Andrzej Pultyn

325213

WSI – zadanie 6

- Wstęp

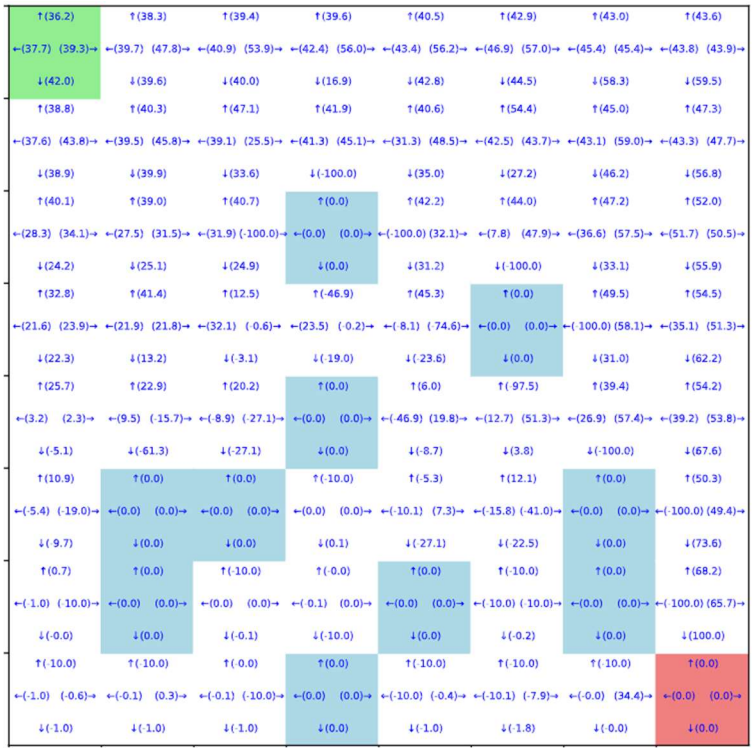
Celem zadania było zaimplementowanie algorytmu Q-learning, oraz użycie go do rozwiązania problemu FrozenLake8x8. Problem ten polega na tym, że mamy agenta, który ze startu musi dotrzeć do mety, omijając dziury w zamarzniętym jeziorze. Domyślna punktacja jest następująca – za dotarcie do mety otrzymuje 100 punktów, za wpadnięcie do dziury -100, w przeciwnym wypadku 0.

- Trening

Trening polega na stworzeniu tzw. QTable, czyli tabeli stanów, w jakich może się znaleźć postać w danym momencie (w naszym przypadku stanów jest 64, każdy odpowiada jednemu polu na planszy), oraz określenie wartości dla każdej możliwej akcji (w naszym przypadku są 4 akcje – pójście w jeden z 4 kierunków). Następnie, algorytm może albo wykonać najlepszy ruch w danej pozycji (ruch o najwyższym wyniku), albo wykonać ruch losowy. Szansę na losowość sterujemy parametrem epsilon. Dodatkowym utrudnieniem jest poślizg, czyli dodatkowy losowy ruch, o ile pierwotny w danej epoce nie zakończył gry. Szansą na poślizg sterujemy parametrem slippery_rate. Po każdym ruchu, specjalna funkcja wylicza wartość danego ruchu.

Warte odnotowania jest, że poślizg bardzo zaburza ocenę ruchu. Przeanalizuję to na przykładzie: znajdujemy się na poniższej mapie na polu o 2 wyższym od mety. Wykonujemy ruch w dół. Niestety, pech chciał, że się poślizgnęliśmy w lewo. Wpadliśmy do dziury. Algorytm w takiej sytuacji potraktuje nasz pierwotny krok w dół jako bezpośrednie wpadnięcie do dziury. Dlatego ważne jest danie chociaż odrobiny losowości w ruchach, bo inaczej taki poślizg mógłby zablokować drogę do mety.

Na koniec treningu otrzymujemy przykładowy QTable:

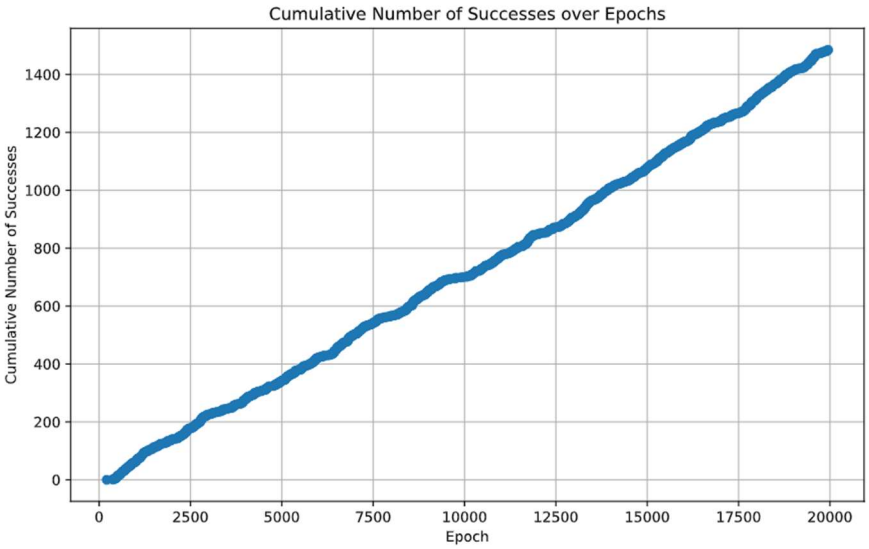


Kolorem zielonym oznaczony został start, czerwonym meta, a dziury są jasnoniebieskie.

