**Raport**

**Wprowadzenie do Sztucznej Inteligencji – Ćwiczenie 6.**

*Uczenie ze wzmocnieniem*

Autor: Aleksandra Jamróz, nr albumu: 310 708

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Treść zadania – *Q-Uber***

Elon Piżmo konstruuje autonomiczne samochody do swojego najnowszego biznesu. Dysponujemy planszą NxN (domyślnie N=8) reprezentującą pole do testów jazdy. Na planszy jako przeszkody stoją jego bezpłatni stażyści. Mamy dwa autonomiczne samochody: Random-car, który kierunek wybiera rzucając kością (błądzi losowo po planszy) oraz Q-uber, który uczy się przechodzić ten labirynt (używa naszego algorytmu). Samochody zaczynają w tym samym zadanym punkcie planszy i wygrywają, jeśli dotrą do punktu końcowego, którym jest inny punkt planszy. Istnieje co najmniej jedna ścieżka do startu do końca. Elon oszczędzał na module do liczenia pierwiastków, dlatego samochody poruszają się przy użyciu metryki Manhattan (góra, dół, lewo, prawo). Jeżeli samochód natrafi na stażystę to kończy bieg i przegrywa. Analogicznie jak wejdzie na punkt końca to wygrywa i również nie kontynuuje dalej swojej trasy. Celem agenta jest minimalizacja pokonywanej trasy.

**Labirynt**

Plansza, po której poruszają się samochody, to lista zawierająca N list o długości N, przy czym N oznacza długość boku planszy. Wszystkie elementy inicjowane są wartością 0, oznaczającą pole, po którym samochody mogą się dowolnie poruszać. Następnie losowane są miejsca dla stażystów, które oznacza się poprzez zamianę zera na jedynkę.

Aby ułatwić przeprowadzanie testów, zaimplementowałam funkcje zapisujące i odczytujące labirynt do pliku tekstowego.

**Taksówka Elona**

Taksówka zawiera planszę do przejechania oraz współrzędne punktu początkowego i końcowego, ustawione domyślnie na przeciwległe jej krańce. Oprócz tego zawiera pole określające, czy taksówka zakończyła już swój bieg – dzieje się to w przypadku uderzenia w stażystę lub dojechania do punktu końcowego. Główna metoda tej klasy polega na wykonaniu ruchu. Mamy do wyboru 4 akcje – ruch w dół – 0, w górę – 1, w lewo – 2 oraz w prawo – 3. Podczas ruchu zmienia się jedna z aktualnych współrzędnych taksówki. Następnie sprawdzana jest poprawność ruchu i na tej podstawie zwracana jest nagroda. Jeżeli taksówka uderza w ścianę, taksówka nie kończy biegu, ale ustawiane są poprzednie współrzędne, a nagroda wynosi -10. Jeżeli uderza w stażystę – kończy jazdę a nagroda to również -10. Nagroda za dojechanie do miejsca docelowego wynosi 20, a za bezwypadkowy ruch: -1. Nagroda wynosząca -1 za teorytycznie poprawny ruch wynika z minimalizacji liczby wszystkich ruchów na planszy.

**Q-learning**

Liczba stanów taksówki jest równa liczbie pól na planszy. Dla wszystkich możliwych kombinacji współrzędnych numer stanu liczy się poprzez pomnożenie wartości współrzędnej x przez liczbę rzędów i dodanie do otrzymanej wartości współrzędnej y. W ten sposób każdy stan otrzymuje unikalny numer. Główny zamysł uczenia ze wzmocnieniem ma miejsce w metodzie *drive*. Odpowiada ona za wykonanie całej trasy taksówki od punktu początkowego do wydarzenia kończącego jazdę. Jazda odbywa się wedle podanych parametrów: epsilon, beta i gamma oraz podanej tablicy q-learningu. Tablicę podaję jako parametr, ponieważ do wykształcenia poprawnej tabeli potrzeba wielu taksówek. Metoda zwraca listę wykonanych po kolei akcji oraz zmodyfikowaną tablicę. Epsilon wyznacza prawdopodobieństwo wykonania ruchu losowego lub ruchu zgodnego z odczytem tabeli. Jeżeli wlosowana wartość z przedziału (0,1) jest od niego mniejsza, numer akcji jest przypadkowy. W innym przypadku wybiera maksymalną wartość z odpowiedniego rzędu w tabeli. Następnie wykonywany jest ruch oraz wyznaczana jest nagroda. Za pomocą funkcji *update\_q\_table* modyfikuję pole poprzedniego stanu i wykonanej akcji w tabeli. Wzór tej operacji wygląda następująco:

