**Inteligentne Techniki Obliczeniowe**

Raport wstępny

**Piotr Walas**

Projekt U4. Uczenie ze wzmacnianiem

Wykonać program przeznaczony do uczenia się agenta strategii poruszania się w świecie zadanym jako 2-wymiarowa macierz komórek. Wyrazić problem uczenia się strategii wyboru akcji jako uczenie ze wzmacnianiem.

Świat ma rozmiar N x N (np. N = 8, 10, 12). W losowych miejscach umieszczone są ściany; których położenie w trakcie jednego uczenia nie zmienia się. Ściana nie dotyka innej ściany i może tylko w jednym punkcie dotykać brzegu. Na brzegach świata znajduje się M wyjść (np. M=1, 3, 5).

Uczenie składa się z serii prób. Próba rozpoczyna się od umieszczenia agenta w losowo wybranym miejscu klatki i kończy się po m × N krokach (np. m=5, 7, 9) albo gdy agent dotrze do jednego z wyjść. Agent ma do dyspozycji 4 akcje: Wschód, Zachód, Północ, Południe; jest ujemnie nagradzany za każdą chwilę pobytu w klatce. Celem nauki jest maksymalizacja (zdyskontowanej) sumy nagród, których agent może oczekiwać w każdej chwili, czyli za jak najszybsze docieranie do wyjścia.

Wykonać prosty interfejs graficzny, który w trakcie nauki ma podawać średnią nagrodę z prób w każdym miejscu, natomiast po nauce (faza aktywnego działania agenta) ma pokazywać zachowanie agenta.

**Założenia projektu**

Projekt zakłada realizację programu w którym agent porusza się w dwuwymiarowym świecie (macierzy 2-wymiarowej komórek). Jego celem jest dotarcie do wyjścia. Świat posiada rozmiar oraz M wyjść. W świecie znajdować będą się ściany ograniczające ruch w taki sposób, by ściana nie dotykała innej ściany oraz tylko w jednym punkcie dotykała brzegu. Parametry N oraz M będzie można zdefiniować w programie. Następnie wygenerowany zostanie świat o żądanych rozmiarach z losowo rozmieszczonymi wyjściami oraz losowo rozmieszczonymi ścianami. W trakcie uczenia agenta świat pozostanie niezmienny (statyczny).

Agent uczy się poprzez wykonanie serii prób dotarcia do wyjścia. Z każdą kolejną iteracją agent z założenia coraz lepiej (w znaczeniu ilości wykonanych ruchów) będzie podążał do wyjścia. W każdej iteracji agent zostanie umieszczony w losowej klatce świata i wykona ruchów (m będzie można zdefiniować w programie przed rozpoczęciem nauki agenta). Iteracja może się również zakończyć gdy agent dotrze do wyjścia. Agent posiada możliwość wykonywania 4 akcji, ruchu w kierunku: Wschód, Zachód, Północ, Południe. Każda chwila (akcja nie powodująca przejścia bezpośrednio do wyjścia) oznacza karę – innymi słowy, agent powinien dotrzeć do wyjścia w możliwie najkrótszej liczbie ruchów.

Celem nauki jest maksymalizacja sumy nagród, których agent może oczekiwać w danej chwili czasu.

Po fazie nauki będzie możliwość ręcznego ustawienia agenta w wybranej komórce świata i uruchomienie symulacji – animacji poruszania się agenta do wyjścia.

Do zrealizowania projektu wykonana zostanie aplikacja graficzna przedstawiająca świat (wyświetlone zostaną średnie nagrody z prób w każdej komórce świata w trakcie nauki), agenta oraz ukazująca ruch agenta w trakcie symulacji (po fazie nauki).

