Wprowadzenie do sztucznej inteligencji - ćwiczenie nr 4

Jan Górski

4 maja 2024

Spis treści

1 Zadanie **3**

1.1 Zadanie 3

1.2 Wskazówki dotyczące ćwiczenia 3

2 Część teoretyczna **4**

2.1 4

2.2 Drzewo decyzyjne 4

2.3 Algorytm ID3: pseudokod 5

2.4 Miara zróżnicowania - entropia 5

3 Część praktyczna **6**

3.1 Subiektywna ocena działania algorytmu 6

3.2 Uwagi dotyczące implementacji algorytmu 6

3.3 Badanie wpływu głębokości drzewa przeszukiwań oraz rodzaju funkcji oceny stanu na liczbę wygranych algorytmu 7

4 Wnioski **10**

1 Zadanie

1.1 Zadanie

Zaimplementować klasyfikator ID3 (drzewo decyzyjne). Atrybuty nominalne, testy tożsamościowe. Podać dokładność i macierz pomyłek na zbiorach: **Breast cancer** i **mushroom**. Dlaczego na jednym zbiorze jest znacznie lepszy wynik niż na drugim? Do potwierdzenia lub odrzucenia postawionych hipotez konieczne może być przeprowadzenie dodatkowych eksperymentów ze zmodyfikowanymi zbiorami danych. Sformułować i spisać wnioski.

1.2 Wskazówki dotyczące ćwiczenia

- Atrybuty nominalne - każdy atrybut może przyjmować jedną z kilku dozwolonych wartości, zakładamy, że wartość atrybutu to napis, np. “kot”, “a”, “20-34”, “>40”.
- Testy tożsamościowe - jeżeli atrybut testowany w danym węźle ma np. 3 dozwolone wartości, np. a, b, c, to z węzła tego wychodzą 3 krawędzie oznaczone: a, b, c.
- Na tym ćwiczeniu klasyfikator trenuje się na zbiorze trenującym, a ocenia jego jakość na zbiorze testującym. Należy losowo podzielić zbiór danych na trenujący i testujący w stosunku 3:2.
- Jeżeli zbiór danych zawiera numery lub identyfikatory wierszy to należy je wyrzucić - nie chcemy uczyć się identyfikatorów wierszy.
- Brakujące wartości atrybutów taktujemy jako wartość, np. jeżeli symbol “?” oznacza brakującą wartość, a symbole “a”, “b” wartości normalne, to z naszego punktu widzenia mamy 3 wartości normalne (fachowo: 3 wartości atrybutu): “a”, “b”, “?”.
- Tak naprawdę to nie musimy rozumieć dziedziny problemu - na wejściu mamy napisy, na wyjściu napisy, nie ważne czy klasyfikujemy sekwencje DNA, grzyby, czy samochody.
- Nazwa pliku ze zbiorem danych jest parametrem algorytmu klasyfikacji, kod klasyfikatora powinien być w stanie obsłużyć inny zbiór danych o tym samym rozkładzie kolumn (czyli nie należy wpisywać wartości atrybutów “na sztywno” w kodzie).
- W repozytorium ze zbiorami danych zwykle w plikach “.names” jest napisane, który atrybut to klasa (czyli wartości której kolumny mamy się nauczyć przewidywać).

2 Część teoretyczna

2.1 Drzewo decyzyjne

Przedmiotem klasyfikacji drzewa decyzyjnego są obiekty w danym zbiorze U charakteryzowane przez pewien zestaw D atrybutów nominalnych. Każdy obiekt z U ma $|D|$ atrybutów, z których każdy ma wartość ze skończonego zbioru. Każdy obiekt w U jest pewnej klasy, przy czym zbiór wszystkich klas to Y.

Zadanie polega na zbudowaniu klasyfikatora, który na podstawie atrybutów będzie odgadywał klasy obiektów.

