Dawid Woźniak
Informatyka - Uczenie Maszynowe I Sztuczna Inteligencja

Uczenie ze Wzmocnieniem

Algorytm Aktor-Krytyk

1. Opis ćwiczenia

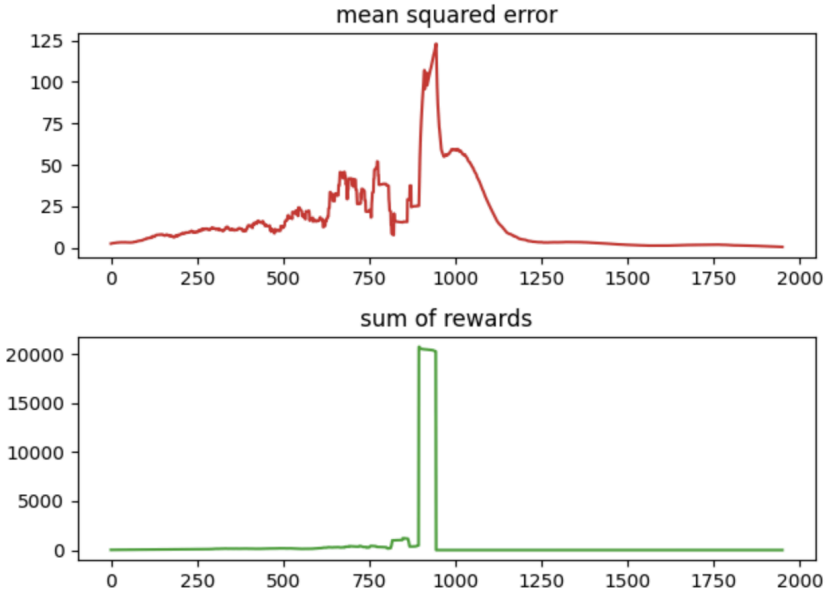
Celem laboratorium było dokończenie implementacji algorytmu Aktor-Krytyk a następnie przetestowanie go w rozwiązywaniu dwóch epizodycznych problemów - balansowania tyczką na ruchomym wagoniku (CartPole) oraz bezpiecznego dotarcia lądownikiem na powierzchnię planety (LunarLander). Początkowo szkielet rozwiązania był zaimplementowany w bibliotece Tensorflow, jednak zdecydowałem się zmigrować do go biblioteki Pytorch.

2. Wyniki

Dla każdego poniższego przykładu, parametr discount factor miał wartość $\gamma = 0.99$

