Inteligencja obliczeniowa w analizie danych cyfrowych

Projekt V

Autorzy projektu:

Natalia Kwiecień
Oliwier Maj

EAIiIB
Katedra Informatyki Stosowanej
Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie

Kraków, 31 maja 2025

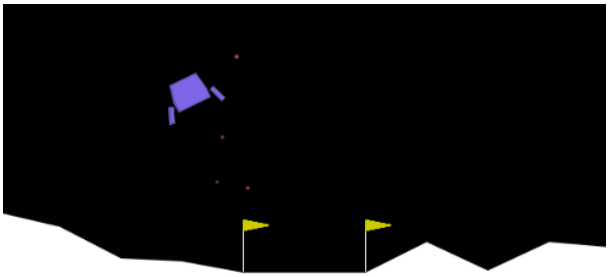
Spis treści

1 Wstęp i cel projektu

Celem projektu jest przetestowanie zastosowania biblioteki Stable Baseline 3 dla uczenia ze wzmacnianiem w ciągłych środowiskach na podstawie jednego ze środowisk Gymnasium. Dla wybranego algorytmu zostaną przeprowadzone eksperymenty z różnymi parametrami modelu, a także różnymi architekturami sieci. Ostatnim krokiem będzie sprawdzenie działania modelu przy wyłączonym trybie eksploracji.

2 Środowisko LunarLanderContinous-v3

W tym zadaniu agent steruje lądownikiem księżycowym, którego celem jest bezpieczne lądowanie na wyznaczonej platformie znajdującej się na środku planszy.



Rysunek 1: LunarLanderContinous-v3 - problem

2.1 Przestrzeń obserwacji

Środowisko opisane jest za pomocą ośmiu zmiennych:

- Pozycja lądownika (x, y),
- Prędkość lądownika (x, y),
- Kąt nachylenia,
- Prędkość kątowa,
- Czy lewa noga dotyka podłoża (bool),
- Czy prawa noga dotyka podłoża (bool).

2.2 Możliwe akcje

Przestrzeń akcji jest ciągła - agent może sterować mocą silnika głównego i bocznego. Oznacza to, że może w każdym kroku czasowym dobrać:

- Siłę silnika głównego (wartość z zakresu $[-1, 1]$) - odpowiada za ruch w pionie,
- Siłę silnika bocznego (wartość z zakresu $[-1, 1]$) - odpowiada za rotację lądownika.

2.3 Nagrody

Nagrody przydzielane są za następujące składowe:

- Zmniejszanie odległości do lądowiska,
- Utrzymywanie lądownika w pozycji pionowej,
- Kontakt nóg z powierzchnią (nagroda),
- Bezkontaktowe zetknięcie z powierzchnią (kara),
- Użycie silników (lekka kara za zużycie paliwa).

Jeśli agent wyląduje w obrębie lądowiska i prędkości oraz kąt są odpowiednio małe – otrzymuje dużą nagrodę za poprawne lądowanie.

2.4 Pozycja początkowa

Pozycja początkowa lądownika jest losowa, zarówno pod względem położenia, jak i orientacji.

2.5 Koniec epizodu

Epizod kończy się w jednym z przypadków:

1. Lądownik wyląduje (z sukcesem lub katastrofą),
2. Liczba kroków przekroczy 1000.

3 Algorytm SAC

Wybranym przez nas algorytmem jest Soft Actor-Critic (https://spinningup.openai.com/en/latest/algorithms/sac.html). Jego kluczową cechą jest utrzymywanie wysokiej entropii - losowości/eksploracji środowiska.

Algorytm wykorzystuje trzy sieci neuronowe:

1. Aktora, który przewiduje akcje (w rzeczywistości, ponieważ algorytm jest stworzony dla środowisk ciągłych, uczy się rozkładu prawdopodobieństwa ruchów). Na wejściu sieci znajduje się aktualny stan środowiska, a na wyjściu parametry rozkładu akcji (z którego losowana jest akcja);
2. Dwóch krytyków (sieci Q), którzy oceniają jak dobra jest akcja w danym stanie. Na wejściu znajduje się para (obserwacja, akcja), natomiast na wyjściu wartość Q, czyli szacowana jakość danej akcji w danym stanie.

Dodatkowo wykorzystywany jest bufor doświadczeń (Replay Buffer), w którym agent zapisuje swoje doświadczenia, z których później się uczy (może robić wiele aktualizacji na podstawie jednego stanu).