**Podstawy teoretyczne**

**Proces decyzyjny Markowa**

Agent w danej chwili czasu znajduję się w określonym stanie i podejmuję decyzję co do dalszych akcji. W omawianym projekcie zakładamy dyskretność czasu (agent porusza się w dwuwymiarowym świecie a każda akcji powoduję wykonanie dyskretnej akcji). Stany oznaczona są jako , a decyzje agenta poprzez . Każda wykonana akcja przenosi agenta do kolejnego stanu . Przejście do kolejnego stanu oznacza przyznanie agentowi nagrody (dodatniej lub ujemnej) . Zadaniem agenta jest nauka podejmowania decyzji prowadzących do możliwie najwyższych nagród w przyszłości.

Formalny opis problemu definiuję się jako Proces Decyzyjny Markowa:

* Czas
* Przestrzeń stanów
* Przestrzeń decyzji
* Rozkład prawdopodobieństwa przejścia stanów według którego losowane są następne stany
* Funkcja określająca nagrodę w chwili na podstawie decyzji i następnego stanu ,
* Zbiór stanów terminalnych epizodu (który jest pusty, jeśli proces decyzyjny nie jest podzielony na epizody).
* Rozkład prawdopodobieństwa stanów początkowych epizodu

Epizody to ciąg decyzji prowadzących do celu, epizod kończy się gdy agent poniesie porażkę, dotrze do celu lub trafi do stanu terminalnego. W takiej sytuacji następuje przeniesienia agenta do losowego stanu i rozpoczęcie kolejnego epizodu.

Zazwyczaj przyjmuję się, że agent posiada niewielką wiedze na temat środowiska, wie jedynie o przestrzeni stanów oraz decyzji.

**Polityka decyzyjna**

Mechanizm podejmowania decyzji opiera się na wykorzystaniu pojęcia polityki decyzyjnej. Polityka przypisuję stanom rozkład prawdopodobieństwa decyzji. Wykorzystanie każdej z polityk oznacza różną zagregowaną nagrodę dla agenta, a celem uczenia jest wyłonienie najbardziej optymalnej polityki. Co ważne, dany stan nie posiada przypisanej tylko jednej decyzji, gdyż początkowo nie wiadomo, które decyzje okażą się najoptymalniejsze. Politykę decyzyjną, należącą do zbioru polityk oznacza się jako .

Celem agent można przedstawić jako:

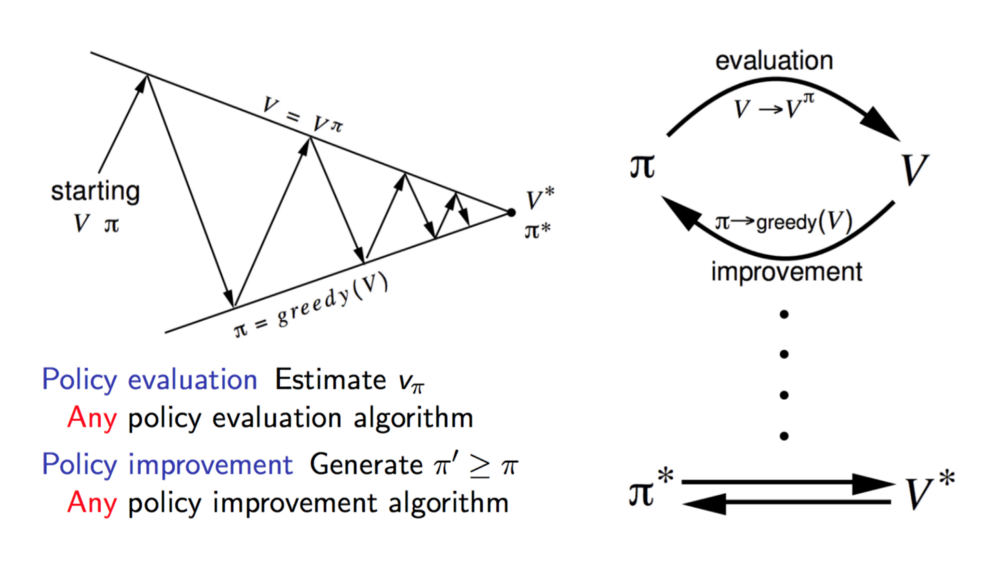
oznacza dyskonto. Funkcja przestawia wartość oczekiwaną sumy zdyskontowanych nagród jeżeli agent znajduje się w stanie i stosuje politykę . Parametr określa jaką wagę mają decyzję podejmowane w dłuższym okresie czasu, trudniej jest jednak wyznaczyć wtedy politykę, choć może być ona lepsza.