2.2 Algorytm ID3: pseudokod

Algorithm 1: Iterative Dichotomiser 3

Data: Y : zbiór klas,
D : zbiór atrybutów wejściowych,
$U \neq 0$: zbiór par uczących
Result: Drzewo decyzyjne

begin
  if $y_i == y \quad \forall \langle x_i, y_i \rangle \in U$ **then**
    return Liść zawierający klasę y.
  end if
  if $|D| == 0$ **then**
    return Liść zawierający najczęstszą klasę w U.
  end if
  $d = \arg\max_{d \in D} InfGain(d, U)$
  $U_j = \{ \langle x_i, y_i \rangle \in U : x_i[d] = d_j \}$, ***gdzie***
   d_j - j-ta wartość atrybutu d
  return Drzewo z korzeniem d oraz krawiędziami:
   $d_1, d_2, \ldots$ prowadzącymi do drzew:
   $ID3(Y, D - \{d\}, U_1), ID3(Y, D - \{d\}, U_2) \ldots$
end

2.3 Miara zróżnicowania - entropia

Kluczowym elementem algorytmu jest wybór atrybutu przypisanego do korzenia drzewa. Najlepiej byłoby wtedy, gdyby na podstawie atrybutu dało się podzielić zbiór U na podzbiory, takie że w każdym z nich występują wyłącznie obiekty innej klasy. Nie jest to zwykle możliwe, dlatego stosuje się kryterium zmierzające do stworzenia sytuacji zbliżonej, tj. jak największego zróżnicowania występowania poszczególnych klas w podzbiorach. Miarą tego zróżnicowania jest entropia:

$$I(U) = -\sum_i f_i \ln f_i,$$

gdzie f_i - częstość i-tej klasy.

Entropia zbioru podzielonego na podzbiory jest to średnia ważona entropii podzbiorów, a mianowicie

$$Inf(d,U) = \sum_j \frac{S_j||}{|S|} I(S_j),$$

gdzie $|S|$ to liczba elementów zbioru S, zaś S_j, $j = 1, 2, \ldots$ to zbiory powstałe przez podział zbioru S ze wzsględu na wartość atrybutu D.

Zdobycz informacyjna służąca do wyboru atrybutu d ma następującą definicję:

$$InfGain(d,U) = I(U) - Inf(d,U).$$

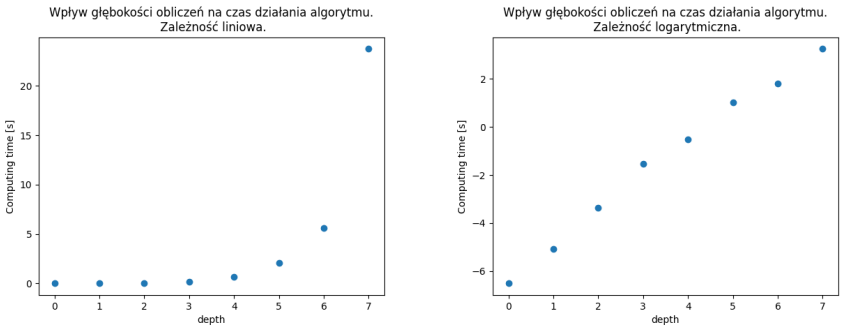
3 Część praktyczna

3.1 Subiektywna ocena działania algorytmu

Celem pierwszej części eksperymetu było zaimplementowanie algorytmu dla gry w warcaby i subiektywna ocena wybranych przez algorytm posunięć.

Algorytm działał zgodnie z przyjętą heurystyką: różnica sumy wartości własnych pionków z sumą wartości pionków przeciwnika. Dla głębokości równej 1 zachowanie komputera można nazwać zachłannym- Gdy tylko mógł wykonać bicie, to to robił. Jednak wielokrotnie samemu poruszał się na pole, z którego mógł zostać zbity bez możliwości odbicia.

Dla kontrastu, przy głębokości równej 6 jego gra była znacznie lepsza. Ruchy komputera wydawały się być rozsądne. Niezależnie jednak od tego, czy ruch był „naturalnym" odbiciem jako reakcja na bicie, czy też występował w trudniejszej do ecenienia pozycji, czas potrzebny na wykonanie ruchu był tak samo długi.



(a) Wykres liniowy.

(b) Wykres logarytmiczny.

Rysunek 1: Wykresy złożoności czasowej działania algorytmu w zależności od głębokości przeszukiwania.

3.2 Uwagi dotyczące implementacji algorytmu

- Kilkakrotnie podczas testów zaobserwowałem, że w pewnym momencie obaj gracze sterowani przez algorytm zaczynały powtarzać posunięcia, co uniemożliwiało określenie, który z nich wygrał. Dodatkowo, wywołując algorytm dla takich samych parametrów otrzymywałem taki sam przebieg partii. Żeby zmienić te rzeczy, mając listę par <ruch, ocena> przed posortowaniem ich ze względu na wartość oceny mieszałem je i dopiero wtedy sortowałem. Pozwalało to na wybór w tej samej sytuacji innych ścieżek, co zmniejszało monotonny charakter rozgrywki.
- Głębokość równa 0 została zaimplementowana jako wybór losowego ruchu spośród dostępnych.
- W przypadku heurystyki stosującej ocenę zwartości grupy lub bycie przy krawędziach planszy zastosowano następującą interpretację: „zwartość" była wyznaczana zarówno poziomo

