływ współczynnika uczenia się - α oraz współczynnika dyskontowania - γ. Sprawdzony został pierwszy epizod, w którym Q-Uber nauczył się, najmniejsza liczba kroków spośród wszystkich epizodów, w którym Q-Uber przeszedł labirynt oraz liczba epizodów , w których Q-Uber i Losowy samochodzik ukończyli labirynt.

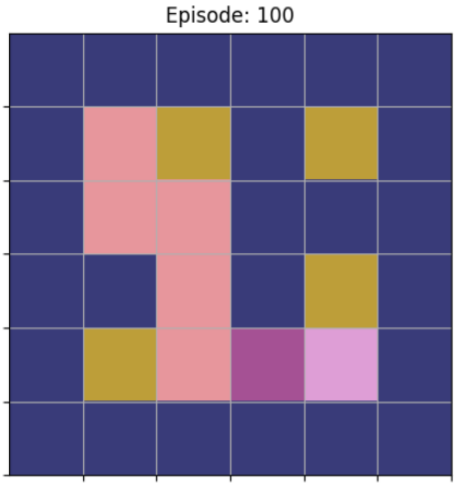
Punktem startowym zawsze był punkt (1,1), a punktem końcowym (N-2, N-2), gdzie N to rozmiar labiryntu. Przy generacji labiryntu prawdopodobieństwo wystąpienia stażysty zawsze 40%. Dla powtarzalności eksperymentów zawsze było ustawiane to samo ziarno.



Rys. 3. Wygląd labiryntu 6x6.

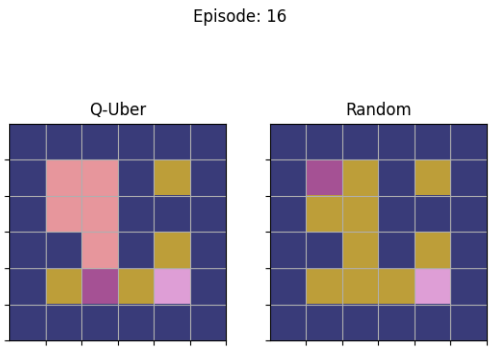
Tabela 1. Badanie wpływy parametrów algorytmu Q-Lerning dla labiryntu 6x6.

Łączna liczba epizodów	α	γ	Pierwsze przejście Q-Ubera	Minimalna liczba kroków	Liczba przejść labiryntu przez Q-Uber	Liczba przejść labiryntu przez losowy samochodzik
100	0	0,8	-	-	0	0
100	0,2	0,8	Epizod nr 12	6	88	0
100	0,4	0,8	Epizod nr 12	6	88	0
100	0,6	0,8	Epizod nr 12	6	88	0
100	0,8	0,8	Epizod nr 12	6	88	0
100	1	0,8	Epizod nr 12	6	88	0
100	0,8	0	Epizod nr 12	40*	3/17*	0
100	0,8	0,2	Epizod nr 12	6	88	0
100	0,8	0,4	Epizod nr 12	6	88	0
100	0,8	0,6	Epizod nr 12	6	88	0
100	0,8	1	Epizod nr 12	6	88	0

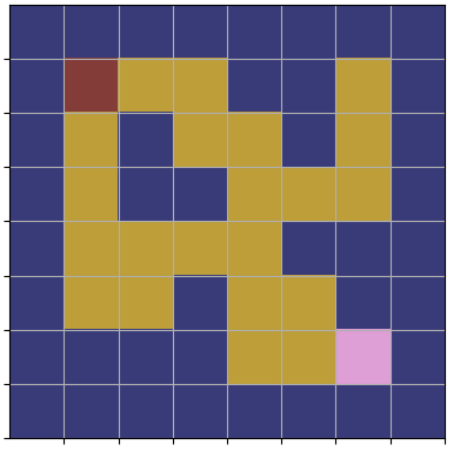


Rys. 4. Najkrótsza znaleziona ścieżka labiryntu 6x6.

* Dla $\gamma = 0$ program zawieszał się na 16 epizodzie (epizody liczone są od zera). Najprawdopodobniej było to spowodowane jakimś bugiem w kodzie. Pozycja, w której dochodziło do zawieszenia się programu została przedstawiona poniżej.



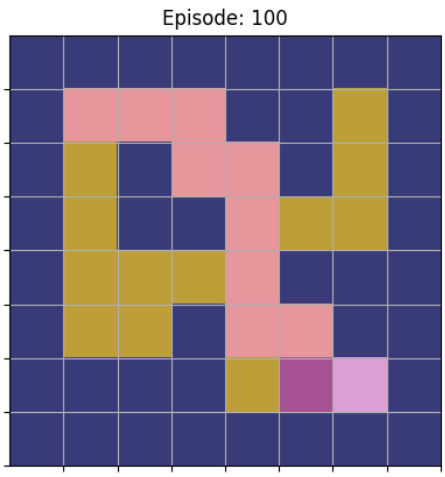
Rys. 5. Stan labiryntu, w którym program się zawieszał.