Poprzez pojęcie wyznaczenia optymalnej polityki rozumieć należy najczęściej iteracyjne indukowanie polityki nie gorszej od aktualnej, ostateczniej zbieżnej do optymalnej polityki decyzyjnej.

Ogólny algorytm iteracji polityki:

1. Określ dowolną politykę początkową
2. Wyznacz funkcję dla polityki .
3. Wyznacz politykę indukowaną przez i przypisz ją do .
4. Jeśli w ostatnim kroku polityka zmieniła się, wróć do 1.

Poniższy diagram ilustruję zależność pomiędzy ewaluacją polityki i jej poprawą w kolejnych iteracjach:



Źródło: https://towardsdatascience.com/introduction-to-various-reinforcement-learning-algorithms-i-q-learning-sarsa-dqn-ddpg-72a5e0cb6287

**Metody TD**

Metody TD polegają na wykorzystaniu obserwowalnych przejść stanów do przewidywania nieznanej końcowej wartości. W przypadku wartościowania strategii wartością przewidywaną jest zdyskontowana suma nagród.

**Q-Learning**

Q-Learning to algorytm uczenia ze wzmocnieniem, którego celem jest wyznaczenie polityki decyzyjnej maksymalizującej zagregowane wartości oczekiwanych nagród dla agenta. Algorytm korzysta ze zdefiniowanej funkcji wartości-decyzji:

Jest to zdyskontowana suma nagród dla agenta znajdującego się w stanie , podejmującego decyzję zgodnie z polityką .

Jeżeli decyzje podejmowane są w taki sposób, by zmaksymalizować funkcję wartości-decyzji w danym stanie , to indukowana jest polityka nie gorsza od polityki . W ten sposób wraz z kolejnymi iteracjami generowane są polityki zmierzające ku optymalnej. Algorytm Q-Learning cechuje się silną zbieżnością do optymalnej polityki dla funkcji wartości-decyzji, lecz żeby została wyindukowana musiałoby nastąpić (teoretycznie) nieskończenie wiele iteracji dla każdej pary stan-decyzja. Oznacza to jednak, że żadna z decyzji dla danego stanu nie może zostać wykluczona. Aby zrealizować ten wymóg należy zastosować np. strategię wyboru decyzji (mówiąc ogólnie, gdyż każda decyzja, nawet pozornie zła, może prowadzić ostatecznie do lepszej polityki).

strategia podejmowania decyzji definiowana jest poprzez poniższy wzór:

T to parametr określający w jakim stopniu wybór decyzji jest zbliżony do wyboru zachłannego – im mniejszy tym częściej wybierana jest decyzja maksymalizująca nagrodę dla danego stanu.

Ostatecznie, algorytm **Q-Learning** przedstawia się następująco:

1. Zainicjalizuj Q;
2. Wylosuj na podstawie Q i .
3. Wykonaj , zarejestruj i .
4. Przypisz .
5. Przypisz i wróć do Punktu 1.

Wyjaśnienie linii 3 algorytmu:

– poprzednia wartość dla danego stanu i decyzji

– parametr szybkości uczenia

- 0 oznacza, że agent nie uczy się niczego (korzysta z wiedzy znanej a priori), a 1 oznacza, że brane są pod uwagę tylko najnowsze informacje