Algorytm działa dość wolno, jednak zapewnia to jego skuteczność - wynika to z tego, że maksymalizuje nie tylko nagrody, ale też entropię. Eksploruje różne możliwości, unikając szybkiego przywiązania do jednej strategii.

4 Strojenie hiperparametrów

Ze względu na to, że środowiska z biblioteki Gym - w tym LunarLanderContinuous-v3 - są dobrze znane i często wykorzystywane w badaniach nad uczeniem ze wzmocnieniem, można znaleźć gotowe modele dostosowane do problemu, wraz z ich parametrami. Przykładem takiego źródła jest repozytorium Stable-Baselines3 Zoo (https://github.com/DLR-RM/rl-baselines3-zoo).

Dla środowiska LunarLanderContinuous-v3, parametry te są następujące:

```
LunarLanderContinuous-v3:
  n_timesteps: !!float 5e5
  policy: 'MlpPolicy'
  batch_size: 256
  learning_rate: lin_7.3e-4
  buffer_size: 1000000
  ent_coef: 'auto'
  gamma: 0.99
  tau: 0.01
  train_freq: 1
  gradient_steps: 1
  learning_starts: 10000
  policy_kwargs: "dict(net_arch=[400, 300])"
```

Gdzie:

- **n_timesteps** – całkowita liczba kroków;
- **policy** – typ użytej sieci neuronowej;
- **policy_kwargs** – architektura sieci neuronowej;
- **batch_size** – liczba próbek w pojedynczej aktualizacji sieci;
- **learning_rate** – współczynnik uczenia;
- **learning_starts** – liczba kroków, po których model zaczyna się uczyć (wcześniej tylko zbiera doświadczenie);
- **ent_coef** – współczynnik entropii;
- **gamma** – współczynnik promujący odległe nagrody;
- **tau** – parametr kontrolujący szybkość aktualizacji sieci docelowej;
- **train_freq** – częstotliwość trenowania modelu;
- **gradient_steps** – liczba aktualizacji gradientu po każdym kroku.

4.1 Eksperymenty z hiperparametrami

Aby ocenić wpływ wybranych hiperparametrów na proces uczenia agenta, zdecydowaliśmy się przeprowadzić trzy eksperymenty z różnymi parametrami. Zdecydowaliśmy się na użycie domyślnych parametrów ze zmianą jedynie learning rate i batch size.

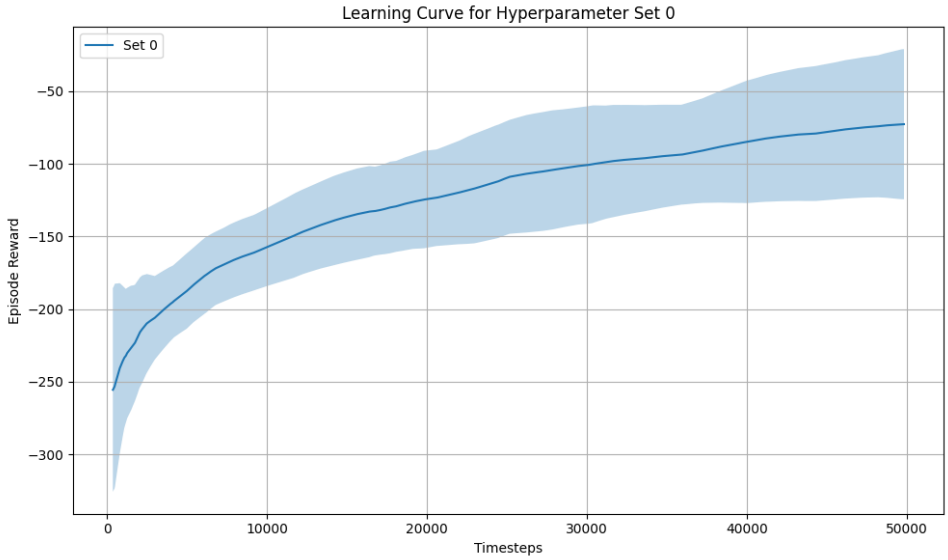
Zastosowane zestawy hiperparametrów są następujące:

1. learning_rate = 3e-4, batch_size = 256
2. learning_rate = 7.3e-4, batch_size = 256
3. learning_rate = 1e-3, batch_size = 128

Każdy eksperyment został powtórzony dziesięć razy w 50 000 krokach, aby umożliwić obliczenie średniego wyniku oraz odchylenia standardowego.