jak i poziomo. Wyliczano ją jako odchylenie standardowe liczone dla zbioru położeń (wybranej składowej) wszystkich figur danego koloru. Następnie obie sumy (dla poziomu i pionu) sumowano. Zwracano różnicę oceny pomiędzy oceną białego a czarnego przeskalowaną o współczynnik. Współczynnik wybrano po obserwacji działania algorytmu dla różnych wartości. Obecność figury przy krawędzi miało wpływ na ocenę danego piona rzędu 0.1.

3.3 Badanie wpływu głębokości drzewa przeszukiwań oraz rodzaju funkcji oceny stanu na liczbę wygranych algorytmu

Celem drugiej części eksperymetu było zbadanie zachowania komputera dla różnych wartości głębokości drzewa przeszukiwań oraz dla różnych funkcji heurystycznych, które miały zostać zaimplementowane.

Wpływ głębokości drzewa przeszukiwań oraz rodzaju funkcji oceny stanu na liczbę wygranych algorytmu. Badany gracz niebieski.		
heurystyka (biały | niebieski)	głębokość (biały | niebieski)	wynik (z perspektywy białego) (wygrana | remis | przegrana)
(default | default)	(2 | 1)	(6 | 1 | 13)
	(2 | 2)	(0 | 0 | 20)
	(2 | 3)	(0 | 0 | 20)
	(2 | 4)	(1 | 0 | 19)
	(2 | 5)	(0 | 0 | 20)
(default | tight)	(2 | 1)	(13 | 6 | 1)
	(2 | 2)	(5 | 10 | 5)
	(2 | 3)	(4 | 12 | 4)
	(2 | 4)	(3 | 17 | 0)
	(2 | 5)	(7 | 9 | 4)
(default | enemy side)	(2 | 1)	(7 | 0 | 13)
	(2 | 2)	(0 | 0 | 20)
	(2 | 3)	(0 | 0 | 20)
	(2 | 4)	(0 | 0 | 20)
	(2 | 5)	(0 | 0 | 20)
(default | hight)	(2 | 1)	(9 | 1 | 10)
	(2 | 2)	(0 | 0 | 20)
	(2 | 3)	(0 | 0 | 20)
	(2 | 4)	(0 | 0 | 20)
	(2 | 5)	(0 | 0 | 20)

Tablica 1: Rezultaty pomiarów. Badany gracz niebieski. Parametry gracza białego: głębokość - 2, heurystyka - domyślna.

Dla domyślnej heurystyki gdy głębokość przeszukiwań niebieskiego wynosiła więcej niż 1, gracz niebieski niemal zawsze wygrywał.

Zastosowanie heurystyki „tight” skutkowało zwiększaniem się liczby remisów.

Pozostałe heurystyki dawały podobne rezultaty do domyślnej.

Wpływ głębokości drzewa przeszukiwań oraz rodzaju funkcji oceny stanu na liczbę wygranych algorytmu. Badany gracz: biały.		
heurystyka (biały | niebieski)	głębokość (biały | niebieski)	wynik (z perspektywy białego) (wygrana | remis | przegrana)
(default | default)	(1 | 2)	(0 | 0 | 20)
	(2 | 2)	(0 | 0 | 20)
	(3 | 2)	(0 | 0 | 20)
	(4 | 2)	(0 | 0 | 20)
	(5 | 2)	(0 | 0 | 20)
(tight | default)	(1 | 2)	(0 | 7 | 13)
	(2 | 2)	(0 | 3 | 17)
	(3 | 2)	(4 | 12 | 4)
	(4 | 2)	(0 | 0 | 20)
	(5 | 2)	(0 | 0 | 20)
(enemy side | default)	(1 | 2)	(0 | 0 | 20)
	(2 | 2)	(1 | 1 | 18)
	(3 | 2)	(0 | 0 | 20)
	(4 | 2)	(1 | 1 | 18)
	(5 | 2)	(0 | 0 | 20)
(hight | default)	(1 | 2)	(0 | 0 | 20)
	(2 | 2)	(0 | 0 | 20)
	(3 | 2)	(0 | 0 | 20)
	(4 | 2)	(0 | 0 | 20)
	(5 | 2)	(0 | 0 | 20)

Tablica 2: Rezultaty pomiarów. Badany gracz biały. Parametry graczaj niebieskiego: głębokość - 2, heurystyka - domyślna.

