WSI

Zadanie 6

Autor:
Michał Paradowski

5.24.2024 (V1.0)

Contents

1 Zadanie

Cel zadania

Celem zadania jest zaimplementowanie przez Państwa algorytmu z podstawami uczenia przez wzmacnianie (Reinforcement learning) w rozwiązaniu problemu Frozen Lake.
Frozen Lake to środowisko dostępne w bibliotece gym – link tutaj. Przykładowe uruchomienie środowiska przedstawione zostało tutaj. Frozen Lake implementuje agenta poruszającego się po lodowej planszy w kierunku celu i unikającego dziurawych pól. Agent może poruszać się w czterech kierunkach w poziomie.
Plansza składa się z czterech typów pól:

- Start (S) - pole startowe,
- Frozen (F) - bezpieczne pole, po którym agent się porusza,
- Hole (H) - dziura, wejście na to pole kończy grę,
- G (Goal) - cel, do którego dąży agent.

Po uruchomieniu środowiska frozen lake muszą Państwo zaimplementować algorytm Q-learning w celu nauki agenta oczekiwanego sposobu działania.

Algorytm Q-learning powinien obejmować:

- Stworzenie Q - tabeli przechowującej wartość oczekiwanej nagrody dla każdej pary stan-akcja.
- Decyzje agenta - Agent wykorzystuje Q-tabelę do podejmowania decyzji o tym, którą akcję wybrać w danym stanie. W tym celu stosuje strategię epsilon-greedy, która balansuje pomiędzy eksploracją (wybieranie losowych akcji) a eksploatacją (wybieranie najlepszych akcji według Q-tabeli).
- Uczenie przez wzmacnianie - Proces uczenia polega na aktualizacji wartości Q w Q-tabeli na podstawie doświadczeń agenta. Po wykonaniu akcji i otrzymaniu nagrody, agent aktualizuje wartość Q dla danej pary stan-akcja zgodnie z równaniem aktualizacji Q-learningu.

Krok uczenia można opisać jako:

$$q^{\text{new}}(s,a) = (1-\alpha)q(s,a) + \alpha\left(R_{t+1} + \gamma \max_{a'} q(s',a')\right)$$

- $q^{\text{new}}(s, a)$ to nowa wartość funkcji Q po akcji a w stanie s,
- α to współczynnik uczenia,
- R_{t+1} to otrzymana nagroda,
- γ to współczynnik dyskontowania określający znaczenie przyszłych nagród w porównaniu z natychmiastowymi,
- $\max_{a'} q(s', a')$ to maksymalna wartość Q spośród wszystkich możliwych do podjęcia akcji a' w nowym stanie s'

Sprawozdanie

W sprawozdaniu należy umieścić: opis działania algorytmu oraz zbadać wpływ parametrów: uczenia, dyskontowania i eksploracji na zbieżność algorytmu – momentu gdy algorytm osiąga stabilny stan i dalsze uczenie nie przynosi znaczących zmian.
W sprawozdaniu należy umieścić wykres sumy wszystkich nagród od kroku uczenia.
Należy przedstawić wyniki dla 3 uruchomień algorytmu z różną wartością random seed.

2 algorytm

Algorytm Q-learning jest jednym z najbardziej popularnych algorytmów uczenia ze wzmocnieniem (reinforcement learning). Jego celem jest nauka optymalnej strategii w środowisku, gdzie stan i akcje są znane, ale nie jest znana dokładna funkcja nagrody. Algorytm zwraca tablicę Q o rozmiarach n na m, gdzie n to liczba stanów, a m to liczba możliwych akcji dla tego stanu. każdy element tej tablicy to 'wycena' danej akcji w danym stanie. Tablica po wykonaniu algorytmu powinna przedstawiać strategię optymalną, to znaczy dla każdego stanu u najbardziej opłacalna akcja v powinna odpowiadać $Q(u, v)$ takiemu że $Q(u, v) = \max_{a'} Q(u, a')$.

1. **Inicjalizacja parametrów**: Parametry, takie jak liczba epizodów (num_episodes), współczynnik uczenia (α), współczynnik dyskontujący (γ), wartość epsilon (ϵ-greedy) i maksymalna liczba kroków na epizod (max_steps) są wczytywane z przekazanych parametrów.

