Ciągłe przestrzenie

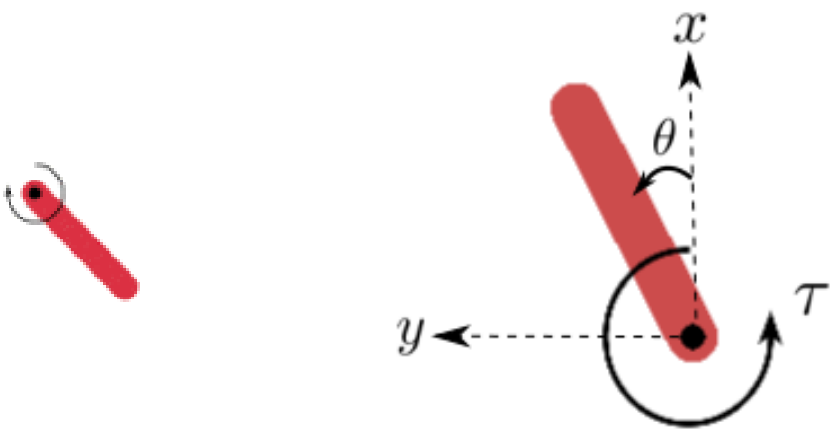
Małgorzata Duda, Maja Gurdek

Cel ćwiczenia :

Celem ćwiczenia jest zastosowanie algorytmu uczenia ze wzmocnieniem Proximal Policy Optimization (PPO) do szkolenia agenta w środowisku ze stanami ciągłymi Pendulum z OpenAI Gym, zbadanie wydajności tego algorytmu z różnymi zestawami hiperparametrów i przeanalizowanie wynikowych krzywych uczenia się.

Opis problemu :

Środowisko **Pendulum**, dostarczone przez OpenAI Gym, symuluje wahadło przymocowanego na jednym końcu do stałego punktu, aby drugi koniec, mógł się wokół niego obracać. Pozycja wyjściowa wahadła jest wybierana losowo. Celem gry jest wychylenie wahadła do pozycji pionowej, poprzez zastosowanie momentu obrotowego do obracającego się końca, aby środek ciężkości był nad punktem stałym.



Przestrzeń akcji reprezentuje wartość momentu obrotowego.

Num	Action	Min	Max
0	Torque	-2.0	2.0

Przestrzeń obserwacji to współrzędne wolnego końca wahadła oraz przyspieszenie kątowe.

Num	Observation	Min	Max
0	x = cos(theta)	-1.0	1.0
1	y = sin(theta)	-1.0	1.0
2	Angular Velocity	-8.0	8.0

Nagroda jest opisana wzorem: $r = -(theta^2 + 0.1 * theta\_dt^2 + 0.001 * torque^2)$
Maksymalna nagroda jest równa zero (dla wahadła ustawionego w pozycji pionowej).

Proximal Policy Optimization (PPO) jest jednym z algorytmów uczenia ze wzmocnieniem. Algorytm poprawia stabilność uczenia agenta, poprzez unikanie zbyt dużych zmian w polityce. Główną ideą jest optymalizacja zastępczej funkcji celu, która przybliża poprawę polityki oraz zapewnia, że po aktualizacji nowa polityka nie będzie zbyt odległa od starej. W tym celu PPO wykorzystuje przycinanie zakresu możliwych wartości, aby uniknąć zbyt dużych aktualizacji.

Algorytm wykorzystuje politykę MLP (Multi-Layer Perceptron Policy). Jest to architektura sieci neuronowej typu feedforward, która mapuje stany (współrzędne wahadła oraz przyspieszenie kątowe) na akcje w środowisku (moment obrotowy).

Algorytm PPO w skrócie :

1. Zbieranie trajektorii - agent wchodzi w interakcję ze środowiskiem, wykonując akcje zgodnie z aktualną polityką i zbiera trajektorie, które składają się ze stanów, akcji, nagród i następnych stanów.
2. Obliczanie szacunkowych korzyści - korzyści reprezentują jakość każdego działania w porównaniu do średniego działania w danym stanie. Są one szacowane przy użyciu funkcji wartości lub funkcji korzyści.
3. Aktualizacja polityki poprzez optymalizację zastępczej funkcji celu.
4. Optymalizacja funkcji wartości (opcjonalnie).
5. Iteracja i powtarzanie - poprzednie są powtarzane przez wiele iteracji, umożliwiając agentowi uczenie się i udoskonalanie swojej polityki w czasie.

