Algorytmy sztucznej inteligencji w Przemyśle 4.0

Maciej Siwicki Paweł Pagacz

PN 13:15

Spis treści

1 Wstęp

Algorytmy sztucznej inteligencji odgrywają znaczącą rolę w Przemyśle 4.0, który charakteryzuje się automatyzacją, cyfryzacją i interakcją między ludźmi, maszynami i systemami.

2 Cel projektu

Celem projektu jest zapoznanie się z metodami uczenia ze wzmocnieniem wraz z implementacją środowiska oraz dwóch wybranych algorytmów uczenia.

2.1 Zakres projektu

W skład projektu wchodzi:

1. Przedstawienie metody uczenia ze wzmocnieniem.
2. Zaimplementowanie środowiska do nauki - Pac-Man.
3. Zaimplementowanie pierwszej metody sztucznej inteligencji.
4. Zaimplementowanie drugiej metody sztucznej inteligencji.
5. Przeprowadzenie badań.

3 Część teoretyczna

3.1 Uczenie ze wzmocnieniem

Uczenie maszynowe to obszar sztucznej inteligencji poświęcony algorytmom, które poprawiają się automatycznie poprzez doświadczenie, czyli ekspozycję na dane. Dziedzinę możemy podzielić na trzy grupy:

1. uczenie nadzorowane,
2. uczenie nienadzorowane,
3. uczenie ze wzmocnieniem.

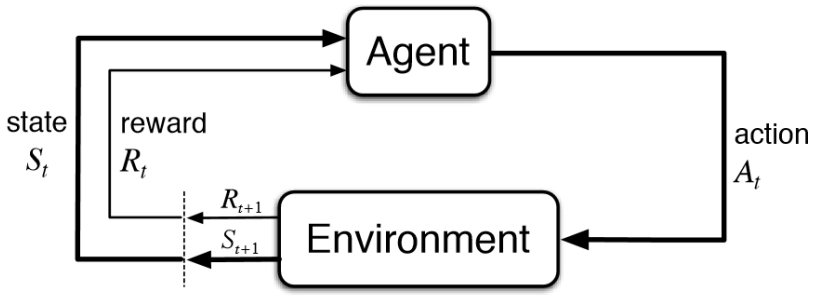
Uczenie ze wzmocnieniem to paradygmat uczenia maszynowego, w którym agent (np. program komputerowy lub robot) uczony jest podejmować decyzje w celu osiągnięcia określonych celów w dynamicznym środowisku. Cechami charakterystycznymi dla uczenia maszynowego jest:

1. brak konieczności posiadania zbioru danych,
2. nagroda może być odłożona w czasie,

3. czas ma znaczenie,
4. działanie agenta ma wpływ na dane, jakie otrzymuje ze środowiska.

Podstawowymi elementami uczenia ze wzmocnieniem jest:

1. Środowisko - to zadanie/symulacja, z którym agent wchodzi w interakcję.
2. Agent - poprzez interakcję ze środowiskiem uczy się, jak najkorzystniejszego oddziaływania ze środowiskiem. Funkcję odpowiedzialną za wybranie odpowiedniej akcji przez agenta nazywamy polityką.
3. Nagroda - wartość zwracana przez środowisko po wykonaniu akcji wybranej przez agenta.
4. Akcja - działanie agenta na środowisko. Akcja może być wybrana ze zbioru lub określoną wartością albo wektorem wartości.
5. Stan - zmienna określająca stan środowiska.



Rysunek 1: Schemat uczenia ze wzmocnieniem.

Agent wchodzi w interakcję ze środowiskiem poprzez podjęcie akcji A_t, ta natomiast prowadzi do zmiany stanu środowiska na S_t oraz otrzymaniem nagrody lub kary R_t przez agenta za podjętą akcję – proces ten następnie powtarza się w kolejnych iteracjach.
Stany muszą zawierać wszystkie zmienne niezbędne do uniezależnienia ich od wszystkich innych stanów. Prawdopodobieństwo następnego stanu (S_{t+1}), biorąc pod uwagę bieżący stan (S_t) i akcję (A_t), jest niezależne od historii. Ta pozbawiona pamięci właściwość MDP jest znana jako właściwość Markowa (1) [2]:

$$P(S_{t+1} \mid S_t, A_t) = P(S_{t+1} \mid S_t, A_t, S_{t-1}, A_{t-1}, \ldots) \tag{1}$$