2. **Inicjalizacja tablicy** Q: Na początku algorytmu tworzona jest tablica Q, która reprezentuje oczekiwaną wartość (oczekiwaną sumę nagród) dla każdej pary stanu i akcji. Tablica Q jest inicjalizowana na początku wszystkimi wartościami zerowymi, ponieważ na początku nie ma wiedzy na temat optymalnych wartości dla stanów i akcji.

3. **Pętla epizodów**: Algorytm wykonuje pętlę po wszystkich epizodach, w których agent oddziałuje ze środowiskiem.

4. **Inicjalizacja epizodu**: Na początku każdego epizodu stan agenta jest resetowany do stanu początkowego środowiska.

5. **Pętla kroków w epizodzie**: W każdym kroku agent podejmuje decyzję na podstawie strategii eksploracji-wykorzystania (epsilon-greedy), a następnie aktualizuje wartości w tablicy Q zgodnie z regułą Q-learningu.

6. **Aktualizacja wartości w tablicy** Q: Wartości w tablicy Q są aktualizowane na podstawie nagrody otrzymanej za wykonanie danej akcji w danym stanie oraz przewidywanej przyszłej nagrody w następnym stanie. Aktualizacja ta odbywa się zgodnie z równaniem Bellmana dla optymalnej funkcji wartości akcji:

$$Q(s, a) \leftarrow (1 - \alpha) \cdot Q(s, a) + \alpha \cdot \left(r + \gamma \cdot \max_{a'} Q(s', a') \right)$$

   gdzie:

   - $Q(s, a)$ - wartość funkcji akcji dla stanu s i akcji a,
   - α - współczynnik uczenia,

- r - nagroda otrzymana po wykonaniu akcji a w stanie s (w przypadku frozen lake, nagroda jest zdefiniowana jako 0 za wpadnięcie do dziury lub trafienie na lód oraz 1 za dotarcie do celu),
- γ - współczynnik dyskontujący,
- $\max_{a'} Q(s', a')$ - maksymalna wartość funkcji akcji dla następnego stanu s'.

3 Wpływ parametrów

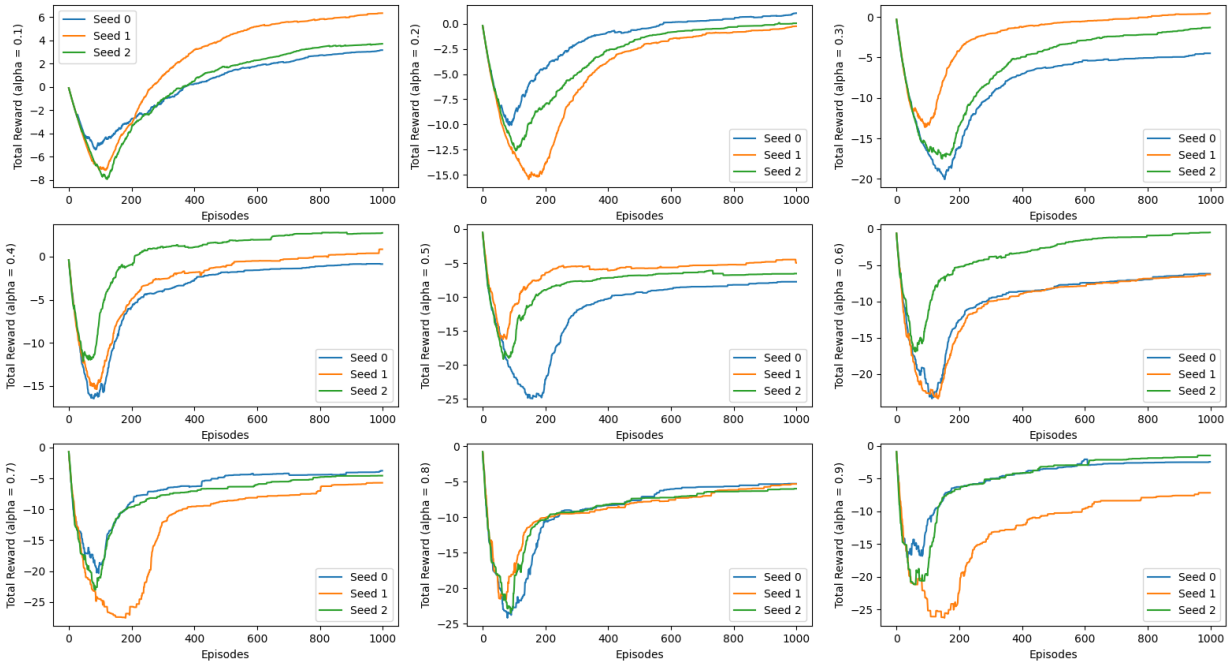
3.1 Parametr uczenia (α)

