

Algorytmy ewolucyjne

Neuroewolucja

Projekt zaliczeniowy

Michał Maras, Michał Kępa

11 lutego 2021

1 Wprowadzenie

2 Środowiska

3 Algorytmy

Wprowadzenie

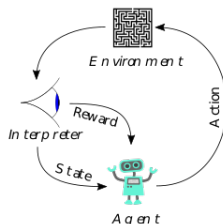
Celem projektu jest wykorzystanie algorytmów ewolucyjnych jako sposobu trenowania sieci neuronowych w problemach uczenia przez wzmacnianie.

Rozważone zostały następujące problemy:

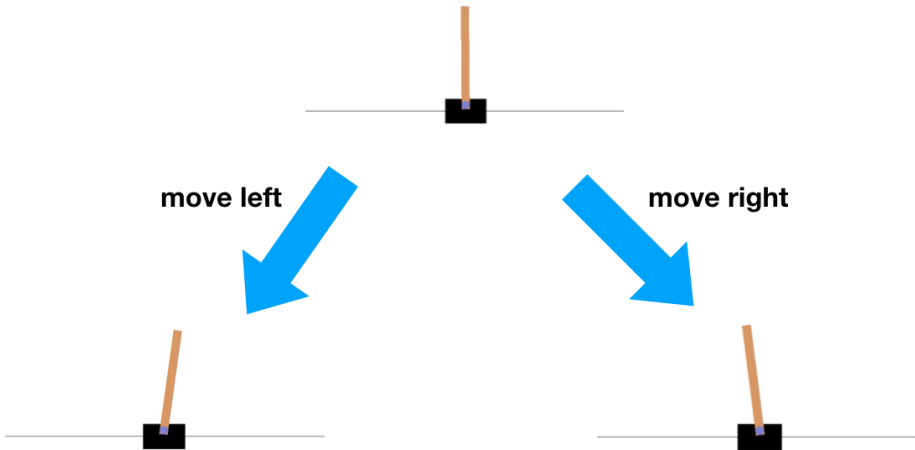
- CartPole – Open AI Gym Benchmark
- gra Pong – Open AI Atari Gym oraz własna implementacja
- gra 2048 – własna implementacja
- gra CarRacing – własna implementacja
- gra Beamrider – Open AI Atari Gym

Uczenie przez wzmacnianie

(ang. *Reinforcement Learning*) – rodzaj uczenia maszynowego, którego zadaniem jest interakcja ze środowiskiem. W uczeniu przez wzmacnianie celem modelu jest wykonywanie odpowiednich akcji na podstawie danych ze środowiska, za które agent otrzyma jak największą nagrodę.



Środowiska



CartPole

Problem

Środowisko zawiera wózek i przymocowany do niego maszt. W stanie początkowym maszt jest położony prawie pionowo, a w wyniku oddziaływania grawitacji zaczyna coraz bardziej przechylać się na jedną stronę. Agent ma za zadanie przesunąć wózek w prawo lub lewo z odpowiednią siłą, aby utrzymać słupek jak najdłużej w pionie. Gdy masz się przewrócić – gra się kończy.

Obserwacja

Agent otrzymuje informacje o pozycji oraz prędkości wózka i masztu.

Nagroda

Nagroda za grę zależy od długości jej trwania. Dodatkowo, w każdej klatce naliczana jest kara zależna od odległości wózka od środka planszy oraz odchylenia masztu od pionu.

2

Q



Pong



Problem

Typowa implementacja gry pong: jest dwóch graczy, którzy sterują paletkami i odbijają piłeczkę. W przypadku własnej implementacji gra trwała 10 rund, a w wersji OpenAI do 21 wygranych rund.

Obserwacja

Agent dostaje stan złożony z pozycji paletek oraz pozycji i prędkości piłki. Inną wersją obserwacji był stan pamięci RAM emulatora (128 bajtów)

Nagroda

Nagroda to wynik gry (liczba zdobytych punktów) z bonusem za wygraną, powiększona (w przypadku przegranej) lub pomniejszona (w przypadku wygranej) o czas trwania gry. Sprawdzona została też modyfikacja, gdzie wynik zależy od odległości paletki od piłki w momencie zdobycia lub straty punktu.