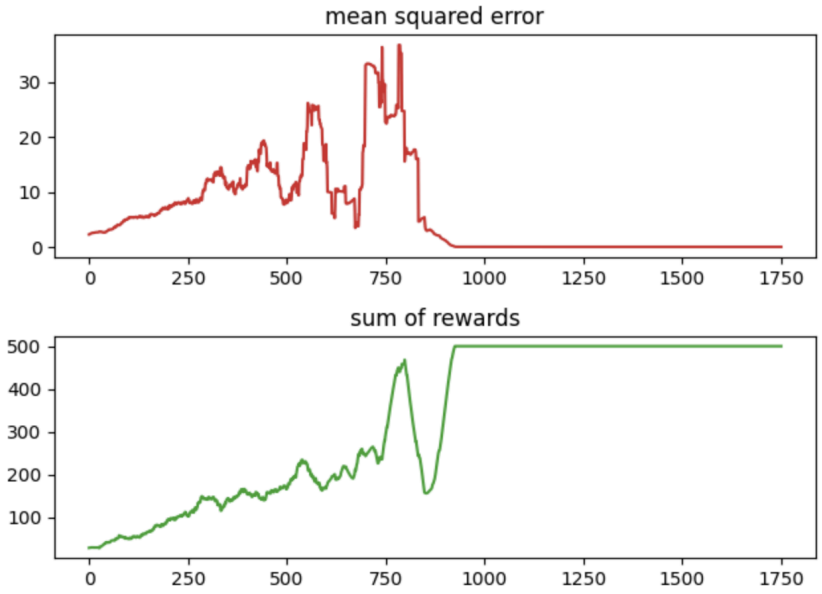
2.1 Balansowanie Tyczki

Po udanej migracji kodu do Pytorcha, uzupełniłem brakującą logikę oraz stworzyłem sieć neuronową o zadanej architekturze i przystąpiłem do treningu mojego rozwiązanie dla problemu balansowania tyczki. W tej części aktor i krytyk posiadały wspólny początek sieci. Wyniki dla wariantu z parametrami $h_1 = 1024$, $h_2 = 256$, $\alpha = 0.00001$ prezentuje rysunek 1.



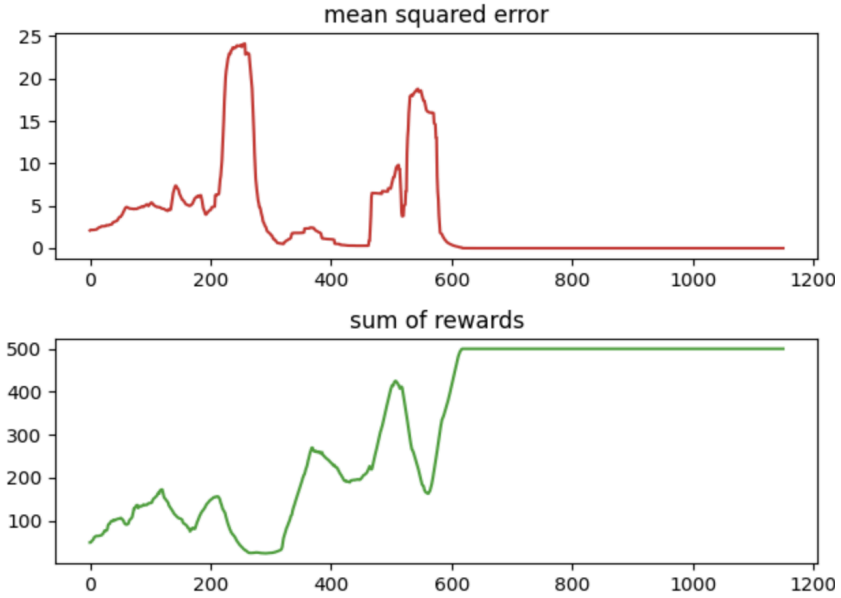
Rys. 1. Przebieg uczenia dla pierwszego modelu.

Jak widać, w pewnym momencie algorytm zaczął utrzymywać pionowo kijek przez bardzo długi czas bo przez około 20 tysięcy kroków. Niestety po pewnym czasie rozwiązanie się zdewaluowało i model stracił umiejętność balansowanie kijkiem. By uporać się z taką sytuacją, zacząłem wykorzystywać parametr $truncated$ zwracany przez środowisko. Parametr ten sugeruje koniec epizodu, mimo że stan terminalny nie został jeszcze osiągnięty. Dla tego problemu było to 500 kroków.



Rys. 2. Przebieg uczenia dla drugiego modelu.

Jak widać dla tej konfiguracji algorytm nauczył się balansować kijek przez 500 kroków w około 900 epizodzie i utrzymywał ten poziom nagrody do końca uczenia.
Kolejnym krokiem było rozdzielenie wspólnego początku aktora i krytyka. Od teraz były to 2 całkowicie osobne sieci neuronowe, o tej samej zadanej architekturze.

