Walker 2D - raport

Uczące się Systemy Decyzyjne
Uczenie z Wzmocnieniem

Ula Żukowska
Paweł Młyniec
Hubert Borkowski

28 lutego 2021

Spis treści

1 Cel zadania

Celem projektu było zaimplementowanie algorytmów uczenia ze wzmocnieniem oraz porównanie ich działania w środowisku testowym *Walker2D* [1]. Zadanie miała być stworzone dla domyślnych interwałów czasowych jak i podwójnej dyskretyacji czasu. Jest to jedno ze środowisk z zestawu *Gym*, czyli grupy problemów umożliwiających testowanie i porównywanie działania różnych algorytmów uczenia ze wzmocnieniem. Są one bardzo uniwersalne - użytkownik może wykorzystać je do testowania dowolnego algorytmu oraz korzystać z dowolnych bibliotek umożliwiających działanie testowanych algorytmów.

W ramach projektu zostały zastosowane dwa algorytmy: DDPG (*Deep Deterministic Policy Gradient*) oraz SAC (*Soft Actor Critic*). Porównane zostały ich działanie oraz tempo uczenia się agenta ze środowiska testowego. Opis programu oraz wyniki testów przedstawione zostały w dalszej części raportu.

2 Opis decyzji projektowych

- Jako środowisko zastosowano nakładkę na bazowe środowiska z biblioteki *gym*, bibliotekę *pybullet-gym* [2].
- W ramach projektu zostały zastosowane dwa algorytmy: DDPG oraz SAC. W implementacji jako bazę przyjęto implementacje pochodzące z publicznie dostępnego repozytorium z przykładami do nauki uczenia ze wzmocnieniem [3].
- Podjęto próbę użycia algorytmu PPO jednak wystąpiły trudności z dostosowaniem go do współpracy z ciągłą przestrzenią akcji i stanów.
- Algorytmy zostały obudowane kodem pozwalającym uruchamianie ich z szeregiem parametrów w celu przeprowadzenia testów i doboru ustawień dających najlepsze wyniki. Badane parametry to: *liczba neuronów sieci neuronowej*, *liczba warstw sieci neuronowej*, *learning rate*.
- Dodatkowo została dodana opcja uruchamiania treningu dla domyślnej dyskretyzacji czasu (*60 klatek/s*) oraz dla dwukrotnie zagęszczonej dyskretyzacji (*120 klatek/s*).
- Program w trakcie trenowania wykonuje zapisy do logów oraz zapisuje do pliku model co określoną liczbę epizodów.

3 Opis przeprowadzonych testów

W ramach projektu porównane zostało działanie zaimplementowanych algorytmów. Wykonane zostały testy tempa i skuteczności procesu nauki. Zbadany został również wpływ architektury sieci neuronowych na proces uczenia agenta. Wyniki testów przedstawione zostały poniżej.

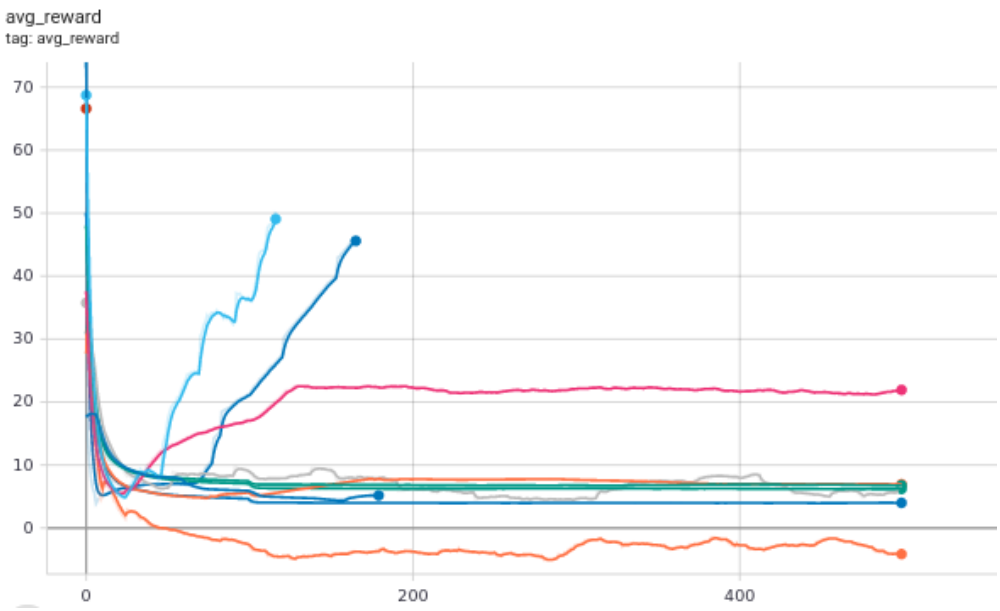
3.1 DDPG

Dla tego algorytmu przetestowano następujące wartości hiperparametrów:

- liczba neuronów = [128, 256, 512]
- liczba warstw = [2, 3]
- learning rate = [0.001, 0.0005, 0.01]

Testy każdej architektury trwały przez 500 epizodów lub do czasu, gdy komputer pod wpływem obliczeń się zawieszał.

Wyniki testów różnych architektur dla algorytmu DDPG zostały przedstawione za pomocą TensorBoarda: `https://tensorboard.dev/experiment/virHxrokRNa9OxmsYQ4f1g/`. Przedstawiają się one następująco.



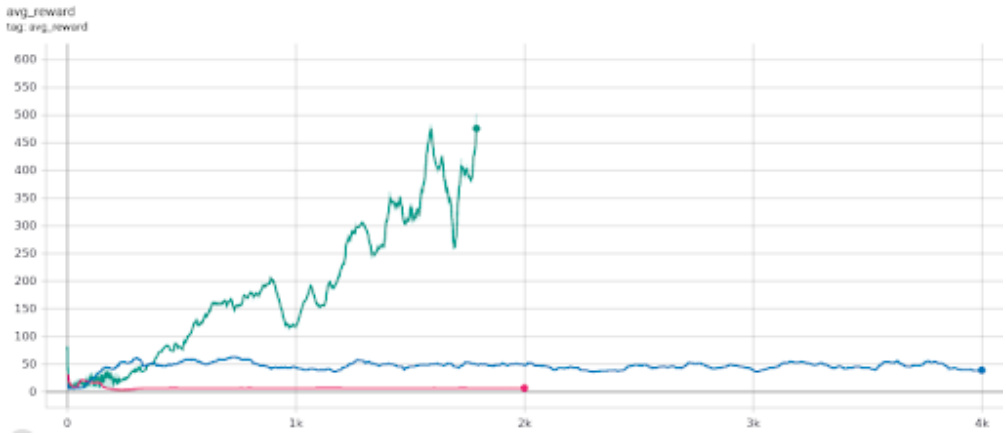
Rysunek 1: Wyniki testów różnych architektur dla algorytmu DDPG

Liczba warstw	Liczba neuronów	Współczynnik uczenia	Średnia nagroda	Liczba epizodów
2	512	0.01	6.2	500
3	128	0.001	50	116
3	512	0.01	-4.2	500
2	256	0.01	6.7	500
3	128	0.0005	47	116
3	256	0.0005	21.11	138
3	256	0.001	6.9	500

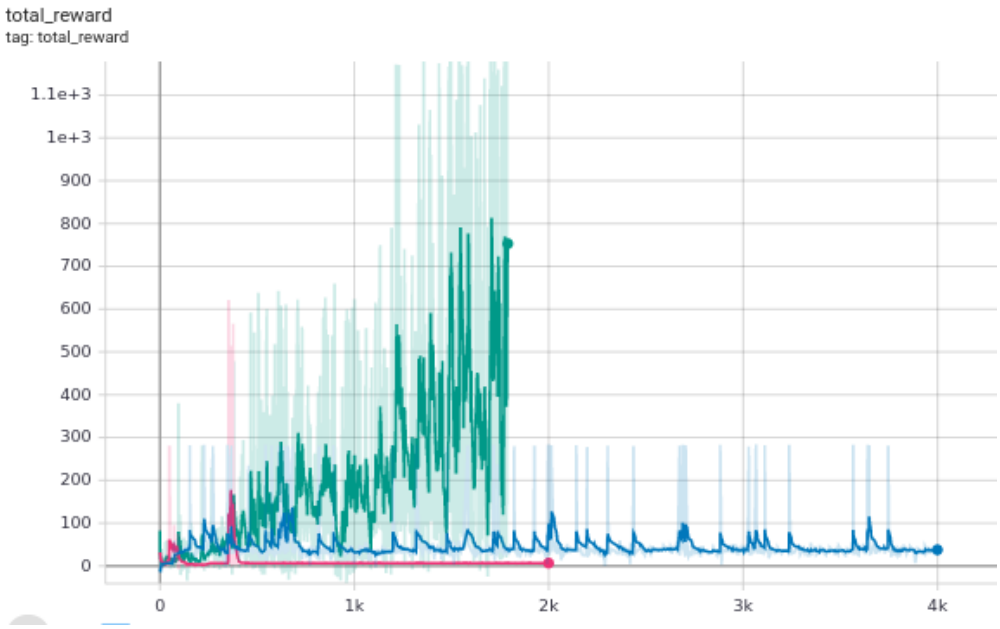
Tablica 1: Wyniki niektórych testów różnych architektur dla algorytmu DDPG