3.2 Środowisko - gra Pac-Man

Gra Pac-Man to klasyczna gra arkadowa, która została stworzona w 1980 roku. Gracz kontroluje postać o nazwie Pac-Man, która ma za zadanie zebrać wszystkie kropki na planszy, unikając kontaktu z duchami. Główne rzeczy, które gracz może robić w grze Pac-Man, to:

1. Poruszanie się: gracz może kontrolować Pac-Mana, przemieszczając go w górę, w dół, w lewo i w prawo po labiryncie, w którym znajdują się kropki do zebrania.
2. Zbieranie kropek: głównym celem gry jest zebranie wszystkich kropek na planszy. Za każdą zebraną kropkę gracz zdobywa punkty.
3. Unikanie duchów: na planszy znajdują się duchy, które ścigają Pac-Mana. Gracz musi unikać kontaktu z nimi, aby nie stracić życia.
4. Zbieranie owoców: okresowo na planszy pojawiają się owoce, które gracz może zebrać, zdobywając dodatkowe punkty.

Środowisko zostało zaimplementowane zgodnie z paradygmatami uczenia ze wzmocnieniem.

3.3 Metoda DQN

Metoda DQN (Deep Q-Network) jest techniką stosowaną w uczeniu ze wzmocnieniem do nauki funkcji wartości stanu-akcji (Q-funkcji) za pomocą głębokich sieci neuronowych. DQN ma na celu naukę Q-funkcji, która przewiduje wartość oczekiwaną nagrody, jaką agent może uzyskać wykonując daną akcję w danym stanie. DQN wykorzystuje głęboką sieć neuronową do przybliżania Q-funkcji. Sieć ta przyjmuje stan środowiska jako wejście i zwraca oszacowaną wartość Q dla każdej dostępnej akcji (2).

$$Q(s_{t+1}|s_t, a) = r(s_t, a) + \mathbb{E}\left(\sum_{k=1}^{\infty} \gamma^k r(s_{t+k}, a_{t+k})\right) \tag{2}$$

Ta funkcja definiuje wartość Q dla konkretnego stanu s_{t+1}. Wybieramy naszą akcję w stanie s_t w taki sposób, aby maksymalizować wartość Q w następnym stanie s_{t+1}. Zdefiniujmy zatem optymalną politykę (3):

$$\hat{\pi}(s_t) = a_t = \arg\max_{a \in A(s_t)} Q(s_{t+1}|s_t, a) \tag{3}$$

co wymaga oszacowania optymalnej wartości Q. Jednakże dokładne oszacowanie Q jest wymagające obliczeniowo, stąd pomocne okazuje się zastosowanie sieci neuronowej.

3.4 Metoda A2C

Metoda A2C (Advantage Actor-Critic) to algorytm w uczeniu ze wzmocnieniem, który łączy w sobie dwie główne techniki: Actor-Critic i Advantage Learning. Aktor reprezentuje politykę decyzyjną w problemie uczenia ze wzmocnieniem. Polityka ta jest odpowiedzialna za wybór akcji w danym stanie. W praktyce, aktor jest implementowany jako sieć neuronowa, która przewiduje prawdopodobieństwa wyboru różnych akcji w danym stanie. Krytyk reprezentuje funkcję wartości stanów i ocenia, jak dobre jest obecne stanowisko

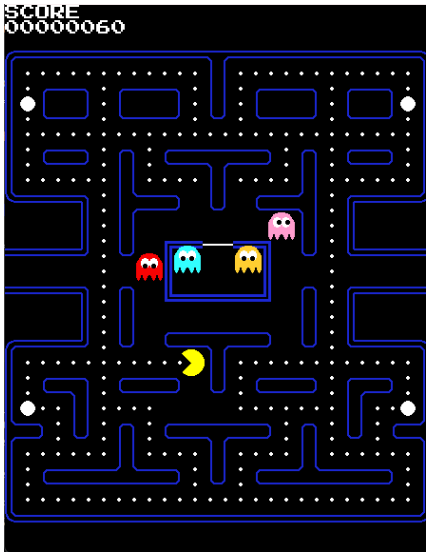
agenta w środowisku. Krytyk pomaga aktorowi poprawiać jego decyzje, dostarczając informacji zwrotnej na temat tego, jakie akcje prowadzą do korzystnych wyników.
W metodzie A2C wykorzystuje się pojęcie zalety, które reprezentuje różnicę między spodziewanym wynikiem działania w danym stanie, a średnią wartością stanów (ocenioną przez krytyka). Zalety pozwalają określić, czy dana akcja była lepsza lub gorsza od przeciętnego stanu.