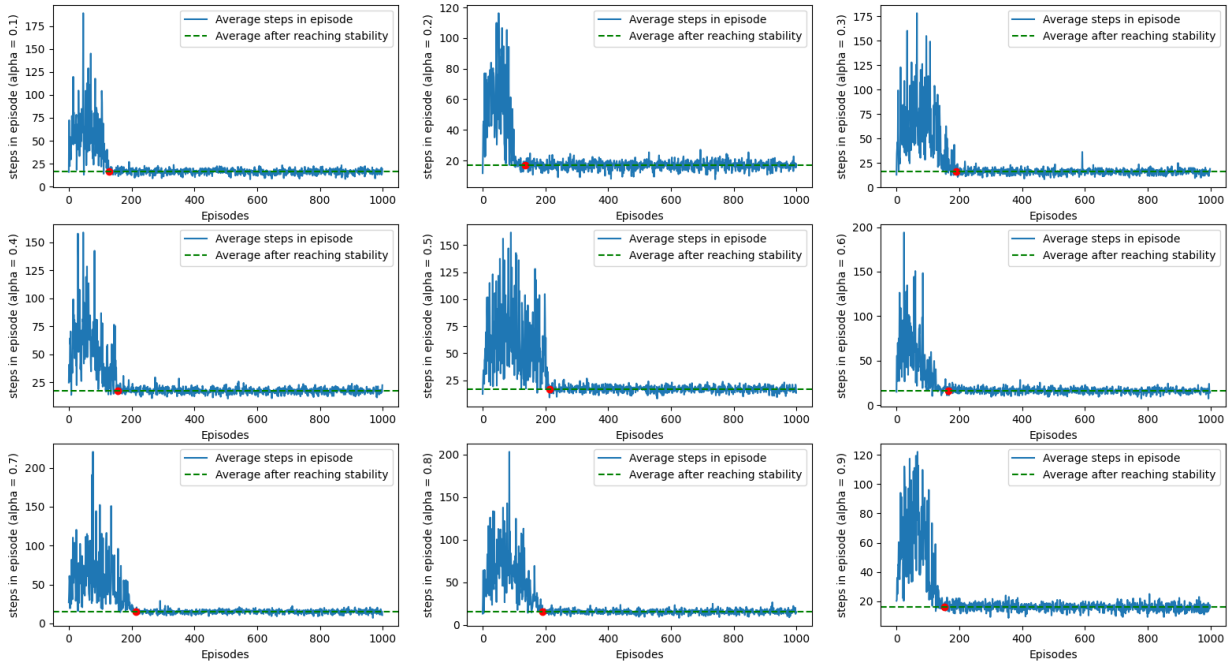
Parametr α wpływa na szybkość uczenia się, to znaczy od niego zależy jak duży wpływ na nową wycenę elementu tablicy Q ma pojedyncza iteracja uczenia się. Gdy parametr α jest mały, wycena pozostaje w dużym stopniu niezmieniona gdyż udział w średniej ważonej $q_{old}(s, a)$ jest przeważający $(1-\alpha)$. Im większa wartość α tym większy wpływ ma część średniej ważonej z wagą α. Przedstawię po dwa wykresy dla kilku różnych wartości α. pierwszy wykres przedstawiać będzie sumaryczną wycenę wszystkich wartości tablicy Q po danych epizodzie (oddzielnie dla 3 różnych wartości random_seed). Drugi wykres przedstawiać będzie liczbę kroków algorytmu w danym epizodzie (uśrednioną dla 3 różnych wartości random_seed). Jeśli w danym epizodzie gra skończy się trafieniem na dziurę, liczba kroków ustawiona zostanie na wartość max_steps. Tę uśrednioną liczbę kroków w i-tym epizodzie oznaczę przez $p[i]$. Na drugim wykresie zaznaczę też czerwoną kropką moment osiągnięcia stabilności przez algorytm. k-ty epizod nazwiemy epizodem osiągnięcia stabilności, jeśli k będzie pierwszą taką liczbą, że

$$\sum_{i=k+1}^{k+15} p[i] - \sum_{i=k+16}^{k+30} p[i] < c$$

Gdzie c to jakaś nieduża stała (na przykład 1). Zieloną linią oznaczę średnią wartość $p[i]$ dla $k \leq i \leq$ max_episodes.
Oto te dwa wykresy:
Parametry wejściowe:

- $\alpha \in \{0.1, 0.2, ..., 0.9\}$
- $\gamma = 0.8$
- $\epsilon = 0.2$
- max_steps = 1000 - współczynnik dyskontujący,
- episodes = 200
- gra na planszy 8x8


Seed 0
Seed 1
Seed 2
Total Reward (alpha = 0.1)
Total Reward (alpha = 0.2)
Total Reward (alpha = 0.3)
Total Reward (alpha = 0.4)
Total Reward (alpha = 0.5)
Total Reward (alpha = 0.6)
Total Reward (alpha = 0.7)
Total Reward (alpha = 0.8)
Total Reward (alpha = 0.9)
Episodes

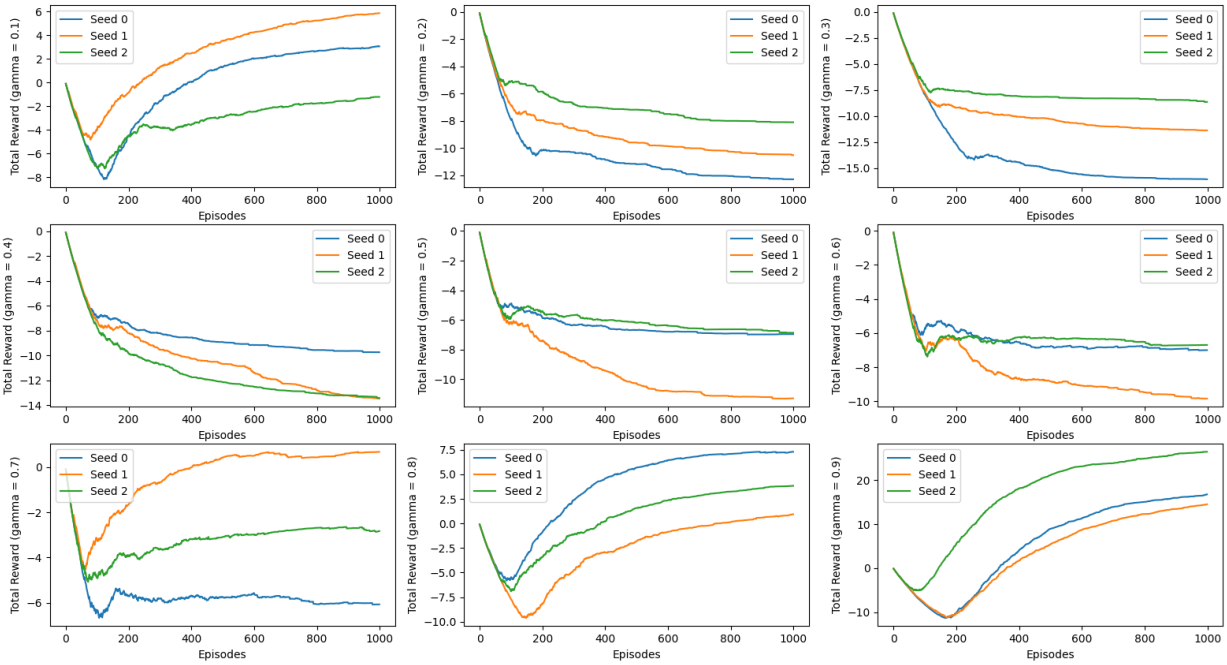

Average steps in episode
Average after reaching stability
steps in episode (alpha = 0.1)
steps in episode (alpha = 0.2)
steps in episode (alpha = 0.3)
steps in episode (alpha = 0.4)
steps in episode (alpha = 0.5)
steps in episode (alpha = 0.6)
steps in episode (alpha = 0.7)
steps in episode (alpha = 0.8)
steps in episode (alpha = 0.9)
Episodes