Znacząco lepiej od innych wypadła architektura złożona z 128 neuronów w 3 warstwach ze współczynnikiem uczenia 0.001. Dla niej też zostały przeprowadzone dalsze testy - zwiększono maksymalną liczbę epizodów oraz średnią nagrodę, po której algorytm zakończy działanie.

Test został przeprowadzony dwukrotnie na dwóch różnych maszynach. Na pierwszej maszynie po 4000 epizodach model zdobył średnią nagrodę równą 39. Natomiast testy na drugiej maszynie uzyskały lepsze wyniki. Po 1700 epizodach nagroda chwilowa dochodziła do 1300, a średnia nagroda przekroczyła 500. Wykresy średniej nagrody z ostatnich 100 epizodów oraz nagrody z aktualnego epizodu przedstawiają się następująco:



Rysunek 2: Wyniki testów DDPG dla maszyny pierwszej i drugiej. Średnia nagroda od epizodu



Rysunek 3: Wyniki testów DDPG dla maszyny pierwszej i drugiej. Chwilowa nagroda od epizodu

Jak widać na pierwszej maszynie model nie uczył się zbyt dobrze. Natomiast testy na drugiej maszynie uzyskały lepsze wyniki. Po 1700 epizodach nagroda chwilowa dochodziła do 1300, a średnia nagroda przekroczyła 500. Wykresy średniej i chwilowej nagrody są przedstawione poniżej. Dla dwukrotnie zwiększonej dyskretyzacji czasu algorytm po 1000 krokach uzyskał średnią nagrodę równą 350.

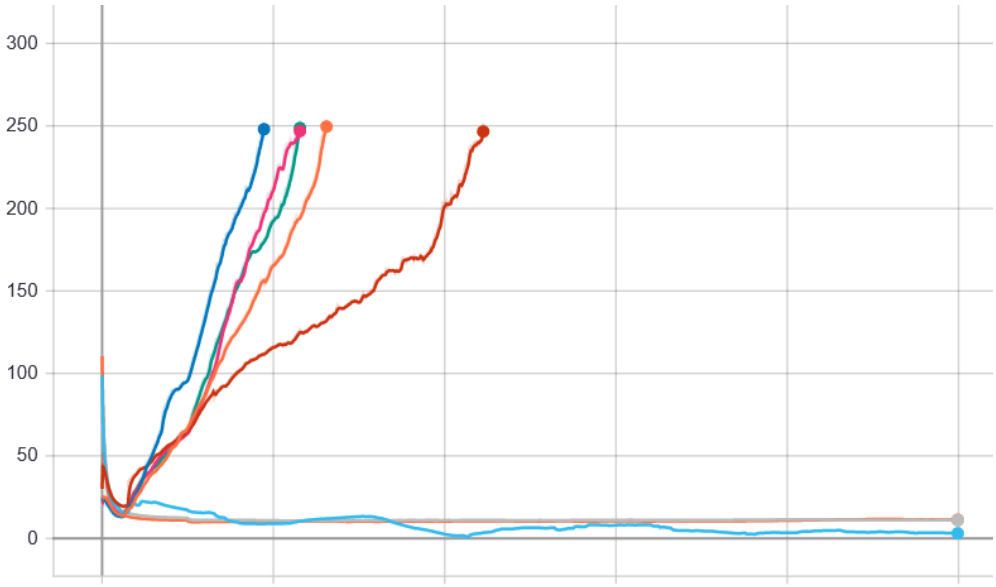
3.2 SAC

Dla tego algorytmu przetestowano następujące wartości hiperparametrów:

- liczba neuronów = [128, 256]
- liczba warstw = [2, 3]
- learning rate = [0.001, 0.0005]

Testy każdej architektury trwały przez 1000 epizodów lub do czasu, gdy średnia nagroda uzyskana przez agenta we wszystkich epizodach była większa niż 250.

