Przydział tematów

Temat	Zespół
LL1	Roman Moskalenko, Pavel Tarashkevich
LL2	-
LL3	-
LL4	Gabriela Ossowska, Jeremiasz Jaworski
LL5	Wojciech Sitek, Dawid Brzozowski
LL6	Monika Plata, Mateusz Kordowski
LL7	Artur Grudkowski, Radosław Mierzwa
LL8	Łukasz Wolanin, Marcin Różański
LL9	Michał Szaknis, Przemysław Stawczyk
LL10	Anna Kudzia, Arkadiusz Rybski

Tematy

LL1. AnyTrading Stocks
LL2. AnyTrading Forex EUR
LL3. AnyTrading Forex USD

Zapoznaj się z *gym-anytrading* (https://github.com/AminHP/gym-anytrading) oraz zbiorem danych tam dostępnym:

- **LL1** - STOCKS_GOOGL
- **LL2**, **LL3** - FOREX_EURUSD_1H_ASK

Przygotuj agenta podejmującego akcje na rynku i uczącego się ze wzmocnieniem oraz porównaj jego skuteczność ze strategią losową. Przetestuj co najmniej 2 różne algorytmy uczenia ze wzmocnieniem. Przygotuj skrypt umożliwiający wizualizację działań wytrenowanego agenta.
Dla **LL2** - jako walutę posiadaną przyjmij EUR.
Dla **LL3** - jako walutę posiadaną przyjmij USD.

LL4. Coach Doom

Zapoznaj się z *Coach* (https://github.com/IntelLabs/coach). Uruchom zawarte w nim środowisko *Doom*. Przygotuj agenta grającego w tę grę i uczącego się ze wzmocnieniem oraz porównaj jego skuteczność ze strategią losową. Przetestuj co najmniej 2 różne algorytmy uczenia ze wzmocnieniem dostępne w tej bibliotece. Przygotuj skrypt umożliwiający wizualizację działań wytrenowanego agenta.

LL5. ML-Agents

Zapoznaj sie z *ML-Agents* (https://github.com/Unity-Technologies/ml-agents). Uruchom zawarte w nim środowisko *Basic* lub *Hallway*. Przygotuj agenta uczącego się ze wzmocnieniem działającego na wybranym środowisku oraz porównaj jego skuteczność ze strategią losową. Przetestuj wbudowane w bibliotekę algorytmy PPO i SAC. Przygotuj skrypt umożliwiający wizualizację działań wytrenowanego agenta.

LL6. ACME

Zapoznaj się z *acme* (https://github.com/deepmind/acme). Wybierz dowolne środowiską z grą Atari z OpenAI Gym (https://gym.openai.com/envs/#atari). Przygotuj agenta uczącego się ze wzmocnieniem działającego na wybranym środowisku oraz porównaj jego skuteczność ze strategią losową. Przetestuj co najmniej 2 różne algorytmy uczenia ze wzmocnieniem dostępne w tej bibliotece. Przygotuj skrypt umożliwiający wizualizację działań wytrenowanego agenta.

LL7. PPO Porównanie

Porównaj działanie różnych implementacji algorytmu PPO. Wybierz co najmniej 2 różne wersje z innych bibliotek (jedna z wersji może być implementacją własną) i przetestuj ich działanie na środowisku HalfCheetach (https://gym.openai.com/envs/HalfCheetah-v2/). Przedstaw i skomentuj podobieństwa i różnice obu wersji. Przygotuj skrypt umożliwiający wizualizację działań wytrenowanego agenta.

LL8. SAC gSDE

Porównaj skuteczność algorytmu SAC bez oraz z mechnanizmem gSDE (https://arxiv.org/pdf/2005.05719v2.pdf). Porównanie wykonaj na podstawie wyników ns środowisku HalfCheetach (https://gym.openai.com/envs/HalfCheetah-v2/). Opisz wpływ gSDE na działanie algorytmu SAC. Przygotuj skrypt umożliwiający wizualizację działań wytrenowanego agenta.

LL9. ACER w PyTorch

Zaimplementuj algorytm Aktor-Krytyk z powtarzaniem doświadczenia i lambda-estymatorami przyszłych nagród, używając biblioteki PyTorch. Możesz wesprzeć się implementacją z https://github.com/lychanl/acerac używającą TensorFlow. Sprawdź działanie algorytmu na środowisku HalfCheetah (https://gym.openai.com/envs/HalfCheetah-v2/). Przygotuj skrypt umożliwiający wizualizację działań wytrenowanego agenta.