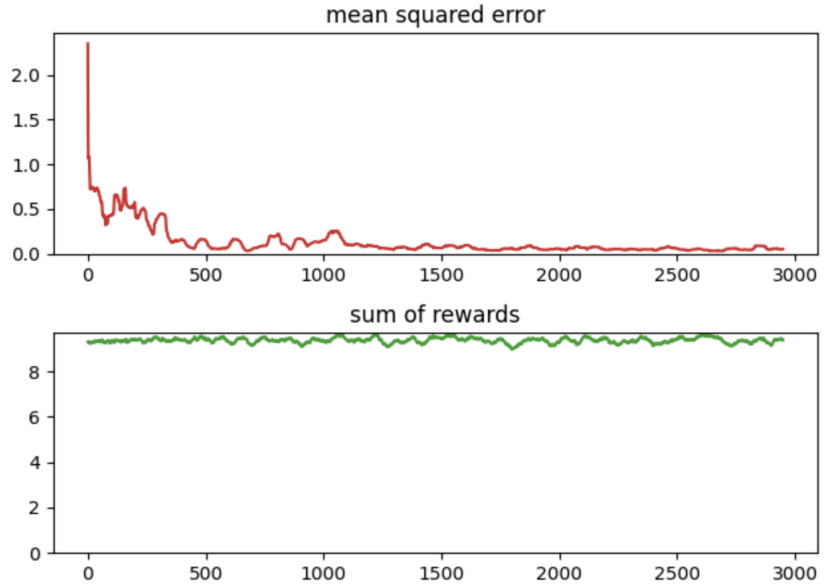


Rys. 3. Przebieg uczenia dla trzeciego modelu.

Dla wersji z osobnymi wagami dla aktora i krytyka, algorytm szybciej nauczył się rozwiązywać problem bo w około 600 epizodzie. Obarczone zostało to większą amplitudą zmian wartości nagród i MSE w procesie uczenia. Wydaje się, że system zachowuje się mniej stabilnie. Można jeszcze zauważyć zależność między błędem wartościowania a otrzymywanymi nagrodami. Po pewnym czasie błąd wartościowania przypomina symetryczne odbicie względem osi X otrzymywanych nagród. Dla małych nagród błąd jest duży a dla dużych mały. Dla wyuczonego modelu, przetestowałem jak krytyk wartościuje wybrane stany:

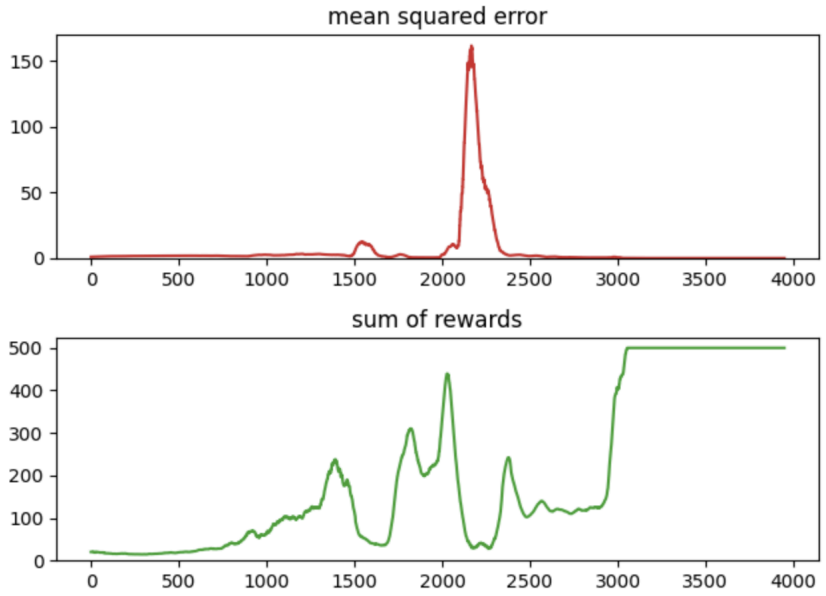
- Kijek stojący pionowo - $V([0, 0, 0, 0]) = 100$
- Kijek zaczynający się szybko przechylać - $V([0, 0, 0.1, 5]) = 56$
- Wagonik znajdujący się blisko krawędzi - $V([4.3, 0, 0, 0]) = 57$

Jak widać, krytyk najlepiej wartościuje najlepszy stan czyli gdy kijek stoi pionowo a prędkości są zerowe. Dla zadanej architektury przetestowałem konfigurację ze zwiększonym krokiem uczenia tj. $\alpha = 0.1$.