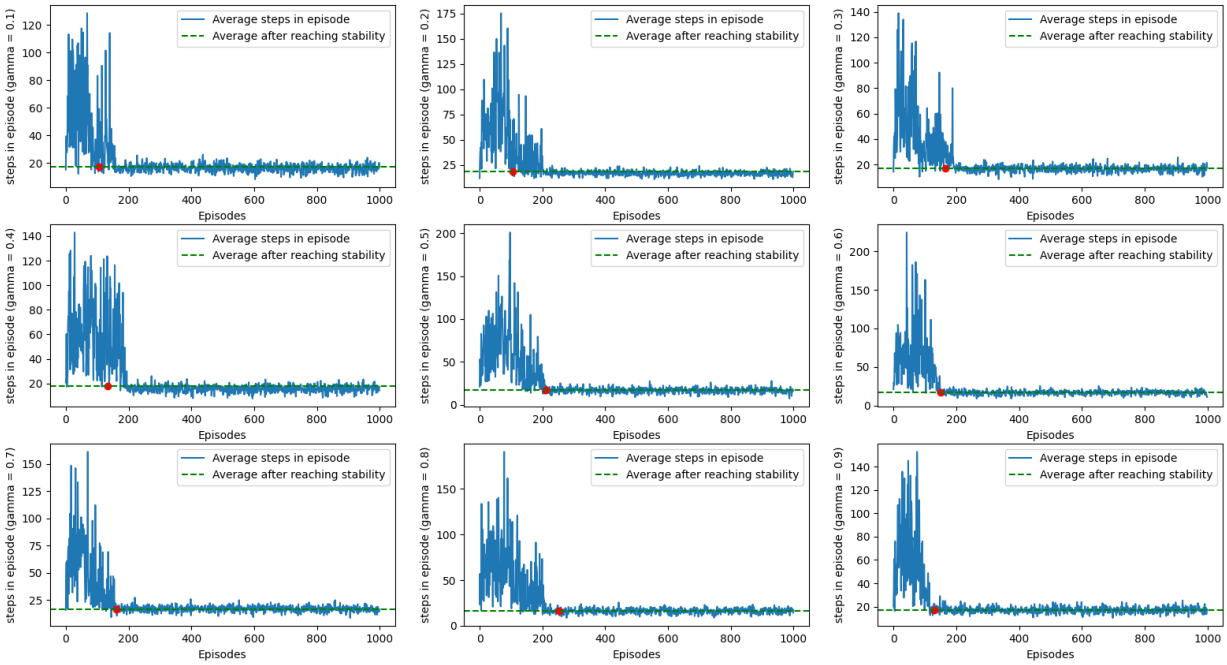
Widzimy, że jedynie dla $\alpha = 0.1$ suma nagród na planszy osiągała wartości znacząco wyższe od zera. Może to wynikać z tego, że dziur jest na planszy więcej niż celów (cel jest tylko jeden), więc na początku większość strategii będzie trafiało na dziury. Jeśli α jest małe, te początkowe porażki nie obniżą wycen na tyle by później zostały poniżej zera gdy algorytm znajdzie już odpowiednie ścieżki. Możemy zobaczyć, że dla wszystkich wartości α algorytm w końcu odnalazł właściwą ścieżkę, jednak dla mniejszych wartości α zrobił to nieco szybciej (około 170 epizodów), zaś później zajmowało mu to więcej czasu (około 200). Jeśli algorytm nie znalazłby dobrej ścieżki, wykresy ilości kroków nie stabilizowałyby się jako że przyjąłem że gdy algorytm trafi do dziur, $p[i] = \text{max\_steps} = 1000$. Dla wszystkich wartości α algorytm znajdował optymalną ścieżkę o długości 14 kroków co ma sens jako że plansza ma wymiary 8x8 a start i cel znajdują się na przeciwległych rogach. (optymalną ścieżkę wyznaczyłem wybierając zawsze akcję o najwyższej wycenie z tablicy Q).

