Q-Learning i Deep Q-Learning

Tomasz Koźmiński (188682), Damian Łączyński (188652), Szymon Lider (188850)

1. Wstęp

Tematem naszego projektu jest porównanie niektórych metod używanych do nauczania ze wzmocnieniem. Jest to sposób uczenia maszynowego, w którym model zdobywa wiedzę na podstawie interakcji ze środowiskiem, nie korzystając z wcześniej przygotowanych danych. Wyróżnia się w nim kilka kluczowych elementów – środowisko, z którym model wchodzi w interakcję (może to być na przykład gra komputerowa), agenta, za pomocą którego model wchodzi w interakcję ze środowiskiem, stan, czyli jednoznaczny wektor cech pozwalający rozpoznać stan środowiska (przykładowo widok gry), pewna pula możliwych do wykonania przez agenta akcji, które zmieniają stan środowiska, oraz nagrodę, czyli zwracaną po każdej akcji wartość informującą agenta, jak korzystny był wykonany krok.

2. Q-Learning

W Q-Learningu agent tworzy i aktualizuje tabelę wartości Q, która reprezentuje ocenę oczekiwanej nagrody za wykonanie konkretnej akcji w określonym stanie. Poprzez iteracyjne eksplorowanie (losowy wybór akcji) i eksploatację (wybór najlepszej akcji na podstawie poznanych wartości Q) środowiska, agent aktualizuje wartości Q według zdobywanych nagród i stara się maksymalizować sumaryczne oczekiwane nagrody. Q-Learning jest oparty na równaniu Bellmana, które pozwala na propagację wartości oczekiwanych nagród z przyszłych stanów do obecnych, umożliwiając agentowi podejmowanie lepszych decyzji w celu osiągnięcia optymalnej strategii działania w dynamicznym środowisku.

3. DQN

Deep Q-Learning (DQN) to zaawansowana wersja Q-Learningu, która wykorzystuje głębokie sieci neuronowe do aproksymacji funkcji wartości Q. W tradycyjnym Q-Learningu tabela wartości Q rośnie wraz z ilością stanów i akcji, co w bardziej skomplikowanych środowiskach prowadzi do bardzo szybkiego wzrostu wymaganej pamięci. W DQN wykorzystuje się sieć neuronową do estymacji funkcji wartości Q, co pozwala na elastyczną reprezentację wartości Q dla różnych stanów i akcji. Proces uczenia polega na gromadzeniu doświadczeń agenta w pamięci, a następnie losowym pobieraniu próbek z tej pamięci w celu aktualizacji wag sieci.

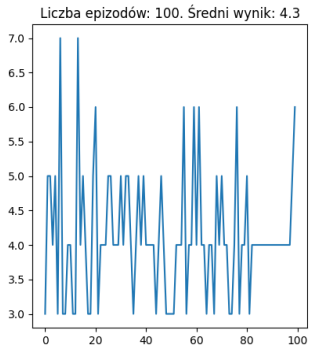
Należy także uwzględnić, że oba algorytmy podczas procesu nauki używają epsilon w funkcji wartości, aby dostosować wykorzystanie eksploracji i eksploatacji. Kiedy nasz model nie ma jeszcze wystarczająco dużo informacji powinien skupić się na eksploracji, czyli tzw. losowy ruch, a gdy już zacznie się dostosowywać do zadania - eksploatację, czyli wykorzystywać zdobytą już wiedzę.

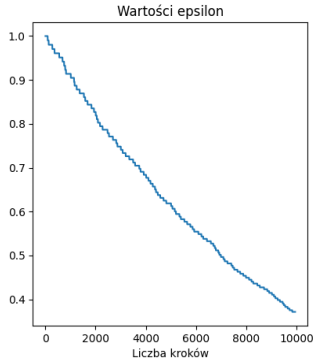
4. Opis gry

Modele trenowaliśmy na własnej implementacji gry *Snake*. W tej grze stanem jest pozycja owocu oraz wszystkich segmentów gracza na planszy, a możliwe akcje to wykonanie ruchu w jednym z czterech kierunków. Stan gry reprezentowany jest przez macierz na której znajdują się wszystkie elementy świata gry. Cały wąż jaki i owoc reprezentowane są przez jedną wartość, a więc algorytm nie potrafi rozróżnić swojej głowy od ciała oraz owocu co może skutkować jego błędną interpretacją stanu gry. Rozwiązanie takie zostało przyjęte ze względu na duże zapotrzebowanie pamięci w algorytmie Q-Learning. Agent otrzymuje nagrodę za zbieranie kolejnych owoców. Ostateczny wynik to liczba zebranych owoców. Gra kończy się w momencie śmierci gracza.

