**WSI zad. 4**@e23f8dc087ae7d0a8c1fe515df65ac5580b354f1 **Grzegorz Rypeść**

W grze komputerowej do wyboru akcji wykorzystujemy zawikłaną funkcję definiowaną przez nasz umysł. W tym zadaniu należy dokonać prostej aproksymacji tej funkcji w grze w węża przy użyciu klasyfikatora. Fachowo, problem ten nazywa się w literaturze anglojęzycznej *behavioral cloning*. Rodzaj klasyfikatora jaki Państwo zaimplementujecie zależy od numeru indeksu w sposób następujący:

Niech x oznacza trzy pierwsze liczby po przecinku z wyrażenia $\sqrt{3.14 * indeks\, studenta}$ . Osoby, których reszta z dzielenia x przez 3 wynosi:

- 0, implementują regresję logistyczną.
- 1, implementują maszynę wektorów nośnych (SVM).
- 2, drzewo ID3.

Proszę te obliczenia oraz swój indeks zawrzeć w raporcie.

**Polecenie**:
Należy wygenerować zbiór danych, zaimplementować klasyfikator, wytrenować model na danych i przetestować w grze. Model ma na podstawie stanu gry zwracać akcję, jaką ma wykonać wąż.

Proszę zagrać w załączoną implementację gry *Snake* (*main.py*). Jest ona tak napisana, że po wyjściu z gry zostanie zrzucony na dysk zapis z rozgrywki. Taki zapis składa się z przykładów danych będącymi krotkami (ang. tuple). Krotka zawiera stan gry i podjętą przez Państwa dla niego akcję. Musicie zagrać kilka razy w grę, wygenerować kilka zapisów, a następnie dokonać inżynierii atrybutów, aby przekształcić stany gry w atrybuty, na których wytrenujecie modele. Zadanie jest problemem klasyfikacji - wąż może poruszać się w górę, prawo, dół i lewo. Istnieją zatem 4 klasy. Chciałbym, aby atrybutów skontruowali Państwo co najmniej 8. Polecam zrobić następujące: 4 binarne (czy w sąsiadującym kwadracie jest przeszkoda) i 4 binarne (czy w danym kierunku jest jedzenie). Złe atrybuty spowodują, że model nie będzie grał dobrze. Proszę krótko opisać stworzone atrybuty w raporcie. Połączone zapisy rozgrywek przekształcone inżynierią atrybutów stanowią zbiór danych. Gotowy, powien być macierzą typu *numpy.ndarray* lub *torch.tensor*. Wiersze to przykłady danych, a kolumny to atrybuty. Przykładów danych powinno być co najmniej 2000.

Zbiór danych należy podzielić na zbiór treningowy i testowy w proporcji 4:1. Do oceny jakości modelu na zbiorze danych proszę użyć miary dokładności (funkcję można wziąć z biblioteki *scikit-learn*, albo *torchmetrics*). Proszę odpowiedzieć na następujące pytania: czy i dlaczego występuje przeuczenie lub niedouczenie (ang. overfitting, underfitting). Jak można zminimalizować ten problem? Proszę to sprawdzić dla zbioru danych wynoszącego 1%, 10% i 100% całego zbioru danych.

Proszę wybrać jeden hiperparametr klasyfikatora i dla 5 różnych jego wartości przebadać i opisać wpływ na wyniki uzyskiwane na zbiorze treningowym i testowym.

Dysponując wytrenowanym modelem należy dokończyć implementację klasy *BehavioralCloningAgent* reprezentującą agenta. Proszę dla 100 rozgrywek węża sprawdzić jakie wyniki osiąga model. Czy model gra tak samo dobrze jak Państwo? Dlaczego tak lub dlaczego nie?

Raport oraz pliki proszę spakować do pliku o nazwie WSI-4-NAZWISKO-IMIE.zip i przesłać na adres grzegorz.rypesc.dokt@pw.edu.pl.

**Uwagi:**

Klasyfikatory muszą być własnoręcznie zaimplementowane, nie można wziąć ich sobie z gotowej biblioteki. Osoby ambitne mogą natomiast użyć modeli z biblioteki *scikit-learn* do porównania wyników. Polecam las losowy.

Jeżeli implementowany model jest klasyfikatorem binarnym, należy stworzyć ich 3 lub 4 do problemu klasyfikacji ruchu węża (ang. One-vs-Rest classification). W przypadku 4 klasyfikatorów każdy przewiduje, czy wąż powinien poruszyć się w danym kierunku czy nie.

Warto zrobić **kopię zapasową** plików z danymi, odczytanie ich w trybie zapisu spowoduje nadpisanie niszcząc je.

Nie wymagam od Państwa, aby agent grał w węża bardzo dobrze. Zagadnienie to jest skomplikowane i wykracza poza ramy tego przedmiotu. Minimum 4 punkty powinien jednak średnio w grze uzyskać.

Zbiorów danych proszę nie wyrzucać, jeszcze się przydadzą. Kod warto napisać modułowo, reużywalnie. Też się jeszcze przyda.

**Wskazówka dla osób, które chcą stworzyć świetny model:**

Generując zbiór danych warto mieć na uwadze, że nie powinno się dla dwóch identycznych stanów podejmować różnych akcji. Warto trzymać się pewnej strategii, np. kręcimy po mapie kółeczka zgodnie ze wskazówkami zegara i zahaczamy o jedzenie lub idziemy, niczym dzik, po jedzenie najkrótszą możliwą ścieżką. Można spróbować też dokonać wstępnego przetworzenia danych i usunąć przykłady sprzeczne ze sobą. Im więcej danych i im lepsze atrybuty tym lepiej. Można też usunąć, bądź zmienić przykłady danych, które doprowadziły do przegrania rozgrywki, azaliż po co model ma powielać to zachowanie?