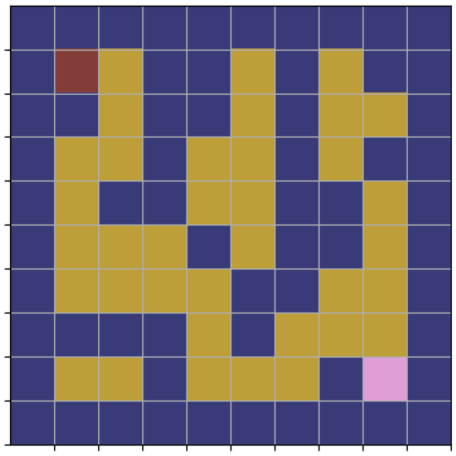
Rys. 6. Wygenerowany labirynt 8x8.

Tabela 2. Badanie wpływy parametrów algorytmu Q-Lerning dla labiryntu 8x8.

Łączna liczba epizodów	α	γ	Pierwsze przejście Q-Ubera	Minimalna liczba kroków	Liczba przejść labiryntu przez Q-Uber	Liczba przejść labiryntu przez losowy samochodzik
100	0	0,8	-	-	0	0
100	0,2	0,8	Epizod nr 31	10	62	0
100	0,4	0,8	Epizod nr 31	10	62	0
100	0,6	0,8	Epizod nr 31	10	62	0
100	0,8	0,8	Epizod nr 32	10	62	0
100	1	0,8	Epizod nr 34	10	64	0
100	0,8	0	-	-	0/26*	0
100	0,8	0,2	Epizod nr 32	10	62	0
100	0,8	0,4	Epizod nr 32	10	61	0
100	0,8	0,6	Epizod nr 32	10	62	0
100	0,8	1	Epizod nr 32	10	61	0



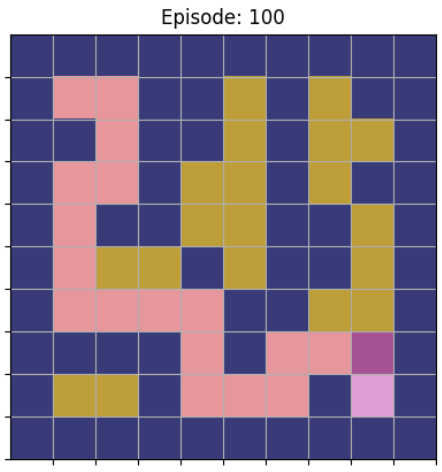
Rys. 7. Najkrótsza znaleziona ścieżka w labiryncie 8x8.



Rys. 8. Wygenerowany labirynt 10x10.

Tabela 3. Badanie wpływy parametrów algorytmu Q-Lerning dla labiryntu 10x10.

Łączna liczba epizodów	α	γ	Pierwsze przejście Q-Ubera	Minimalna liczba kroków	Liczba przejść labiryntu przez Q-Uber	Liczba przejść labiryntu przez losowy samochodzik
100	0	0,8	-	-	0	0
100	0,2	0,8	Epizod nr 38	18	44	0
100	0,4	0,8	Epizod nr 38	18	61	0
100	0,6	0,8	Epizod nr 38	18	61	0
100	0,8	0,8	Epizod nr 38	18	61	0
100	1	0,8	Epizod nr 39	18	61	0
100	0,8	0	-	-	0/14*	0
100	0,8	0,2	Epizod nr 38	18	61	0
100	0,8	0,4	Epizod nr 38	18	61	0
100	0,8	0,6	Epizod nr 38	18	61	0
100	0,8	1	Epizod nr 38	18	58	0



Rys. 9. Najkrótsza znaleziona ścieżka dla labiryntu o wymiarach 10x10.

Obserwacje i wnioski

Pomijając skrajne przypadki, gdy $\alpha = 0$ lub $\gamma = 0$ nie został zauważony znaczący wpływ zmiany parametrów na działanie algorytmu. Na podstawie wyników zamieszczonych w tabelach 1-3 można powiedzieć, że im większy współczynnik uczenia się tym większa jest liczba epizodów, w który Q-Uber dojechał do mety. Większy rozmiar labiryntu powodował to, że Q-Uber potrzebował więcej epizodów, żeby nauczyć się przechodzić labirynt.

W każdym badanym przypadku Q-Uber prędzej czy później nauczył się najkrótszej ścieżki przechodzenia labiryntu. Losowy samochód nigdy nie dotarł do punktu końcowego, ponieważ możliwość trafienia na stażystę nie była zabroniona.

Być może, że sprawdzenie znacznie większych labiryntów byłoby bardziej miarodajne.

Możliwa modyfikacja algorytmu to dodanie epsilona przy wyborze akcji, aby Q-Uber czasem wybrał losowy ruch zamiast zawsze maksymalizować zysk. Ulepszyłoby to proces eksploracji algorytmu.