*q\_table[state, action] += beta \* (reward + gamma \* next\_state\_max - q\_table[state, action])*

**Random taxi**

Samochód poruszający się po labiryncie w losowy sposób ma nikłe szanse na jego przejście, zwłaszcza za każdym razem zaczynając od zera, to znaczy nie posiadając żadnej wiedzy na temat labiryntu. Aby przejść labirynt, którego używałam do testów taksówki Elona, musiałby wykonać bezbłędnie ponad 15 ruchów pod rząd. Prawdopodobieństwo takiego zdarzenia wynosi ok. 9.31e-10, czyli jest mniejsze niż szansa na trafienie szóstki na loterii lotto.

**Wizualizacja**

Wykorzystałam bibliotekę pygame do wizualizacji trasy pokonywanej przez samochód. Żeby włączyć odpowiednią trasę, należy ręcznie wkleić uzyskaną liczbę ruchów lub zastosować listę zwróconą bezpośrednio przez funkcję *drive* taksówki, po czym uruchomić program.

**Obraz zawierający tekst, urządzenie, pomiar

Opis wygenerowany automatycznie**

Rysunek 3. Przykład macierzy pomyłek dla klasyfikacji 11000:1950 elementów

Obraz zawierający tekst, urządzenie, pomiar

Opis wygenerowany automatycznie

Rysunek 4. Macierz pomyłek dla klasyfikacji dla pomieszanego zbioru w proporcji 0.75:0.25

**Testowanie algorytmu**

Zdecydowałam się przeprowadzić 2 rodzaje testów. Pierwszy polegał na analizie różnic między wynikami klasyfikacji wykorzystującej walidację krzyżową o różnych wartościach parametru. Drugi natomiast na analizie różnic w wynikach klasyfikowania zbioru o różnych proporcjach zbioru treningowego i testowego. Oba rodzaje testów przeprowadzałam na dwóch plikach   
z danymi: w jednym dane były domyślnie posortowane, w drugim – specjalnie pomieszane.

1. Walidacja krzyżowa

Intuicyjnym byłoby stwierdzenie, że im więcej elementów zawiera zbiór treningowy, tym dokładniejsze będą predykcje klasyfikatora. Ta zależność w przypadku danych posortowanych ma miejsce. Przy podziale zbioru na 5 elementów wzwyż, dokładność klasyfikacji zwiększa się. Ze schematu wyłamuje się pierwszy przypadek, w którym dzielimy zbiór na 3 części – wyniki w tej sytuacji są prawie równie dobre, jak przy 10 częściach. Może to być spowodowane tym, że posortowane dane podzieliły się na znaczące części, co zwiększa liczbę elementów nieznajdujących swoich argumentów w drzewie.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Dane** | **k** | **Recall** | **Fall\_out** | **Precision** | **Accuracy** | **F1\_score** |
| posortowane | 3 | 0,95 | 0,95 | 0,89 | 0,81 | 0,92 |
| posortowane | 5 | 0,83 | 0,83 | 0,83 | 0,69 | 0,77 |
| posortowane | 7 | 0,84 | 0,84 | 0,87 | 0,76 | 0,82 |
| posortowane | 10 | 0,96 | 0,96 | 0,88 | 0,80 | 0,92 |
| posortowane | 20 | 1,00 | 1,00 | 0,97 | 0,93 | 0,98 |
| Średnia wartość | 9 | 0,91 | 0,91 | 0,89 | 0,80 | 0,88 |
| Odchylenie standardowe | 6,67 | 0,08 | 0,08 | 0,05 | 0,09 | 0,08 |

Rysunek 5. Tabela metryk klasyfikacji z walidacją krzyżową dla posortowanego zbioru danych

Z kolei w przypadku danych przemieszanych, nie ma zasadniczej różnicy, na ile elementów podzielimy zbiór, wyniki wychodzą zawsze podobnie. Są jednocześnie znacznie lepsze niż te dla danych posortowanych. Wynika to z większej różnorodności danych w każdym zestawie. Pozwala to na uzyskanie większej kombinacji wartości, tym samym bardziej rozbudowane drzewo, a co za tym idzie – dokładniejszy model.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Dane** | **k** | **Recall** | **Fall \_out** | **Precision** | **Accuracy** | **F1\_score** |
| pomieszane | 3 | 1,00 | 1,00 | 0,98 | 0,95 | 0,99 |
| pomieszane | 5 | 0,98 | 0,98 | 0,98 | 0,96 | 0,98 |
| pomieszane | 7 | 0,99 | 0,99 | 0,98 | 0,96 | 0,99 |
| pomieszane | 10 | 0,99 | 0,99 | 0,98 | 0,96 | 0,99 |
| pomieszane | 20 | 1,00 | 1,00 | 0,98 | 0,96 | 0,99 |
| Średnia wartość | 9 | 0,99 | 0,99 | 0,98 | 0,96 | 0,99 |
| Odchylenie standardowe | 6,67 | 0,01 | 0,01 | 0,00 | 0,00 | 0,00 |