5. Wyniki i obserwacje: Q-Learning

Model dla Q-Learningu był trenowany na planszy 5×5 z uwagi na wykładniczo rosnące wymagania pamięciowe przy większych wymiarach. Przykładowy trening osiągnął następujące rezultaty.





Średni wynik nie jest stały i może być nieco inny przy różnych uruchomieniach programu, jednak mieści się on w przedziale 3,5 – 4,5. Można zaobserwować, że w końcowych epizodach, czyli kiedy zgodnie z wartościami epsilon model więcej eksploatował niż eksplorował – a więc korzystał z posiadanej już wiedzy – osiągał on konsekwentnie wyniki równe 4 lub więcej.

Rozmiar używanej tabeli Q można wyznaczyć z zależności:

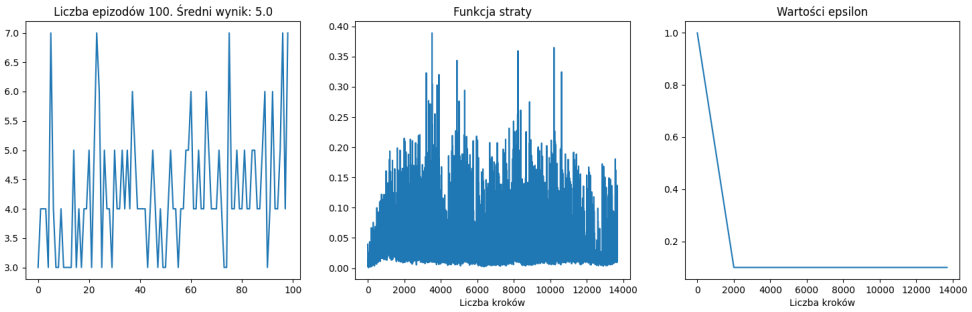
$$T = a \cdot s^{(X \cdot Y)}$$

a – liczba akcji, s – liczba stanów jednego pola, X, Y – wymiary planszy

Mamy więc do czynienia z funkcją wykładniczą, przez co wymagane miejsce rośnie bardzo szybko. Wykorzystanie algorytmu dla większych planszy okazuje się bardzo problematyczne.

6. Wyniki i obserwacje: Deep Q-Learning

Takie wyniki osiągnął DQN dla takiej samej planszy.



Średni wynik tym razem osiąga wartości z przedziału 4,5 – 5,2, a więc statystycznie nieco wyższe. Dziwić może fakt, że funkcja straty nie zmniejsza się, tak jak powinno mieć to miejsce w uczeniu z nadzorem. Dzieje się tak, ponieważ proces uczenia modyfikuje etykiety.

Główną przewagą algorytmu jest możliwość wykorzystania go do większych plansz bez nakładów pamięciowych Q-Learningu.

7. Wnioski

Wyniki DQN mogą być znacznie lepsze niż te osiągane przez tradycyjny Q-Learning w trudniejszych zadaniach. DQN może nauczyć się bardziej optymalnej strategii, szczególnie w przypadku złożonych środowisk. DQN jest również bardziej elastyczny w radzeniu sobie z dynamicznymi i zmieniającymi się środowiskami, ponieważ sieć neuronowa może dostosowywać się do nowych warunków.

Warto jednak pamiętać, że DQN może wymagać większego nakładu obliczeniowego oraz być bardziej podatny na zjawiska takie jak zapamiętywanie i przekazywanie błędnych informacji, co może prowadzić do niestabilnego uczenia się. W praktyce dobór metody powinien zależeć od problemu oraz dostępnych zasobów.

Użyte źródła:

- *https://github.com/Curt-Park/rainbow-is-all-you-need.git*
- *https://towardsdatascience.com/q-learning-algorithm-from-explanation-to-implementation-cdbeda2ea187*
- *https://datascience.eu/pl/uczenie-maszynowe/q-learning/*
- *https://www.youtube.com/watch?v=aCEvtRtNO-M&ab_channel=SirajRaval*