4 Badania

Do zaimplementowanego ręcznie środowiska Pac-Man (Rysunek 2) zaimplementowano dwie metody:

1. Metoda DQN, (rozdział 4.1. Metoda DQN) składającą się z trzech warstw gęstych: dwóch warstw ukrytych z funkcją aktywacji ReLU oraz warstwy wyjściowej z funkcją liniową,

2. Metoda A2C, dostępna w bibliotece *stable_ baselines* (rozdział 4.2. Metoda A2C).



Rysunek 2: Zrzut z ekranu z zaimplementowanego środowiska.

Zakładając, że nasze zadanie uczenia się przez wzmacnianie przebiega według skończonego procesu decyzyjnego Markowa (MDP) (1), zdefiniować można:

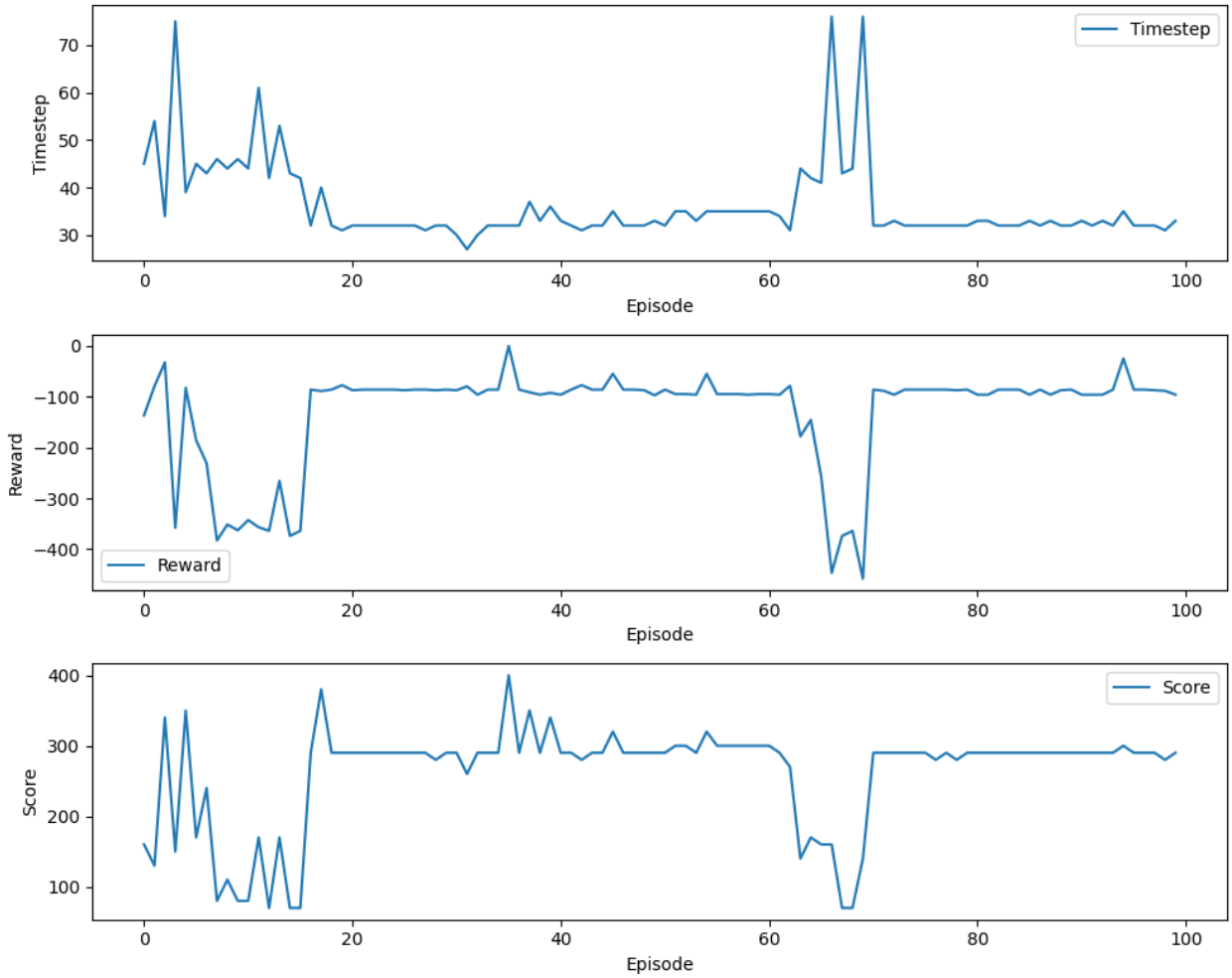
- Przestrzeń stanów S: pozycje (X, Y), kierunek ruchu wszystkich duchów (blinky, pinky, inky, clyde) oraz Pac-Mana. Dodatkowo, aby obserwować skuteczność zjadania pożywienia przez Pac-Mana, do przestrzeni stanów dodana jest również informacja o ilości zjedzonego pożywienia,