Realizacja rozwiązania :

Do realizacji rozwiązania została wykorzystana biblioteka Stable Baselines3, która umożliwiła implementację algorytmu PPO oraz szkolenie agenta w środowisku Pendulum. Kluczową rolę w wydajności agenta odgrywają hiperparametry. Obejmują one:

- **szybkość uczenia** - tempo, z jakim agent aktualizuje wagi w sieci
- **rozmiar parti**i - liczba próbek, które są używane do aktualizacji wag sieci podczas jednej epoki treningowej.
- **współczynnik dyskontowy (gamma)** - określa wagę przyszłych nagród w procesie podejmowania decyzji.

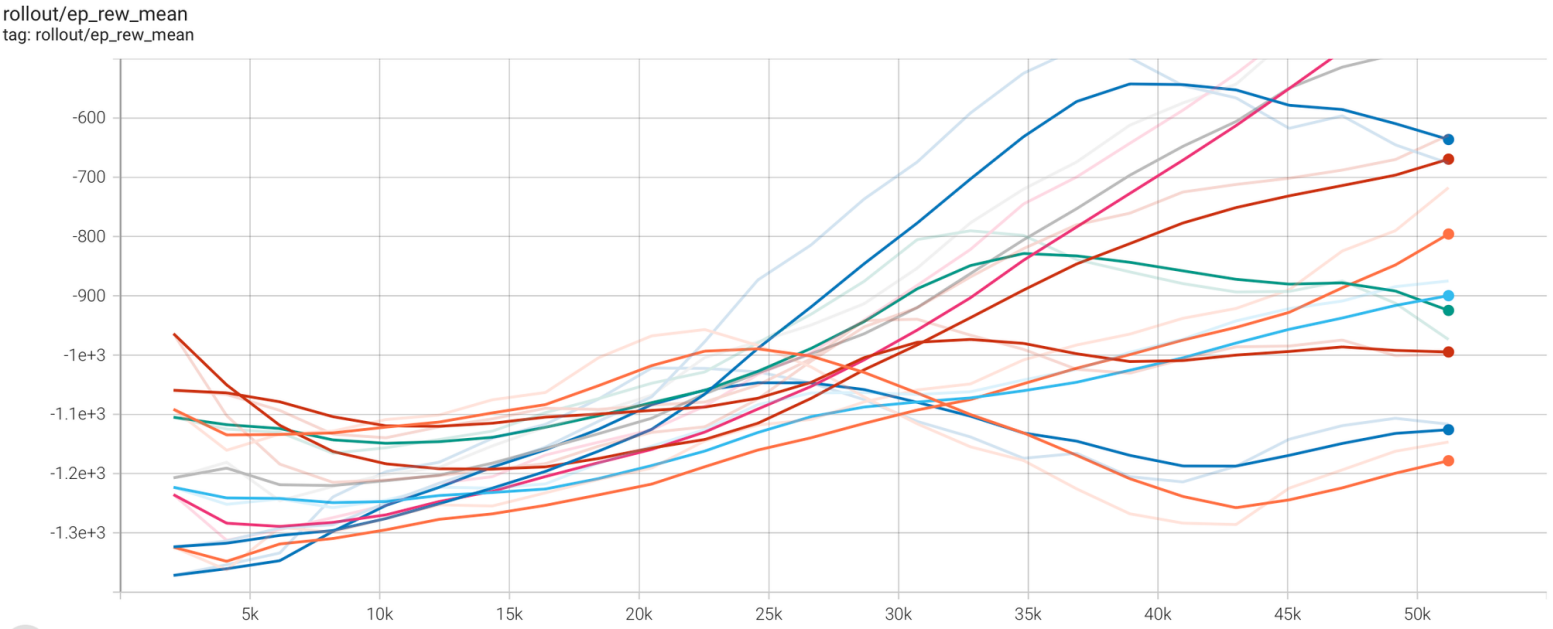
Kod iteruje po każdym zestawie zdefiniowanych jako słownik hiperparametrów. W każdej iteracji przeprowadzanych jest dziesięć eksperymentów w celu oceny wydajności agenta.