- Testy

  Testowanie polegało na wykonanie treningu z odpowiednimi parametrami (parametry dla poszczególnych treningów można znaleźć w plikach json). Następnie, wykonywałem 1000 prób, w których agent brał zawsze najlepszy ruch w pozycji, jednak dalej z występującym poślizgiem. Ustawiłem limit kroków w pojedynczym teście na 100, żeby uniknąć pętli.

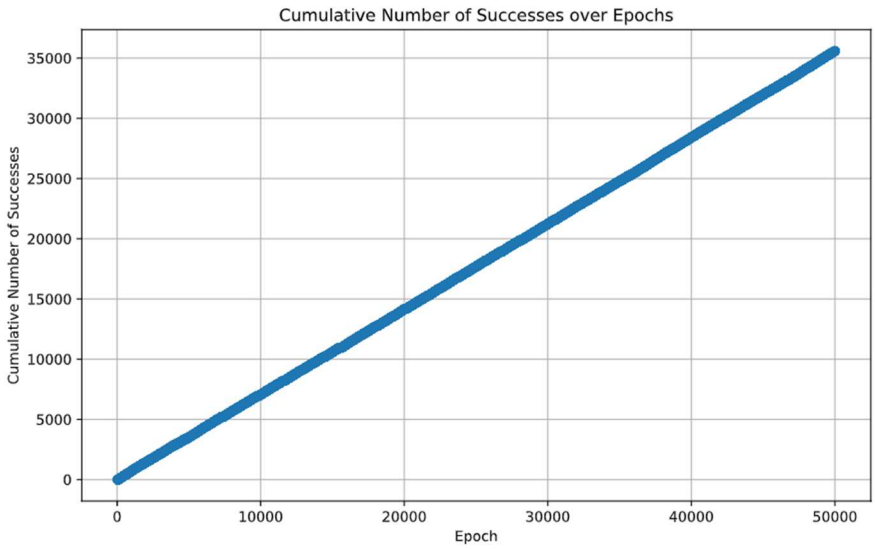
  Moje rozwiązanie posiada 2 systemy przyznawania punktów. Pierwszy jest domyślny, czyli punkty za dotarcie do mety, ujemny punkty za wpadnięcie do dziury. Ten algorytm sprawdziłem na 2 mapach, jednej bardziej otwartej (map.txt), a drugiej bardziej zamkniętej (map3.txt). Są to treningi zawierające w nazwie pliku no_bonus. Dla każdej mapy dałem 2 treningi – z dużą losowością i z małą. Okazuje się, że tylko w jednym przypadku algorytm odnalazł drogę do mety – było to na otwartej mapie i z dużą losowością (pliki zaczynające się od 1_slip_no_bonus_high_random). Podczas treningu tylko ok. 7% epok kończyło się sukcesem. Podczas testów wypadł dużo lepiej, ponieważ aż 86,9% testów kończyło się sukcesem, co przy szansie na poślizg równej 30% jest naprawdę niezłym wynikiem. Natomiast wykres ilości sukcesów podczas treningu jest bardzo nieregularny, co wraz z niewielką ich ilością wskazuje na czyste szczęście algorytmu:



- Ulepszony algorytm

Ulepszając algorytm, dodałem mu nową nagrodę i nową karę. Algorytm otrzymuje dodatkową nagrodę za każdy ruch, który go przybliża do mety (i karę za oddalanie się), oraz otrzymuje karę za każdy ruch który by go zderzył z krawędzią mapy (w mojej implementacji ruch na ścianę liczy się jako wykonany ruch, ale sam agent nie zmienia pozycji). Ten algorytm również przetestowałem na tych samych dwóch mapach, znowu z dużą i małą losowością. Od razu mogłem odrzucić testy z dużą losowością, ponieważ nie dawały one się wykazać algorytmowi. Ciekawie zaczęło się dziać przy niskiej losowości:

  - Na otwartej mapie algorytm podczas treningu uzyskał 71% sukcesów, oraz podczas testów, 100%:



Nie dość, że dużo więcej sukcesów, to zdarzały się one praktycznie co ten sam interwał (dlatego wykres jest zbliżony do prostej).

  - Zamknięta mapa okazała się jeszcze ciekawsza – ze wszystkich testów na 3 mapie, jako jedynemu udało się znaleźć prawidłową drogę, osiągając 9% sukcesów na

treningu, i 53% na testach. I udało mu się tego dokonać mimo tego, że często musiał wykonywać ruchy sprzeczne z założeniami (musiał się oddalać od mety).