16

4

128

16

32

64

2

16

8

8

2

2048

Problem

Implementacja podobna do oryginalnej wersji, ale w przypadku wykonania niedozwolonego ruchu gra natychmiast się kończy. Gracz ma możliwość przesunąć wszystkie bloczki na planszy w określonym kierunku. Gdy dwa bloczki z takim samym numerem się spotkają, łączą się w jeden bloczek o dwa razy większej wartości.

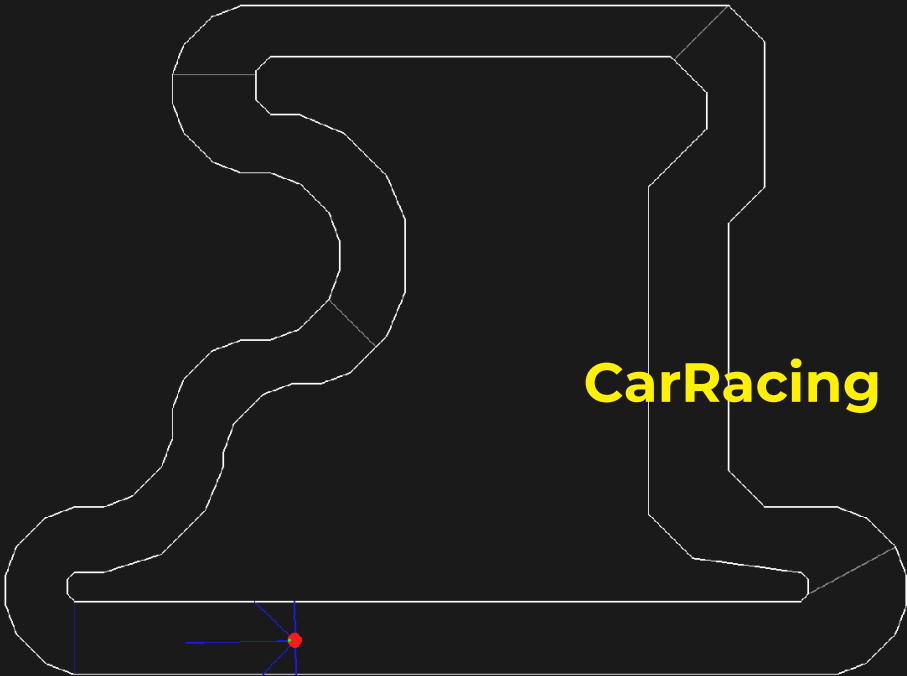
Obserwacja

Agent dostaje stan złożony z logarytmów wartości kolejnych kafelków na planszy (lub 0 w przypadku pustego pola).

Nagroda

Nagroda to suma wartości kafelków na planszy.

CarRacing



Problem

Środowisko składa się z samochodu, toru oraz listy punktów kontrolnych, przez które kolejno musi przejechać agent. Gdy gracz wjedzie w ścianę – przegrywa.

Obserwacja

Agent dostaje stan złożony z odległości do najbliższej przeszkody w pięciu różnych kierunkach.

Nagroda

Nagroda to długość przejechanej trasy.

00000000
SECTOR 00



Beamrider



ACTIVISION

Problem

Środowiskiem jest symulacja gry Beamrider z konsoli Atari 2600.

Obserwacja

Agent dostaje stan pamięci RAM emulatora (128 bajtów).

Nagroda

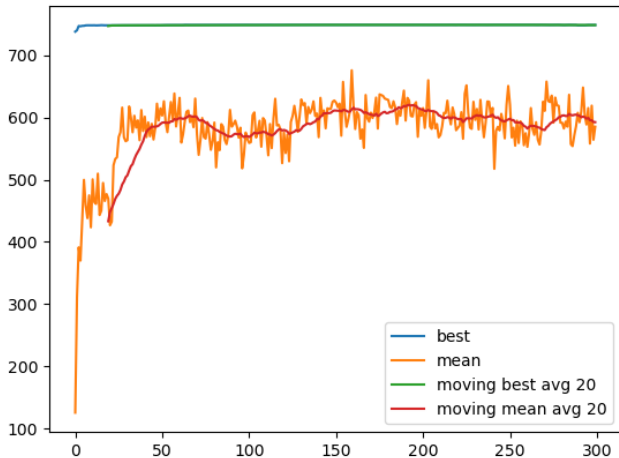
Liczba punktów zdobytych w grze, która zależała od długości gry, liczby pokonanych przeciwników i liczby zakończonych etapów.