3.2 Współczynnik dyskontowania (γ)

Im większa wartość γ, tym większy wpływ na wartość danego elementu Q mają elementy oddalone od niego. Jeśli na dojście z jakiegoś pola tablicy Q do innego potrzeba n akcji, to wpływ tego drugiego pola na to pierwsze będzie proporcjonalny do γ^n. Mniejsze wartości skutkować więc mogą bardziej lokalnymi strategiami, zaś zbyt duże nierozróżnianiem odległości od celu co może się skończyć nieoptymalną ścieżką.
Wykresy zostały zaprojektowane w pełni analogicznie: Parametry wejściowe:

- $\alpha = 0.1$
- $\gamma \in \{0.1, 0.2, ..., 0.9\}$
- $\epsilon = 0.2$
- max_steps = 1000 - współczynnik dyskontujący,
- episodes = 1000
- gra na planszy 8x8


Seed 0
Seed 1
Seed 2
Total Reward (gamma = 0.1)
Total Reward (gamma = 0.2)
Total Reward (gamma = 0.3)
Total Reward (gamma = 0.4)
Total Reward (gamma = 0.5)
Total Reward (gamma = 0.6)
Total Reward (gamma = 0.7)
Total Reward (gamma = 0.8)
Total Reward (gamma = 0.9)
Episodes

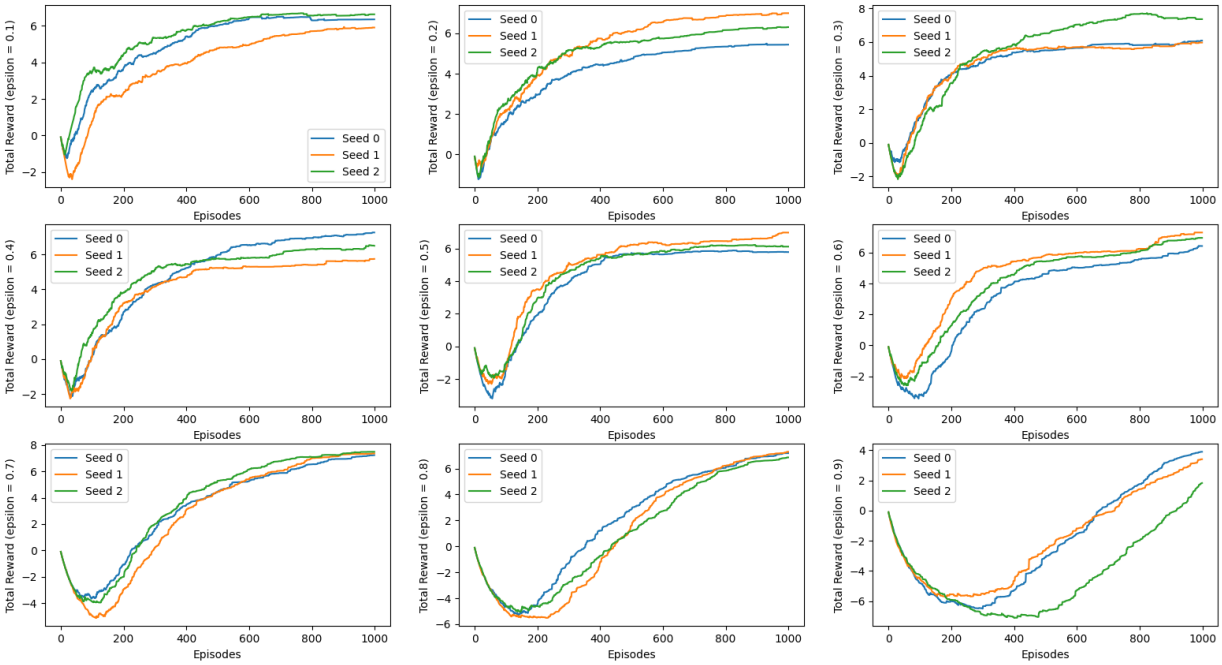

Average steps in episode
Average after reaching stability
steps in episode (gamma = 0.1)
steps in episode (gamma = 0.2)
steps in episode (gamma = 0.3)
steps in episode (gamma = 0.4)
steps in episode (gamma = 0.5)
steps in episode (gamma = 0.6)
steps in episode (gamma = 0.7)
steps in episode (gamma = 0.8)
steps in episode (gamma = 0.9)
Episodes

