Sztuczna inteligencja
Pracownia 5
Część 1

Zadania z (potencjalnej) części drugiej P5 wszystkie będą miały gwiazdkę. Czyli nie wpływają na maksimum.

Zadanie 1. (2p) Jest to zadanie „wirtualne": masz zaimlementować grę w Dżunglę samodzielnie, nie korzystając ze sprawdzaczki (być może już to zrobiłeś, wówczas zgłoś to prowadzącemu). Oznacza to, że chętni mogą (rezygnując z tych dwóch punktów) wykorzystać w swoim kodzie fragmenty kodu sprawdzaczki dla Dżungli.

Zadanie 2. (5p) Wybierz jedną grę z P4, dla której napisałeś już jakiś program grający w nią. Zaimplementuj dla niej algorytm Monte Carlo Tree Search i następnie:

a) Dobierz parametry obu programów tak, żeby ruch trwał mniej więcej 0.5 sekundy.

b) Przeprowadź mecz 10 partii (każdy program zaczyna 5 razy), notując wyniki meczów.

Do eksperymentów możesz wykorzystać sprawdzaczkę.

Zadanie 3. (3p) Skonstruuj sieć neuronową (lub inny mechanizm klasyfikujący), który próbuje przewidzieć dla danej sytuacji na planszy w Reversi, która strona wygra. Wykorzystaj ten mechanizm w funkcji heurystycznej oceniającej sytuację na planszy (wartością sytuacji dla białego będzie prawdopodobieństwo tego, że biały wygra).

W zadaniu tym klasyfikowana powinna być „surowa" sytuacja na planszy, bez żadnych cech wyznaczonych „ręcznie". Na stronie pojawi się zbiór danych (przykładowe sytuacje wraz z oceną), wyznaczone za pomocą pewnej liczby losowych gier (ocena sytuacji dla białego zależy od procentu gier losowych zaczynających się w tej sytuacji, które wygrał biały).

Zadanie 4. (3p+X) Podobnie jak w poprzednim zadaniu, Twoim celem jest konstrucja sieci neuronowej klasyfikującej sytuacje na planszy. Tym razem jednak powinieneś przekazać sieci cechy „wyliczone", na przykład bilans białych i czarnych pionków, informacje o narożnikach i ich okolicy, sytuację na brzegach planszy, liczba możliwych ruchów, liczba niezbijalnych pionków, ...

Plan minimum (3p) to wykorzystanie następujących 3 cech:

1) bilans postawionych na planszy pionków,

2) bilans pionków narożnych,

3) bilans pionków umożliwiających zdobycie rogu (czyli sąsiadujących z pustym polem narożnym)

Można dostać dodatkową uznaniową premię (do 2p) za dodawanie innych cech (do powyższych trzech) i empiryczne wykazanie ich sensowności.

Zadanie 5. (4p) Rozważamy prosty model autka, poruszającego się po dyskretnym torze (czyli podczas ruchu auto przeskakuje z kratki na kratkę). Stan autka dany jest przez cztery liczby całkowite: pozycję (x,y) oraz prędkość (vx, vy), przy czym każda składowa prędkości należy do zbioru $\{-3,-2,-1,0,1,2,3\}$. Akcja jest parą (dvx, dvy) mówiącą o tym, jak chcemy zmienić prędkość. Przyrost składowej prędkości należy do zbioru $\{-1,0,1\}$, czyli mamy 9 możliwych akcji.

Podstawowa mechanika autka dana jest zatem programem (**kod poprawiony**):

```
x,y,vx,vy = state
dvx,dvy = action   # (*)
vx += dvx
vy += dvy
if vx > 3: vx = 3
if vy > 3: vy = 3
if vx < -3: vx = -3
if vy < -3: vy = -3
x += vx
y += vy
new_state = x,y,vx,vy
```

Autko, które zakończy ruch na polu `e` kończy jazdę, wygrywa wyścig (i dostaje nagrodę 100), auto, które zakończy ruch na poza drogą (na polu `'.'`), również kończy jazdę, ale z nagrodą równą -100. Jeżeli autko rozpoczyna ruch na polu z rozlanym olejem (`'o'`), wówczas jego prędkość ulega zaburzeniu, realizowanemu w ten sposób, że w miejscu `(*)` wykonywany jest kod (**kod poprawiony**):

```
rx = random.choice( [-1,0,1])
ry = random.choice( [-1,0,1])
dvx += rx; dvy += ry
```

Auto startuje na wskazanym polu (na mapce oznaczonym przez `'s'`), z zerową prędkością. Napisz program, który wyznacza *optymalną politykę* dla autka, czyli funkcję, która dla każdego możliwego stanu wyznacza akcję. Politykę zapisujemy w formacie tekstowym składającym się z wierszy zawierających 6 liczb całkowitych (stan i akcję, którą należy wykonać w danym stanie):

```
x y vx vy   dvx dvy
```

Zakładamy ponadto, że $\gamma = 0.99$ (przypominam, że wypłata jest mnożona przez γ^t), a koszt jednego ruchu to 0.1.

Procedura testowania programu będzie następująca:

a) Program studenta wyznacza politykę dla konkretnego toru i zapisuje ją do pliku (w utworzonej przez studenta kartotece, w której są również mapy torów).

b) Program `policy_tester1.py` przegląda wszystkie polityki (i tory) w danej kartotece,

c) wykonując dla każdego pewną liczbę testów i informując, w jakim stopniu wyniki są satysfakcjonujące.

d) Dodatkowo zakładamy, że brak informacji o jakimś stanie powoduje wykonanie akcji poprzedniej. Nie można stać (czyli przebywać z prędkością 0) dwa razy na tym samym polu (taka sytuacja oznacza zapętlenie).

Polityka wyznaczana jest off-line (sprawdzaczka nie kotroluje czasu). Można politykę wyznaczyć przed zajęciami, wymaga się jedynie by dla każdego przypadku testowego student mógł powtórzyć obliczenia polityki przy prowadzącym (czyli w czasie zajęć).

Zapowiedzi P5.2

I. Zadania (za 1p) polegające na wystawieniu swojego programu do walki z przygotowanymi przez prowadzących agentów. Będzie co najmniej 2 agentów dla Reversi i co najmniej 1 dla Dżungli.

II. Trudniejsza wersja zadania z autkami, z dużo większymi planszami i/lub trudniejszymi zasadami ruchu.

III. Zadanie za 1 + X punktów: wystawienie programu do turnieju w Reversi (X zależy od wyników turnieju)

IV. Zadanie za 1 + X putnków: wystawienie programu do turnieju w Dżungli (X zależy od wyników turnieju)