Wykonywane są następujące kroki:

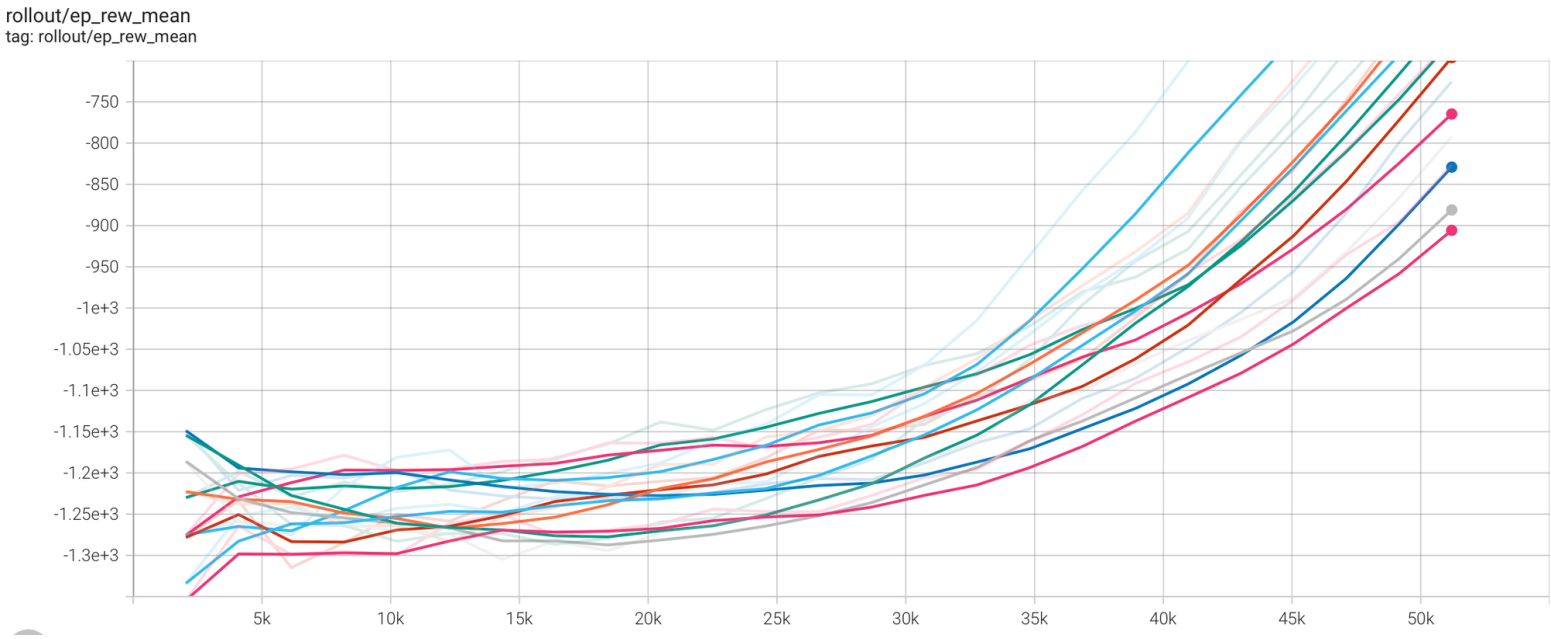
- Tworzone jest środowisko wektorowe w celu obsługi równoległego wykonywania wielu środowisk.
- Inicjalizowany jest agent z określonymi hiperparametrami.
- Konfigurowane jest rejestrowanie w celu zapisania postępu szkolenia.
- Agent trenowany jest w środowisku Pendulum przez łącznie 50 000 kroków czasowych.
- Wyniki są odczytywane z wygenerowanego przez loggera pliku "progress.csv".
- Obliczana jest średnia i odchylenie standardowe nagród epizodów.
- Po zakończeniu wszystkich eksperymentów kod wizualizuje krzywe uczenia przedstawiające średnie nagrody oraz odchylenie standardowe.