Rys. 4. Przebieg uczenia dla czwartego modelu.

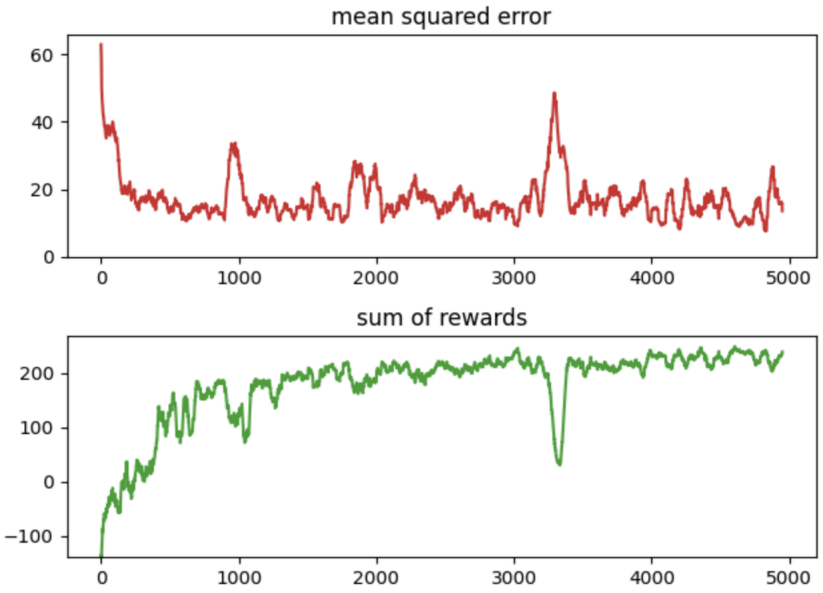
Dla takiego kroku, model nie był w stanie niczego się nauczyć. Wynika to z tego, że metoda aktor krytyk jest bardzo niestabilna i żeby osiągnąć zadowalające wyniki musimy stosować mały krok uczenia. Następnie wróciłem do poprzedniej stałej uczenia i przetestowałem sieć o mniejszym rozmiarze tj. $h_1 = 128,\ h_2 = 32,\ \alpha = 0.00001$. Model ten był w stanie się nauczyć rozwiązywać problem balansowania kijka, lecz zajęło mu to znacznie więcej epizodów bo około 3000 a sam proces uczenia wydaje się być najmniej stabilny.



Rys. 5. Przebieg uczenia dla piątego modelu.

