
Laboratorium

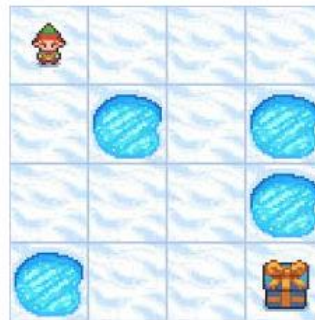
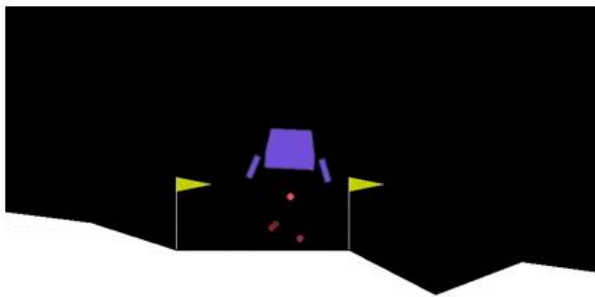
Test algorytmów w wirtualnym środowisku

Uwaga! To ostatni zestaw zadań w tym semestrze. Należy się z nich rozliczyć do ostatnich zajęć. Każde zadanie jest warte 1 punkt.

Zadanie 1

Na wykładzie były pokazywane biblioteki Gym i Gymnasium do symulacji pewnych procesów (lub grania) w wirtualnym środowisku.

- a) Uruchom pliki pythonowe z wykładu. Sprawdź, czy gry LunarLander i FrozenLake działają. Pamiętaj o zainstalowaniu odpowiednich paczek i ewentualnych dodatkowych czynnościach (instalacja swig -> patrz wykład).



- b) Uruchom dodatkowe gry. Najlepiej po jednej z pięciu kategorii (Classic Control, Box2D, ToyText, MuJoCo, Atari). Uwaga! Każda kategoria może wymagać instalacji dodatkowych bibliotek.
- c) Przeglądając dokumentację, zidentyfikuj po jednej grze z następujących kategorii:
- Stan gry i zestaw akcji są dyskretne (tzn. jest to skończony zestaw).
 - Stan gry jest ciągły (nieskończony, liczby zmiennoprzecinkowe), ale zestaw akcji jest dyskretny.
 - Stan gry i zestaw akcji jest ciągły.
- Uruchom te gry i dopisz w komentarzy w kodzie, który z typów to jest.
- d) Dla dwóch wybranych gier, podmień losowe próbki akcji, na zestaw akcji stworzony przez Ciebie, który będzie dobrze działał („dobrze” tzn. osiągniemy sukces, albo przynajmniej zbliżymy się do osiągnięcia sukcesu).

Zadanie 2

Rozwiąż FrozenLake8x8 (z wyłączonymi poślizgami) oraz LunarLander za pomocą algorytmu genetycznego. Możesz wykorzystać paczkę pygad. W kodzie dodaj komentarze opisujące, jaką postać mają chromosomy i jak działa funkcja fitness.

Pokaż jak działa najlepsze rozwiązanie znalezione dla obu gier.

Zadanie 3

Na rozgrzewkę, użyjemy paczki `simpful`, by stworzyć prosty kontroler rozmyty do obliczania napiwków (przykład był na wykładzie).

a) Poczytaj o paczce `simpful` np. tutaj:

- <https://pypi.org/project/simpful/>
- https://pythonhosted.org/scikit-fuzzy/auto_examples/plot_tipping_problem_newapi.html
- https://www.researchgate.net/publication/346395808_Simpful_A_User-Friendly_Python_Library_for_Fuzzy_Logic

Zwróć uwagę na wstawki kodu i sposób tworzenia kontrolerów (zmienne, reguły, wyostrzanie).

- b) Zainstaluj paczkę i skopiuj z wybranej strony kod tworzący system do dawania napiwków (3 zmienne lingwistyczne, 3 reguły).
- c) Wyświetl wykresy zmiennych lingwistycznych.
- d) Przetestuj działanie kontrolera. Daj kilka danych (liczby dla jedzenia i obsługi) i wyświetl jaki napiwek (0-30%) proponuje system dla tych inputów.

Następnie wykorzystaj paczkę `simpful` do stworzenia kontrolera rozmytego dla wybranej gry z Gym (tutaj najlepiej sprawdzi się gra ze stanami i akcjami ciągłymi/zmiennoprzecinkowymi) np.

- LunarLander (wersja continuous)
- Bipedal Walker
- Pendulum

Rozwiązanie powinno się składać z następujących kroków:

- e) Zdefiniowanie sensownych zmiennych lingwistycznych dla każdej zmiennej. Wyświetl wykresy.
- f) Zdefiniowanie zestawu sensownych reguł wnioskowania rozmytego. Zastanów się czy lepszym operatorem będą „or” czy „and”.
- g) Odpalenie systemu i obliczanie wyostrzonych zmiennych sterujących (consequents). Zmienne powinny być obliczane w każdej klatce symulacji. W każdej klatce animacji fuzzy controller oblicza co trzeba zrobić i wykonuje tę akcję.

Postaraj się, by system działał w miarę dobrze (nie musi działać idealnie). W razie czego zmodyfikuj zmienne lingwistyczne lub reguły rozmyte.

Zadanie 4

Należy napisać trzy programy uczące się, z dziedziny DP i RL.

Dla wybranych gier z Gym przetestuj algorytmy poznane na wykładzie.

- Value Iteration Algorithm (Dynamic Programming)
- Q-Learning (Reinforcement Learning)
- Deep Q-Network (Reinforcement Learning)

Zwróć uwagę, czy typ gry pasuje do algorytmu (ciągłe vs dyskretne). Zasymluj znalezione najlepsze rozwiązanie.