Wyniki testów różnych architektur dla algorytmu SAC zostały przedstawione za pomocą TensorBoarda: `https://tensorboard.dev/experiment/uoS11jWPQpqToM728tJsJQ/#scalars`. Przedstawiają się one następująco.



Rysunek 4: Wyniki testów różnych architektur dla algorytmu SAC

Liczba warstw	Liczba neuronów	Współczynnik uczenia	Średnia nagroda	Liczba epizodów
2	128	0.0005	254	262
2	128	0.001	252.3	189
3	128	0.0005	251.6	445
3	128	0.001	3.158	1000
2	256	0.0005	251.5	231
2	256	0.001	253.5	231
3	256	0.0005	11.11	1000
3	256	0.001	11.3	1000

Tablica 2: Wyniki testów różnych architektur dla algorytmu SAC

Wyraźnie widać, że najlepsze wyniki uzyskała sieć z 2 warstwami po 128 neuronów i współczynnikiem uczenia równym 0.001. Uzyskała ona wynik progowy w 189 kroków. Warto zauważyć, że trzy spośród testowanych architektur uzyskały bardzo słabe wyniki. Agent nie był w stanie dokonywać lepszych kroków. Odzwierciedla to wartość uzyskanej średniej nagrody w procesie uczenia.

Najlepsza architektura została potem wykorzystana w procesie uczenia agenta. Zostało wykonanych 1000 epizodów, po których średnia nagroda wynosiła około 580. Niestety po symulacji Walker stał w miejscu i nie wykonywał kroków. Zostały podjęte próby zwiększenia dwukrotnie nagrody za przebyty dystans, zmniejszenia dwukrotnie nagrody za przeżycie, i zwiększenia kary za oszczędzenie energii razy 1.2. Nagroda pozostawała w tych samych okolicach lecz symulacja dalej się nie poprawiała.

Dla podwójnej dyskretyzacji czasu algorytm wypadał podobnie i uzyskał średnią nagrodę 550 po 616 krokach.

4 Wnioski

Algorytm SAC zadziałał lepiej niż algorytm DDPG dla środowiska testowego *Walker 2D*. Agent uczył się szybciej i średnio uzyskiwał znacznie większe nagrody. Proces nauki był również bardziej stabilny. Oprócz wyraźnego braku uczenia dla pewnych architektur sieci neuronowych, średnia nagroda stale rosła, co widać dobrze na załączonym wykresie.

Jest to dosyć spodziewany wynik. Algorytm SAC jest niejako połączeniem dwóch grup algorytmów i tym samym nie posiada tych samych wad czy słabych punktów co np. algorytm DDPG. SAC jest obecnie powszechnie wykorzystywanym algorytmem w różnych problemach uczenia ze wzmocnieniem, również w środowiskach testowych z zestawu *Gym*.

Jednak po stworzeniu symulacji można zauważyć, że jak model dla SAC stoi w miejscu to najlepszy model DDPG sprawnie się porusza. Jest to pokłosie zbytniej stabilizacji uczenia gdzie SAC dobrze się uczy zdobywać duże średnie nagrody, lecz miał problemy z eksploracją rozwiązania.

Po zastosowaniu podwójnej dyskretyzacji czasu nie zauważono znaczących różnic w wynikach.

5 Bibliografia

[1] Walker2d-v2, *https://gym.openai.com/envs/Walker2d-v2/*

[2] PyBullet-Gym, *https://github.com/benelot/pybullet-gym*

[3] *https://github.com/abhisheksuran/Reinforcement_ Learning*

[4] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, Daan Wierstra *Continuous control with deep reinforcement learning*, ArXiV, vol. 1509.02971 (2017)

[5] Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., Levine, S., *Soft Actor-Critic Algorithms and Applications*, ArXiV, vol. 1812.05905 (2019)