Do neuroewolucji zastosowaliśmy następujące algorytmy:

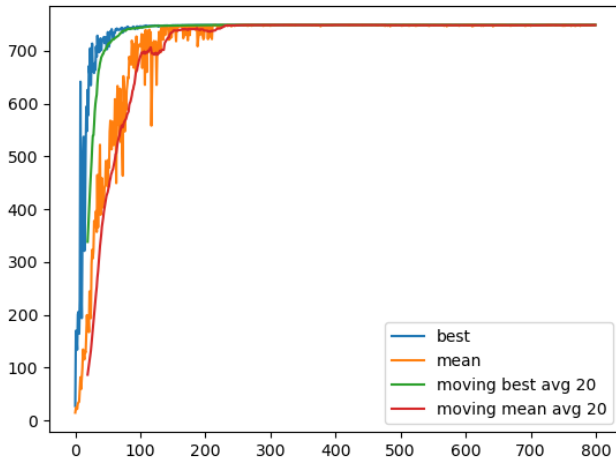
- CMA-ES,
- NEAT,
- LM-MA-ES.

Rezultaty

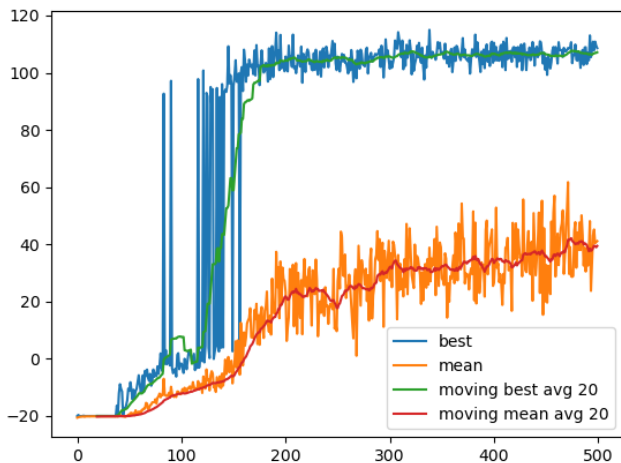
CartPole NEAT



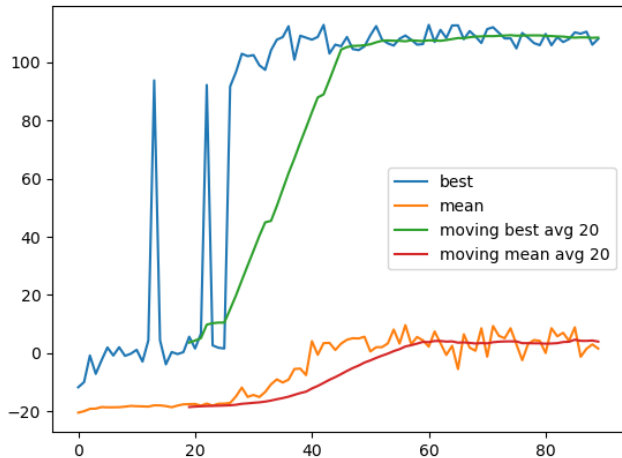
CartPole CMA-ES



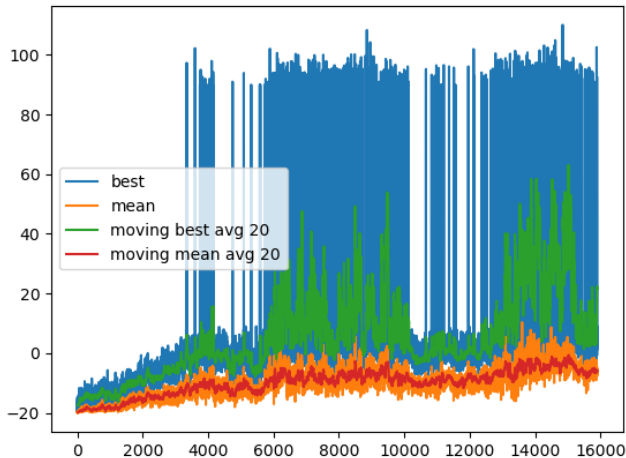