Rezultaty:
Przeprowadziłyśmy eksperymenty dla trzech różnych zestawów parametrów

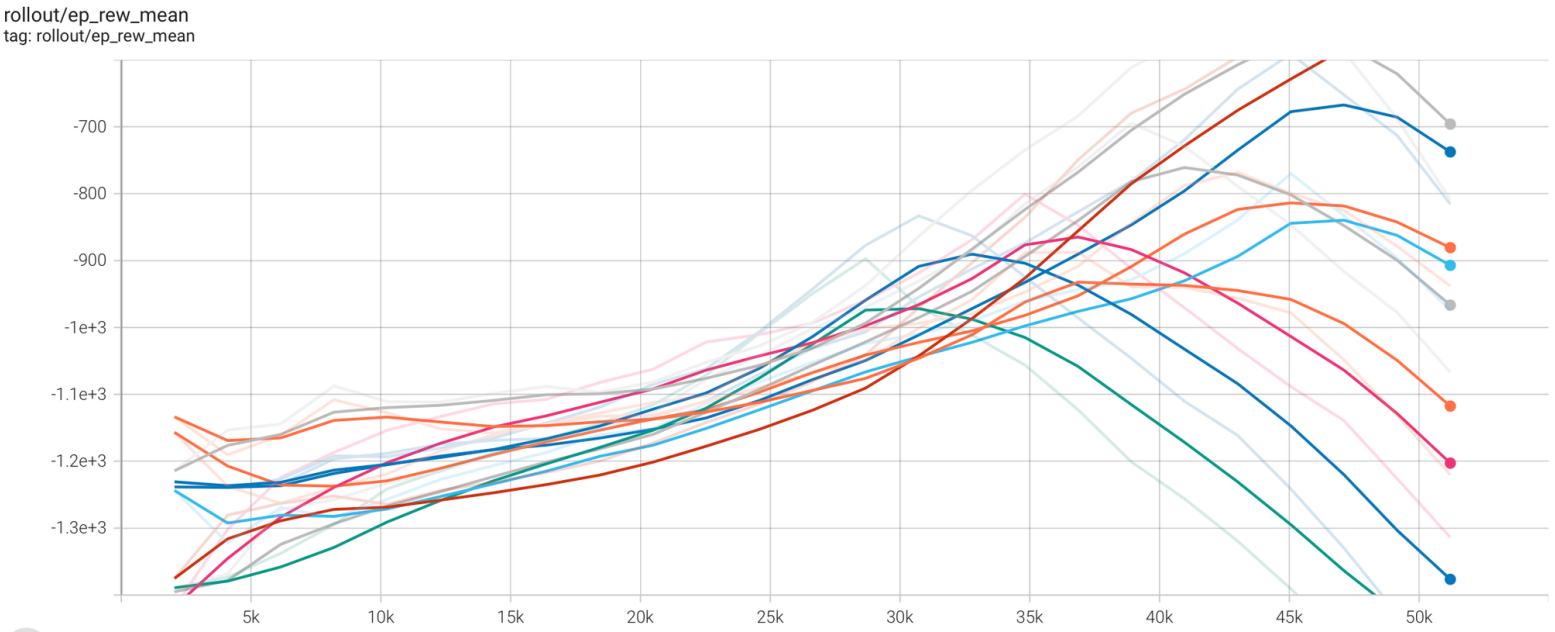
1. learning_rate: 0.01, batch_size: 128, gamma: 0.85



2. learning_rate: 0.001, batch_size: 128, gamma: 0.9



3. learning_rate: 0.01, batch_size: 64, gamma: 0.99



Czas wykonania eksperymentu waha się w granicach 12 - 15 sekund, czyli jeden krok czasowy zajmuje około 0,00027 sekund.

Wnioski:
W trakcie ćwiczenia przeprowadziłyśmy kilka eksperymentów z różnymi zestawami hiperparametrów. Krzywe uczenia pokazały jak różne ustawienia hiperparametrów wpływały na postępy i stabilność szkolenia agenta. Wybór hiperparametrów miał widoczny wpływ na proces uczenia się i końcową wydajność.
Drugi przypadek (learning_rate: 0.001, batch_size: 128, gamma: 0.9) wyróżniał się od pozostałych, ponieważ wykazał się największą stabilnością uczenia oraz osiągał najlepsze rezultaty.
Ćwiczenie wykazało skuteczność PPO w rozwiązywaniu zadania sterowania w środowisku Pendulum, w zależności od ustawionych hiperparametrów. Agent nauczył się balansować wahadłem poprzez zastosowanie odpowiedniego momentu obrotowego. Krzywa uczenia wykazała początkową fazę eksploracji, po której następowała stopniowa poprawa nagród, gdy agent nauczył się stabilizować wahadło.