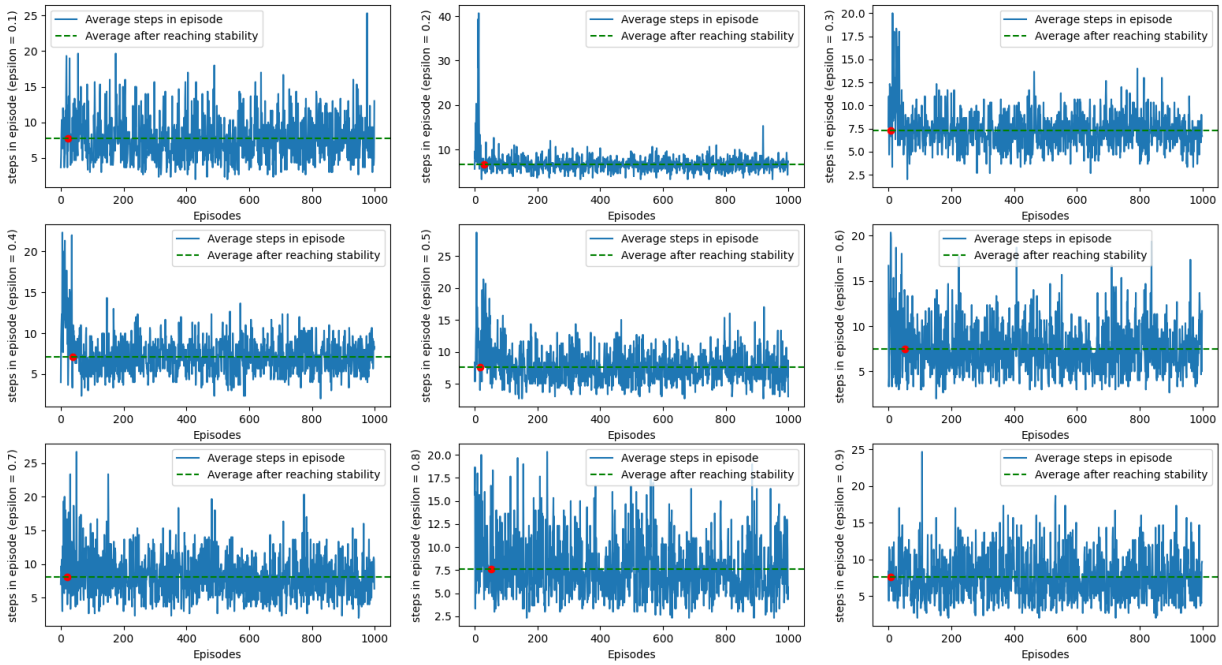
Pierwszą ciekawą obserwacją, jest dużo większy wpływ random_seed na rezultat. Szczególnie na wykresie $\gamma = 0.7$, seed 0 oceniło większość ścieżek bardzo negatywnie. Tego typu chaos jest spodziewany gdyż na ostateczny rezultat duży wpływ ma to czy najpierw trafimy na dziurę, czy do celu. Widzimy że i tutaj liczba epizodów do osiągnięcia stabilności przyjmowała wartości bliskie 200. każda z tych instancji algorytmu znalazła ścieżkę o długości 14.

3.3 Parametr eksploracji (ϵ)

I tutaj wykresy przedstawiają te same wartości:

- $\alpha = 0.1$
- $\gamma = 0.8$
- $\epsilon \in \{0.1, 0.2, ..., 0.9\}$
- max_steps = 1000 - współczynnik dyskontujący,
- episodes = 200
- gra na planszy 8x8


Total Reward (epsilon = 0.1)
Total Reward (epsilon = 0.2)
Total Reward (epsilon = 0.3)
Total Reward (epsilon = 0.4)
Total Reward (epsilon = 0.5)
Total Reward (epsilon = 0.6)
Total Reward (epsilon = 0.7)
Total Reward (epsilon = 0.8)
Total Reward (epsilon = 0.9)
Seed 0
Seed 1
Seed 2
Episodes


steps in episode (epsilon = 0.1)
steps in episode (epsilon = 0.2)
steps in episode (epsilon = 0.3)
steps in episode (epsilon = 0.4)
steps in episode (epsilon = 0.5)
steps in episode (epsilon = 0.6)
steps in episode (epsilon = 0.7)
steps in episode (epsilon = 0.8)
steps in episode (epsilon = 0.9)
Average steps in episode
Average after reaching stability
Episodes

Wpływ tego parametru jest zobrazowany najlepiej na wykresach. Parametr ten mówi nam, jaka jest szansa na odchylenie się od strategii zapisanej w dotychczasowej tablicy Q. Jeśli ϵ jest małe, zazwyczaj wybierzemy jedną z najlepiej wycenianych obecnie akcji. W przeciwnym wypadku szansa na wybranie akcji losowej jest większa. Na pierwszy rzut oka widać, że dla $\epsilon = 0.2$ Stabilność wykresu liczby kroków jest największa, a wykres sumy nagród najszybciej dąży do maksimum. Dlatego też właśnie tę wartość wybrałem do wcześniejszych eksperymentów. Warto zaznaczyć jednak że i tu, wszystkie instancje odnalazły ścieżkę o długości 14 prowadzącą do celu.