- Przestrzeń akcji A: Wszystkie możliwe ruchy Pac-Mana (ruch w prawo, lewo, w górę, w dół),

- Funkcja nagrody: Na podstawie testów środowiska dla zdefiniowanej przestrzeni stanów, ustalono następującą nagrodę:
  - Nagroda za jedzenie pożywienia +1,2;
  - Nagroda za zwycięstwo +150;
  - Kara za przegraną -20;
  - Kara za wykonywanie niedozwolonych poprzez architekturę mapy akcji -10;

4.1 Metoda DQN

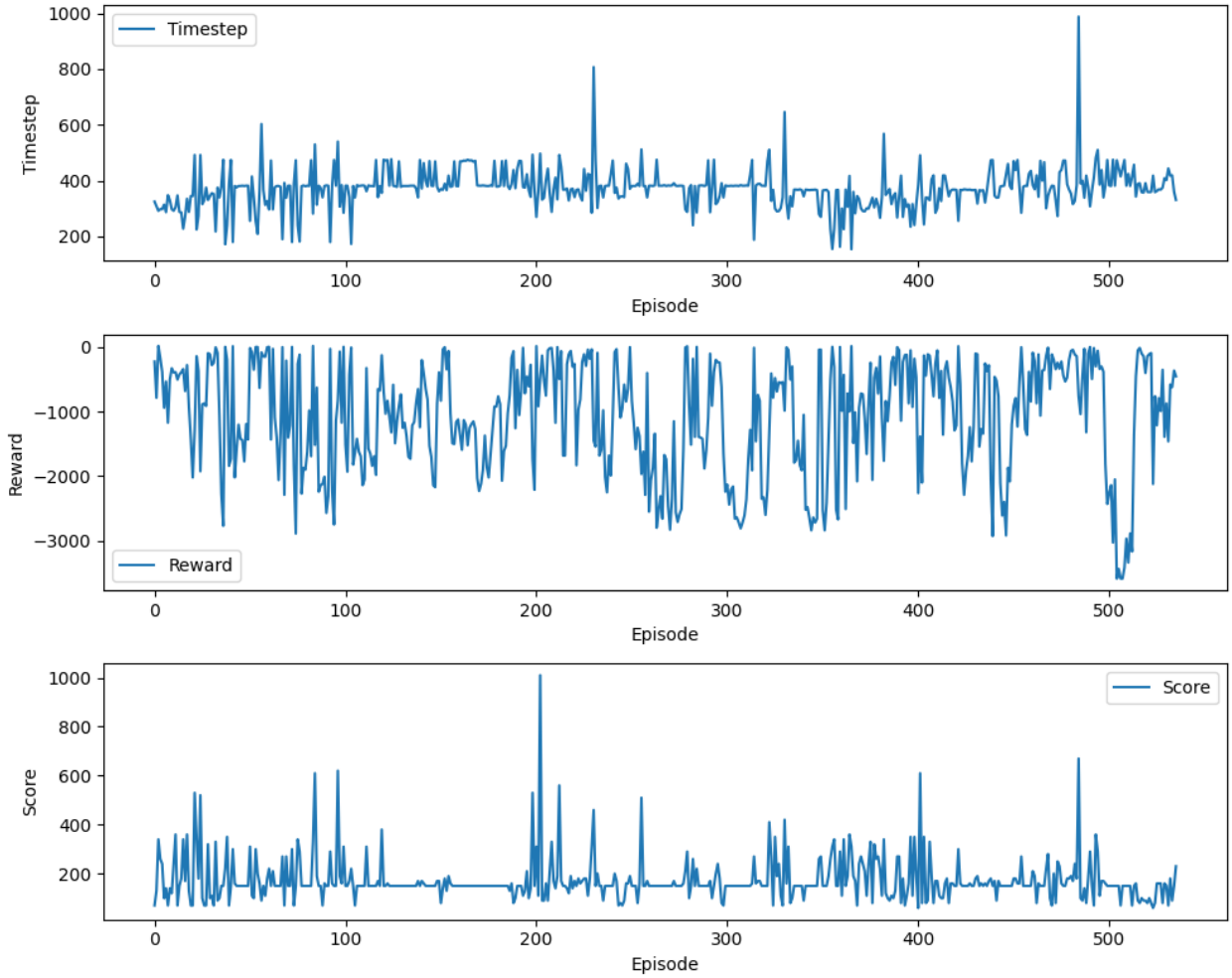
W celu przebadania metody DQN, podczas uczenia modyfikowano wartość epsilon decay, zwiększając tym samym czas uczenia. Co każdy epizod (czyli co każdy restart środowiska) zbierano informację na temat ilości kroków czasowych, wartości nagrody oraz ilość punktów zebranych podczas gry. Przykładowy przebieg wykresów zaprezentowano poniżej (Rysunek 3):