Pong CMA-ES



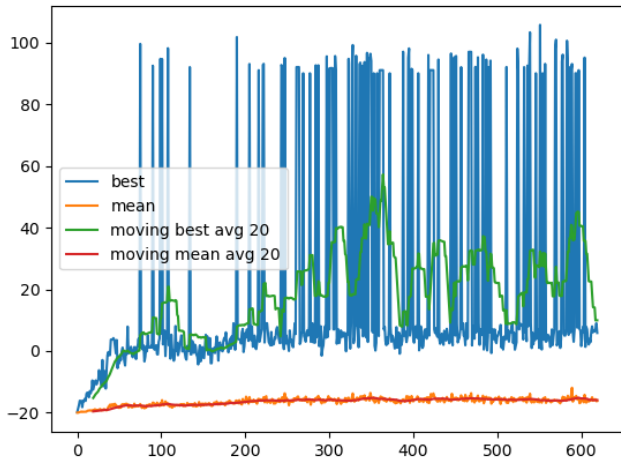
Pong NEAT



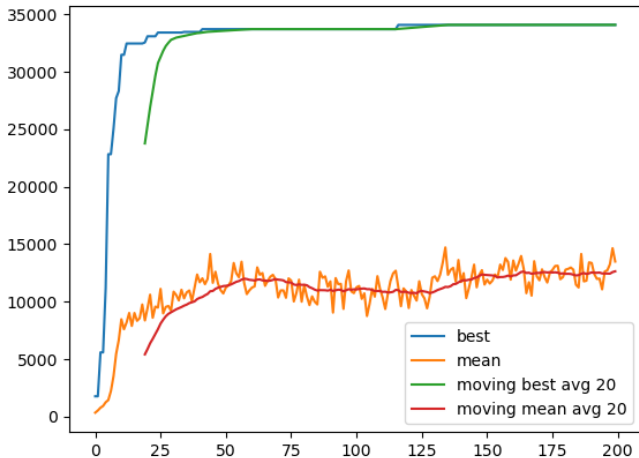
Pong LM-MA-ES (RAM)



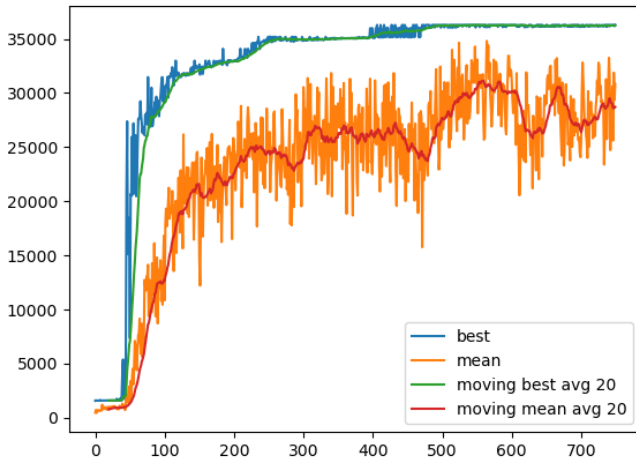
Pong NEAT (RAM)



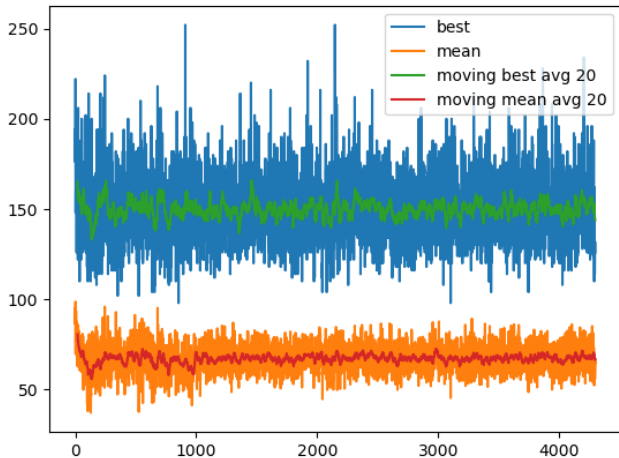
CarRacing NEAT



CarRacing CMA-ES



2048 NEAT



2048 LM-MA-ES