3.4 Przykładowa tablica Q oraz przebieg strategii optymalnej

Przykładowa instancja algorytmu:

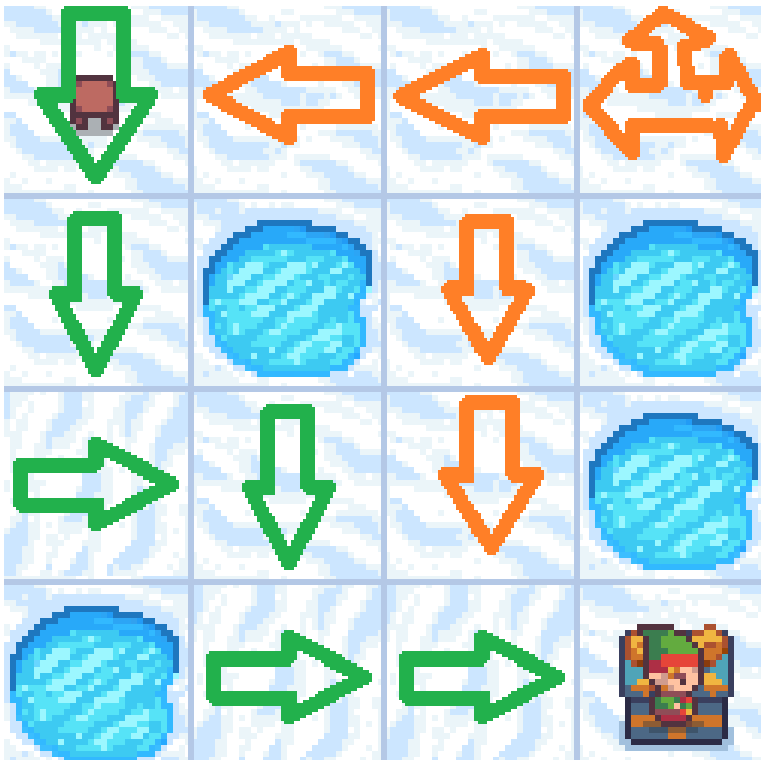
- $\alpha = 0.1$
- $\gamma = 0.8$
- $\epsilon = 0.2$
- max_steps = 1000 - współczynnik dyskontujący,
- episodes = 10000
- gra na planszy 4x4

Poniżej przedstawię zawartość tablicy Q. każdy element tablicy 4x4 odpowiada polu planszy. Każda z czterech liczb to odpowiednio:

- lewa-górna - wycena kroku w lewo
- prawa-górna - wycena kroku w dół
- lewa-dolna - wycena kroku w prawo
- prawa-dolna - wycena kroku w górę

0.2611 0.3277 0.2032 0.2606	0.2609 -0.4686 0.0128 0.0812	0.0898 0.0 0.0 0.0	0.0 -0.19 0.0 0.0
0.3268 0.4096 -0.9948 0.2602	0.0 0.0 0.0 0.0	-0.19 0.6167 -0.19 0.005	0.0 0.0 0.0 0.0
0.4092 -0.9943 0.512 0.3269	0.4049 0.6304 0.64 -0.9953	0.5082 0.8 -0.9921 0.44378	0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0	-0.5217 0.2028 0.7997 0.2671	0.6137 0.7963 1.0 0.6348	0.0 0.0 0.0 0.0

Tak więc optymalna ścieżka, to:



widać że dla pól drugiego, trzeciego i czwartego w pierwszym wierszu ścieżka nie została dobrana

optymalnie, jednak głównym celem było znalezienie ścieżki z lewego górnego rogu. Przy innych parametrach wejściowych na pewno udałoby się uzyskać strategię optymalną dla całej planszy. Po puszczeniu algorytmu z tymi samymi parametrami ale dla 10000 epizodów wynikowa strategia była w pełni optymalna.

List of Figures