Rysunek 3: Zależność kroków czasowych, nagrody oraz wyniku gry od epizodu

4.2 Metoda A2C

Metoda A2C przebadana została w podobny sposób co metoda DQN (podczas procesu uczenia), jednakże ze względu na architekturę gotowej funkcji dostępnej w bibliotece *stable_ baselines*, parametrem zmiennym była całkowita ilość kroków czasowych. Modyfikacja tej wartości zmieniała ilość epizodów gry, a tym samym długość nauki. Przykładowy przebieg wykresów dla metody A2C przedstawiono poniżej (Rysunek 4):



Rysunek 4: Zależność kroków czasowych, nagrody oraz wyniku gry od epizodu

5 Wnioski

1. Z badań wynika, że algorytmy takie jak Deep Q Network (DQN) czy Advantage Actor-Critic (A2C), mają ograniczony potencjał do efektywnego nauczania agenta gry w skomplikowanym środowisku. Wynika to z wielkości przestrzeni stanów do przeanalizowania co zwiększa czas nauki,

2. Przy tworzeniu modelu uczenia ze wzmocnieniem kluczowymi aspektami okazały się implementacja zmiennych stanu oraz zaprojektowanie odpowiedniej funkcji nagrody. Dobór odpowiednich parametrów pozwoliłby na efektywniejszą naukę agenta,

3. W celu przyspieszenia nauki warto byłoby zastanowić się nad zmniejszeniem przestrzeni stanów poprzez ograniczenie pola widzenia agenta lub zmniejszenie mapy i zmiennych.

4. Na wykresach przedstawionych na rysunku 3 (metoda DQN) można zauważyć, że agentowi udało się znaleźć optymalną (dla tak zdefiniowanego czasu uczenia) ścieżkę, zapewniającą mu stały przyrost nagrody. Analizując wynik, można zauważyć, że udało mu się zjeść około 30 pelletów, co równoważne jest z 20% gry. Niestety chwilę później ginął, co świadczy o tym, że nauka trwała zbyt krótko.

Literatura

[1] Jing An, Jordi Feliu, and Abeynaya Gnanasekaran. Reinforcement learning in pacman. 2017.

[2] Miguel Morales. *Deep Reinforcement Learning*. Manning Publications Co., 20 Baldwin Road, Shelter Island, NY 11964, (2020).