Pomimo tego, że głębokość białego zmieniała się od wartości 1 do 5, podczas gdy głębokość niebieskiego pozostawała stała równa 2, dla każdej heurystyki poza heurystyką „tight", to niebieski wygrywał większość partii. Przeprowadzenie badania dla większej wartości głębokości białego nie było możliwe z uwagi na czas potrzebny na wykonanie testów.

W związku z powyższym przeprowadzono jeszcze jeden test. W stosunku do poprzedniego zmieniła się w nim tylko jedna rzecz: głębokość niebieskiego została zmieniona na wartość 1.

Wpływ głębokości drzewa przeszukiwań oraz rodzaju funkcji oceny stanu na liczbę wygranych algorytmu. Badany gracz: biały.		
heurystyka (biały | niebieski)	głębokość (biały | niebieski)	wynik (z perspektywy białego) (wygrana | remis | przegrana)
(default | default)	(1 | 2)	(9 | 0 | 11)
	(2 | 2)	(7 | 0 | 13)
	(3 | 2)	(9 | 0 | 11)
	(4 | 2)	(9 | 0 | 11)
	(5 | 2)	(11 | 0 | 9)
(tight | default)	(1 | 2)	(0 | 5 | 15)
	(2 | 2)	(0 | 1 | 19)
	(3 | 2)	(0 | 0 | 20)
	(4 | 2)	(0 | 0 | 20)
	(5 | 2)	(0 | 0 | 20)
(enemy side | default)	(1 | 2)	(10 | 0 | 10)
	(2 | 2)	(14 | 0 | 6)
	(3 | 2)	(8 | 0 | 12)
	(4 | 2)	(16 | 0 | 4)
	(5 | 2)	(9 | 1 | 10)
(hight | default)	(1 | 2)	(10 | 0 | 10)
	(2 | 2)	(8 | 0 | 12)
	(3 | 2)	(9 | 0 | 11)
	(4 | 2)	(6 | 0 | 14)
	(5 | 2)	(12 | 1 | 7)

Tablica 3: Rezultaty pomiarów. Badany gracz biały. Parametry graczaj niebieskiego: głębokość - 1, heurystyka - domyślna.

Zmniejszenie wartości głębokości niebieskiego spowodowało, że biały częściej wygrywał dla każdej heurystyki innej od „tight".

Dla pozostałych heurystyk nie widać wpływu głębokości działania algorytmu na rezultaty białego. Niezależnie od przyjętej głębi biały wygrywa około połowę partii pomimo tego, że w niektórych przypadkach głebokość jego obliczeń jest większa nawet o 4 w porównaniu do niebieskiego gracza.

4 Wnioski

Udało się zaimplementować minimax z przycinaniem $\alpha - \beta$.

Głębokość przeszukiwania drzewa znaczącą wpływa na na czas obliczeń.

Z wyników partii nie można jednoznacznie określić, która funkcja heurystyczna dawała najlepsze rezultaty. Wyniki partii w większości nie wyglądały, jakby były skorelowane z głębokością przeszukiwań. Jedynie w sytuacji, gdy głębokość niebieskiego wynosiła 1 przy badaniu jego działania, wynik różnił się od pozostałych. Funkcja heurystyczna „tight" dawała najgorsze rezultaty, w porównaniu do reszty heurystyk dla niebieskiego. Dla białego pozwalała jedynie trochę częściej uzyskać remis, gdy reszta heurystyk powodowała niemal wyłącznie porażki.

Przyczyną dziwnych rezultatów badań może być charakter gry do której algorytm został zastosowany. Gracz niebieski odpowiada na ruchy gracza białego, co być może jest preferowaną opcją, ponieważ czeka się wtedy tylko na błąd oponenta. Ustawienie figur na szachownicy wymusza to, że w końcu dojdzie do kontaktu i serii wymian figur. Czarny mając ostateczny ruch może na tym zyskać. Warto zaznaczyć, że idealna gra warcab kończy się remisem, więc biały może mieć tutaj mniejszą korzyść z zaczynania, niż ma to miejsce w przypadku niebieskiego.

W celu lepszego zbadania algorytmu należałoby przeprowadzić testy dla większych wartości głębokości przeszukiwania drzewa. Dodatkowo możnaby spróbować innych heurystyk lub zmodyfikować te już użyte.