2.2 Lądownik

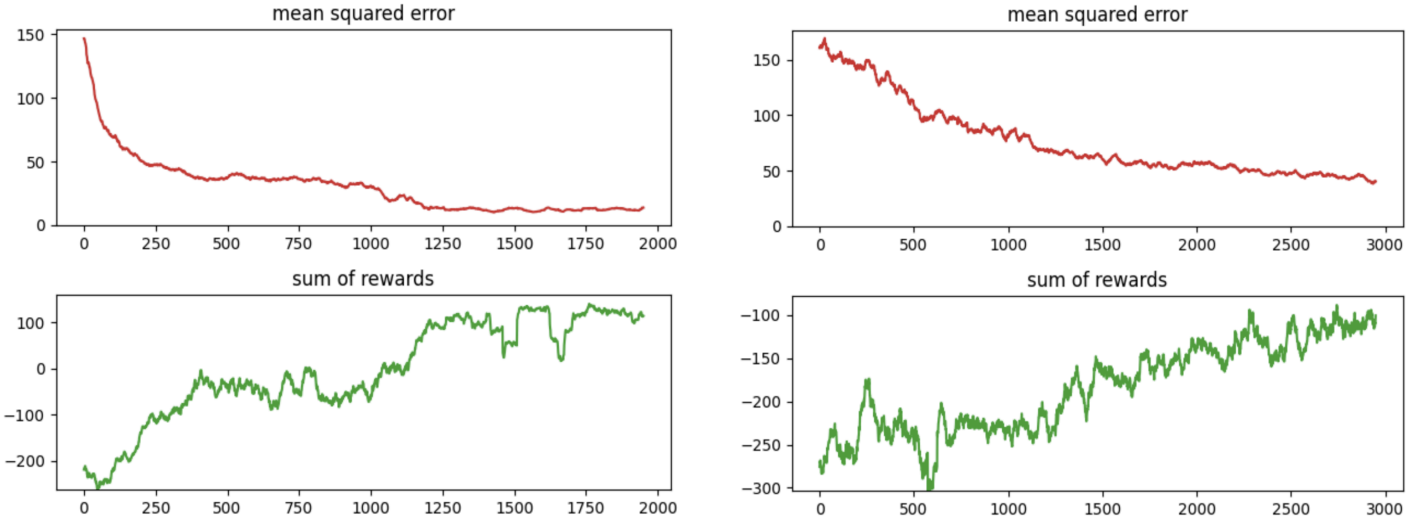
Dla tego przykładu wybrałem model, który moim zdaniem dał najlepsze efekty dla problemu z kijkiem a więc większe osobne sieci neuronowe dla aktora i krytyka z małym krokiem uczącym tj. $h_1 = 1024$, $h_2 = 256$, $\alpha = 0.00001$.



Rys. 6. Przebieg uczenia dla pierwszego modelu.

Taka konfiguracja, była w stanie osiągnąć wartość nagród około 230 co wskazuje na poradzenie sobie z problemem. Proces uczenia trwał dłużej niż dla wariantu z kijkiem co wynika z większej złożoności modelu. Sprawdziłem również jak radzi sobie sieć dla

mniejszych stałych uczących $\alpha = 0.000001$, $\alpha = 0.0000001$. Dla pierwszej wartości model uczył się wolniej a dla drugiej znacząco wolniej.



Rys. 7. Przebieg uczenia dla drugiego i trzeciego modelu.

Po obejrzeniu animacji lądowania stwierdzam, że model stara się umiejscowić rakietę na wysokości obszaru między chorągiewkami a potem powoli tam ją opuścić. Główny silnik jest uruchomiony przez większośc czasu co zapewnia powolne opadanie i brak rozbicia się. Krytyk w następujący sposób wartościuje wybrane stany:

- Chwila przed lądowaniem - $V([0,\ 0.01,\ 0, -\ 0.5, 0, 0, 0, 0]) = 87$
- Statek mocno przechylony w lewo - $V([0,\ 0,\ 0,\ 0.5,\ -\ 6,\ 0,\ 0,\ 0]) = 54$
- Statek mocno przechylony w prawo - $V([0,\ 0,\ 0,\ 0.5,\ 6,\ 0,\ 0,\ 0]) = 53$

Krytyk najlepiej wartościuje najlepszy stan czyli pozycję chwilę przed samym lądowaniem. Dla statku mocno przechylonego w lewo, polityka zaproponowała akcję 1 czyli włączenie lewego silnika, a dla statku przechylonego w prawo akcję 3 czyli włączenie prawego silnika. Jest to poprawne zachowanie polityki.

3. Wnioski

Udało się poprawnie dokończyć implementację algorytmu Aktor-Krytyk co potwierdziło przetestowanie go na dwóch środowiskach w bibliotece gymnasium. Algorytm ten jest bardzo niestabilny co skutkuje tym, że musimy używać bardzo małej stałej uczącej przy jego treningu. Ważną wskazówką okazało się regularne zapisywanie modelu po poszczególnych krokach uczenia, gdyż zdarzało się, że rozwiązanie się zdewaluowało.