Rysunek 6. Tabela metryk klasyfikacji z walidacją krzyżową dla pomieszanego zbioru danych

1. Różne proporcje zbioru testowego

Testując różne proporcje wielkości zbioru trenującego i testującego chciałam sprawdzić, jaka jest różnica w prawdomówności klasyfikatora. Przeprowadziłam próby dla 3 proporcji: 6:4, 75:25 oraz 9:1. Dla każdej z nich włączyłam algorytm dwukrotnie. Podobnie jak podczas prób z walidacją krzyżową, gorsze wyniki we wszystkich przypadkach uzyskał model działający na posortowanych danych. Różnica w wynikach przy proporcji 6:4 a 9:1 jest zauważalna.   
W przypadku danych pomieszanych różnica jest znikoma bądź nie ma jej wcale.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Dane** | **Test\_size** | **Recall** | **Fall\_out** | **Precision** | **Accuracy** | **F1\_score** |
| posortowane | 0,1 | 1,00 | 1,00 | 0,98 | 0,96 | 0,99 |
| posortowane | 0,1 | 1,00 | 1,00 | 0,98 | 0,96 | 0,99 |
| posortowane | 0,25 | 0,93 | 0,93 | 0,98 | 0,96 | 0,96 |
| posortowane | 0,25 | 1,00 | 1,00 | 0,97 | 0,95 | 0,99 |
| posortowane | 0,4 | 0,98 | 0,98 | 0,98 | 0,95 | 0,98 |
| posortowane | 0,4 | 1,00 | 1,00 | 0,97 | 0,95 | 0,99 |
| Odchylenie standardowe | 0,134 | 0,027 | 0,027 | 0,003 | 0,004 | 0,013 |
| Średnia | 0,250 | 0,986 | 0,986 | 0,977 | 0,955 | 0,981 |

Rysunek 7. Tabela metryk dla klasyfikacji danych posortowanych dla różnych wartości proporcji

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Dane** | **Test\_size** | **Recall** | **Fall\_out** | **Precision** | **Accuracy** | **F1\_score** |
| pomieszane | 0,1 | 1,00 | 1,00 | 0,98 | 0,96 | 0,99 |
| pomieszane | 0,1 | 1,00 | 1,00 | 0,98 | 0,95 | 0,99 |
| pomieszane | 0,25 | 1,00 | 1,00 | 0,98 | 0,96 | 0,99 |
| pomieszane | 0,25 | 1,00 | 1,00 | 0,98 | 0,96 | 0,99 |
| pomieszane | 0,4 | 0,97 | 0,97 | 0,98 | 0,95 | 0,97 |
| pomieszane | 0,4 | 0,99 | 0,99 | 0,98 | 0,95 | 0,98 |
| Odchylenie standardowe | 0,134 | 0,011 | 0,011 | 0,001 | 0,003 | 0,006 |
| Średnia | 0,250 | 0,993 | 0,993 | 0,977 | 0,955 | 0,985 |

Rysunek 8. . Tabela metryk dla klasyfikacji danych pomieszanych dla różnych proporcji

**Budowa drzewa**

Drzewo decyzyjne budowane za pomocą algorytmu ID3 nie musi być symetryczne.   
W zależności od rozkładu danych na tej samej głębokości drzewa decyzja może zależeć od różnych kategorii. W przypadku zbioru danych o przedszkolakach, sekwencja kolejnych kategorii jest w większości przypadków stała. Główny nurt przebiega następująco:

Health 🡪 Has\_nurs 🡪 Parents 🡪 Social 🡪 Housing 🡪 Finance 🡪 Form 🡪 Children

**Uwagi**

Zaimplementowany przeze mnie algorytm jest algorytmem uniwersalnym. To znaczy, że nie ma żadnych zdefiniowanych na stałe zmiennych ani nazw powiązanych z zadanym zbiorem danych. Można uruchomić go z dowolnym plikiem csv i będzie działał poprawnie.