nagroda

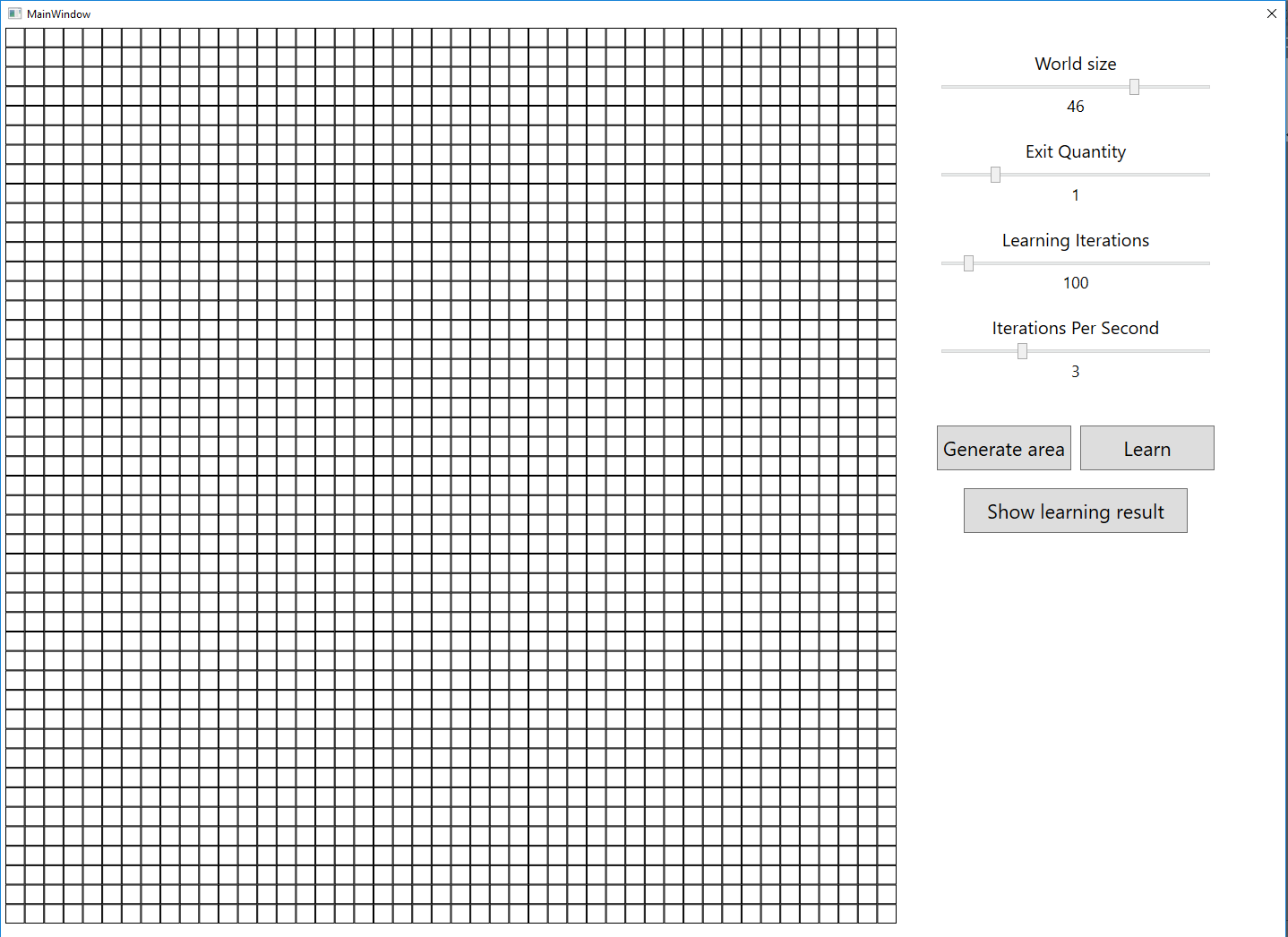
- determinuje poziom znaczenia przyszłych (późniejszych) nagród

estymacja przyszłej maksymalnej nagrody dla stanu

**Wykorzystanie algorytmu Q-Learning do rozwiązania zadanego problemu.**

Zgodnie z założeniami teoretycznymi w projekcie wykorzystany zostanie algorytm Q-Learning do wytrenowania agenta.