4.1.1 Eksperyment 1

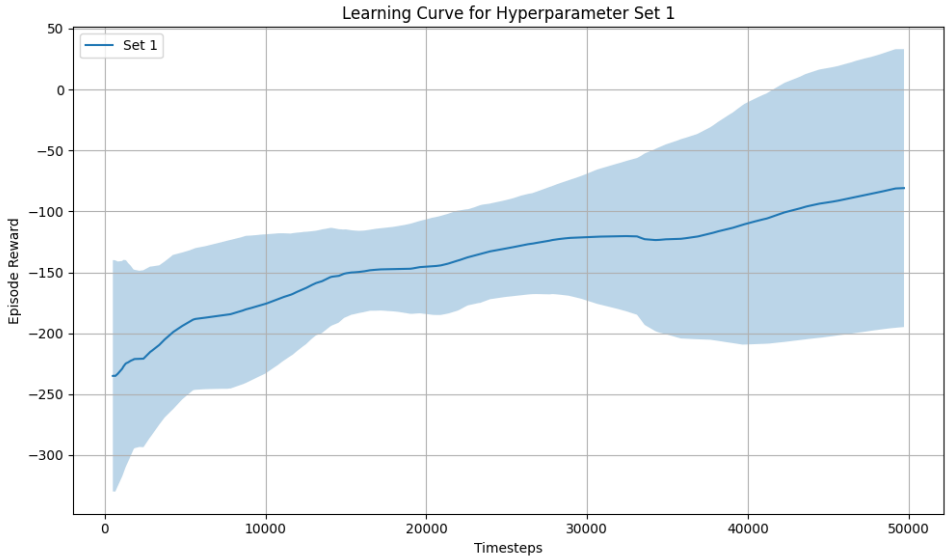
Poniżej znajduje się krzywa uczenia dla parametrów learning_rate = 3e-4, batch_size = 256:



Rysunek 2: learning_rate = 3e-4, batch_size = 256

4.1.2 Eksperyment 2

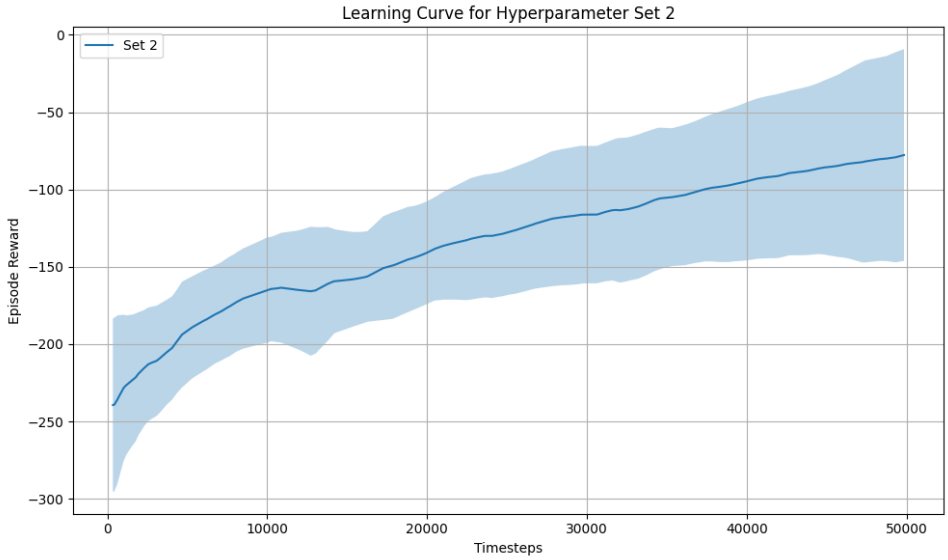
Poniżej znajduje się krzywa uczenia dla parametrów learning_rate = 7.3e-4, batch_size = 256:



Rysunek 3: learning_rate = 7.3e-4, batch_size = 256

4.1.3 Eksperyment 3

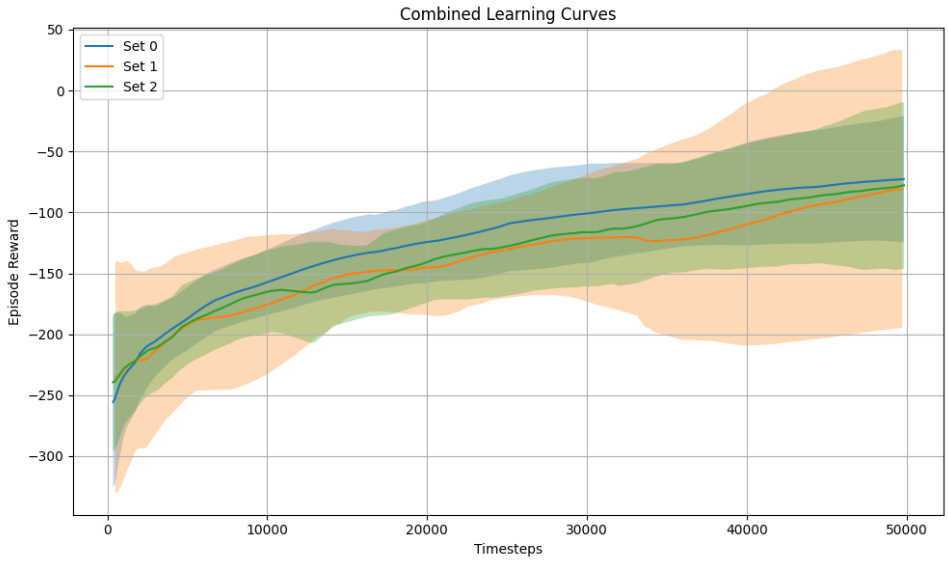
Poniżej znajduje się krzywa uczenia dla parametrów learning_rate = 1e-3, batch_size = 128:



Rysunek 4: learning_rate = 1e-3, batch_size = 128

4.1.4 Porównanie

Poniżej znajduje się porównanie wszystkich eksperymentów:

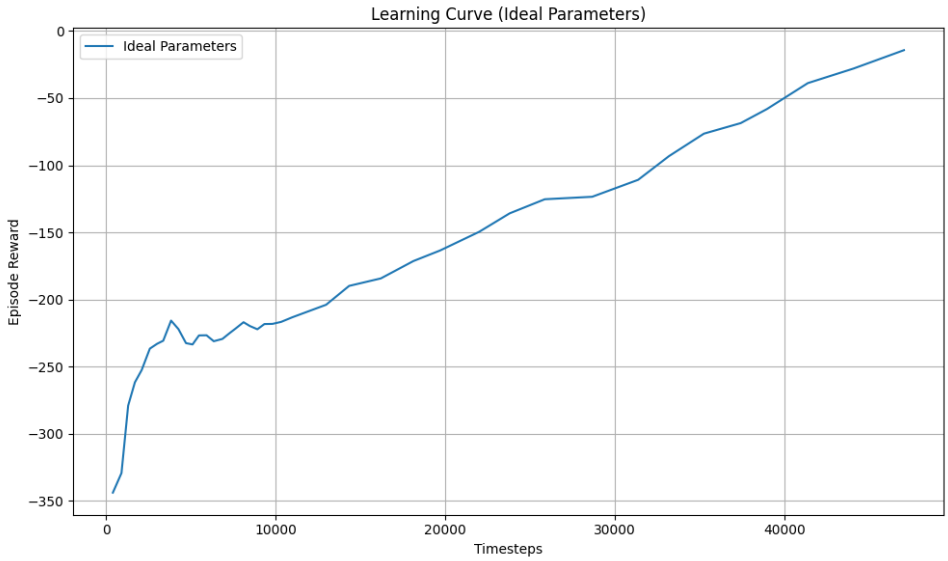


Rysunek 5: Porównanie wyników

Wszystkie eksperymenty osiągnęły podobny wynik, jednak dla drugiego eksperymentu (gdzie współczynnik uczenia był znacznie wyższy) odchylenie standardowe jest większe - oznacza to, że wyniki uzyskiwane przez agenta były mniej stabilne. Może to oznaczać, że zbyt duży współczynnik uczenia prowadzi do zbyt dużej zmienności. Jednocześnie jest to wartość zalecana przez Stable Baseline3 Zoo - zły wynik w naszym eksperymencie wynika zapewne z tego, że inne hiperparametry nie były do niego dostosowane.

4.2 Zalecany model z Stable Baseline3 Zoo

Poniżej znajduje się krzywa uczenia dla zalecanego modelu w jednym eksperymencie:



Rysunek 6: Parametry zalecane przez Stable Baseline3 Zoo

Osiągnięto tutaj wyższy wynik niż w większości naszych eksperymentów.