LL10. ACER z adaptacyjną eksploracją

Zaimplementuj algorytm Aktor-Krytyk z powtarzaniem doświadczenia, lambda-estymatorami przyszłych nagród i adaptacyjnie dobieraną eksploracją według koncepcji: http://home.elka.pw.edu.pl/~llepak/files/ACERAX4USD.pdf. Możesz zaimplementować ten algorytm rozbudowując odpowiednio implementację algorytmu ACER z https://github.com/lychanl/acerac. Sprawdź działanie zaimplementowanego algorytmu na środowisku HalfCheetah (https://gym.openai.com/envs/HalfCheetah-v2/). Przygotuj skrypt umożliwiający wizualizację działań wytrenowanego agenta.

Zasady oceniania

Projekt - 35 pkt, zalicza 18 pkt

- **Dokumentacja wstępna - 5 pkt**
    - **Termin oddania - 19.11.2021**
    - Oddanie do 26.11.2021 - max. 3 pkt
    - Po 26.11.2021 dokumentacje nie będą przyjmowane
    - **Warunek konieczny zaliczenia projektu**
- **Dokumentacja końcowa + kod - 30 pkt**
    - **Oceniane aspekty projektu:**
        - Reprodukowalność, uzyskane wyniki, analiza i wnioski
        - Czytelność i poprawność kodu
        - Czytelność i poprawność dokumentacji
    - **Konieczna prezentacja rozwiązania i dokumentacji końcowej** stacjonarnie (p. 25 WEiTI) lub zdalnie (MS Teams)
    - Do 14.01.2022 można oddać lub skonsultować projekt z możliwością "bezstratnej" poprawy
    - **Termin oddania - 21.01.2022**
    - Oddanie do 28.01.2022 - max. 23 pkt
    - Oddanie do 04.02.2022 - max. 15 pkt
    - Po 04.02.2022 projekty nie będą przyjmowane

Dokumentacja wstępna

Dokumentacja wstępna powinna prezentować Państwa zapoznanie się z przydzielonym tematem oraz wstępne przygotowanie planu realizacji. Powinna ona zawierać (w zależności od przydzielonego tematu):

- Przyjęte założenia projektowe
- Krótki opis wybranego lub przydzielonego środowiska
- Algorytmy wybrane do zaimplementowania lub przetestowania
- Biblioteki wybrane do realizacji projektu
- Propozycja eksperymentów do przeprowadzenia

Format dokumentacji wstępnej: PDF (1-2 strony)

Dokumentacja końcowa + kod

Dokumentacja końcowa wraz z kodem źródłowym są końcowym efektem Państwa pracy. Ich ocena dzieli się na trzy części:

- **Reprodukowalność, uzyskane wyniki, analiza i wnioski**
    - Instrukcja stworzenia wirtualnego środowiska oraz instalacji wymaganych bibliotek (instrukcja w pliku *README*, biblioteki z wersjami w *requirements.txt* (venv) lub *environment.yml* (conda))
    - Instrukcja uruchomienia skryptów pozwalających na odtworzenie wszystkich omówionych w dokumentacji eksperymentów

    - Dokładne i czytelne przedstawienie uzyskanych wyników
    - Analiza uzyskanych rezultatów
    - Wnioski wynikające z uzyskanych wyników i ich analizy
- **Czytelność i poprawność kodu**
    - Poprawność zaimplementowanego rozwiązania oraz jego zgodność z opisem w dokumentacji
    - Rozdzielenie algorytmów od problemów, które rozwiązują (zmiana algorytmu nie powinna powodować konieczności przepisania całego kodu, i vice versa)
    - Samodokumentowalność, *clean code* (odpowiednie nazwy zmiennych, wyjaśnienie bardziej skomplikowanych fragmentów kodu, podział kodu, brak “magicznych stałych”)
- **Czytelność i poprawność dokumentacji**
    - Opis zmian względem dokumentacji wstępnej
    - Poprawność gramatyczna i językowa
    - Czytelność zamieszczonych tabel, wykresów i rysunków

Format dokumentacji końcowej: PDF (kilka stron), Jupyter Notebook

Kontakt

- **p. 25 WEiTI (po wcześniejszym uzgodnieniu)**
- **MS Teams**
- **mail:** lukasz.lepak.dokt@pw.edu.pl