Wstępny (niekompletny) interfejs programu przedstawia się następująco:



Widoczna macierz reprezentuje świat, każda komórka przedstawiać będzie jeden z trzech możliwych obszarów: puste, ściana, wyjście (dodatkowo komórka przedstawiająca aktualne położenia agenta). Poszczególne obszary reprezentowane będą odpowiednimi kolorami. Każda wykonana akcja agenta będzie się wiązała z pewną nagrodą (niekoniecznie dodatnią): ruch na puste pole to -1 punkt, ruch „na ścianę” (agent „zderza się ze ścianą”, agent pozostaje w tej samej komórce świata co przed zderzeniem) kosztuje -10 punktów, a dotarcie do celu to +100 punktów. Są to wartości orientacyjne, ich ostateczna postać może być inna.

Zgodnie z Bolzmanowską strategią wyboru decyzji (spośród 4 możliwych dla agenta: ruch wschód, zachód, północ, południe) w trakcie nauki nie zawsze wybierana będzie akcja dająca najwyższą (w danej chwili) nagrodę. Dzięki temu w trakcie nauki eksplorowane będą również polityki mogące dać ostatecznie wyższą zdyskontowaną sumę nagród, lecz dające niższą (w danej chwili) nagrodę natychmiastową. W tym miejscu nadmienić trzeba, że zgodnie z przyjętymi założeniami, można domyśleć się, iż nie będzie to miało większego znaczenia (gdyż agentowi nigdy nie będzie się opłacał ruch „na ścianę”), lecz miałoby to istotne znaczenie gdybyśmy do programu wprowadzili przeszkody (zamiast omijać przeszkodę może warto ją przeskoczyć), bądź wprowadzili różnego rodzaju powierzchnie dla pola „pustego” (o karze za przejście różnej od -1).

Podczas inicjalizacji procesu nauki agenta wygenerowana zostanie losowa polityka decyzyjna, która wraz z każdą iteracją zbliżać się będzie do optymalnej polityki decyzyjnej. Początkowo agent często będzie wybierał ruch w stronę ściany, bądź wybierał nie optymalną (w znaczeniu dążenia do wyjścia) ścieżkę po pustych polach, lecz wraz z kolejnymi iteracjami polityki jego zachowanie ulegnie obiektywnej poprawie. Jest to oczywiście związane bezpośrednio z 3 linią przedstawionego algorytmu Q-Learning, w której nanoszone są poprawki na oczekiwaną nagrodę w funkcji stan-decyzja.

Zastosowanie algorytmu Q-Learning powoduje, iż agent nie musi posiadać wiedzy o środowisku.

**Struktura programu i zastosowane technologie**

Program zostanie wykonany zgodnie ze wzorcem projektowym MVC (Model-View-Controller). Wzorzec pozwala na oddzielenie modelu (reprezentacji świata i wszelkich danych wykorzystywanych do obliczeń) od ich graficznej reprezentacji (warstwy prezentacji). Pozwala to na niezależne, efektywne czasowo dokonywanie obliczeń i aktualizacji modelu świata. W przypadku omawianego programu zostanie zastosowana wariacja tego wzorca o nazwie MVVC (związana stricte z zastosowaną strukturą ramową).

Do realizacji projektu wykorzystany zostanie język C# wraz z silnikiem graficznym WPF (wspieranym biblioteką PRISM).

**Bibliografia**

1. Włodzimierz Kasprzak, Inteligentne Techniki Obliczeniowe, wykład 10 - Uczenie ze wzmacnianiem, Ośrodek Kształcenia na Odległość, Politechnika Warszawska, Warszawa 2019
2. Paweł Wawrzyński, Preskrypt wykładu dla studentów technik informacyjnych - Podstawy Sztucznej Inteligencji, Warszawa 2013
3. Introduction to Various Reinforcement Learning Algorithms. Part I (Q-Learning, SARSA, DQN, DDPG) - https://towardsdatascience.com/introduction-to-various-reinforcement-learning-algorithms-i-q-learning-sarsa-dqn-ddpg-72a5e0cb6287