5 Architektury sieci

W celu zbadania wpływu architektury sieci neuronowej na jakość uczenia, postanowiliśmy przeprowadzić eksperyment z rekomendowanymi parametrami, w których zmieniona jest jedynie architektura sieci.

W bibliotece Stable-Baselines3 architekturę sieci ustala się poprzez przekazanie odpowiednich parametrów do argumentu policy_kwargs w momencie tworzenia modelu (https://stable-baselines3.readthedocs.io/en/master/_modules/stable_baselines3/sac/policies.html).

Budowa sieci w tym algorytmie została opisana w sekcji *Algorytm SAC*.

5.1 Architektura zalecana przez Stable Baseline3 Zoo

W przypadku modelu bazowego wykorzystano sieć typu MLP (ang. Multi-Layer Perceptron) o następującej strukturze:

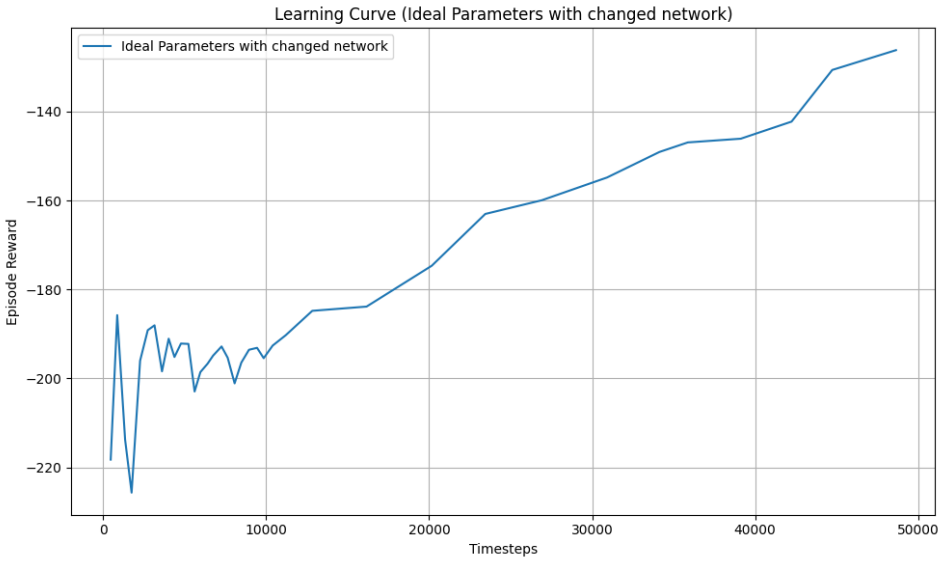
- 2 warstwy ukryte:
  - pierwsza warstwa: 400 neuronów,
  - druga warstwa: 300 neuronów,
- funkcja aktywacji: ReLU.

5.2 Eksperyment dla zmienionej architektury przy zachowaniu innych parametrów

Aby przetestować alternatywną strukturę, zmieniliśmy wyłącznie architekturę sieci, pozostawiając wszystkie inne hiperparametry bez zmian. W nowym wariancie wykorzystano:

- 3 warstwy ukryte:
  - pierwsza warstwa: 256 neuronów,
  - druga warstwa: 256 neuronów,
  - trzecia warstwa: 128 neuronów,
- funkcja aktywacji: Tanh.

Uzyskaliśmy następującą krzywą uczenia:



Rysunek 7: Krzywa uczenia dla zmienionej architektury sieci

Uzyskaliśmy znacznie gorszy wynik. Wartość nagrody nie przekroczyła nawet -130 punktów, kiedy bez zmiany architektury wartość ta była bliska 0. Oznacza to, że budowa sieci ma znaczny wpływ na działanie całego algorytmu.

6 Wyłączenie trybu eksploracji

Następnie przetestowaliśmy rekomendowany model w symulacji z wyłączonym trybem eksploracji. W 10 eksperymentach uzyskaliśmy średnią 86.2. Podczas uczenia ani razu nie przekroczono zera - wynika to z tego, że model specjalnie popełniał błędy, aby eksplorować możliwe strategie. Tutaj w każdym kroku model wykonywał jego zdaniem najlepszy ruch. Dlatego właśnie podczas nauki oraz działania modelu (szczególnie w trybie deterministycznym) wartości te są tak różne.