Bootcamp Lvl Up

Wstęp do uczenia ze wzmocnieniem - Jakub Łyskawa

Przygotowanie środowiska

Pobranie/instalacja SWIG

Pobranie pakietów Python:

- gym
- pyglet
- box2d-py
- tensorflow-probability (1.14.1 dla tf 2.6)

numpy, tensorflow, scipy...

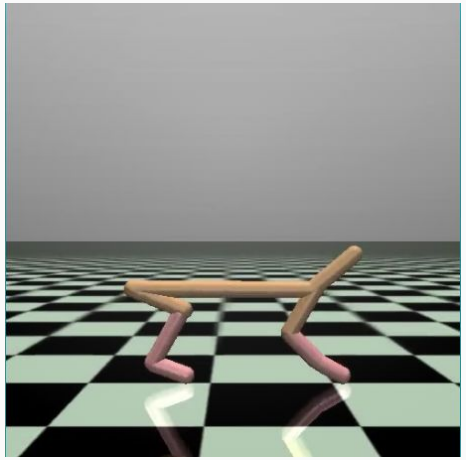
Test:

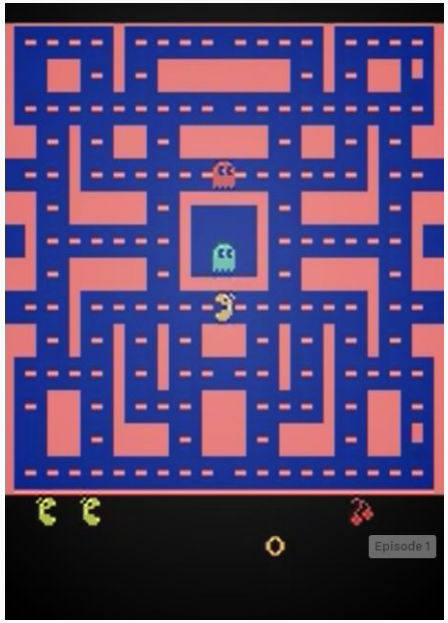
```
import gym
import tensorflow_probability
env = gym.make('LunarLander-v2')
env.render()
env.close()
```

Uczenie ze wzmocnieniem (Reinforcement Learning, RL)

Problemy rozwiązywane przez RL

- Sterowanie
- Decyzje biznesowe
- Boty do gier
- Czatboty



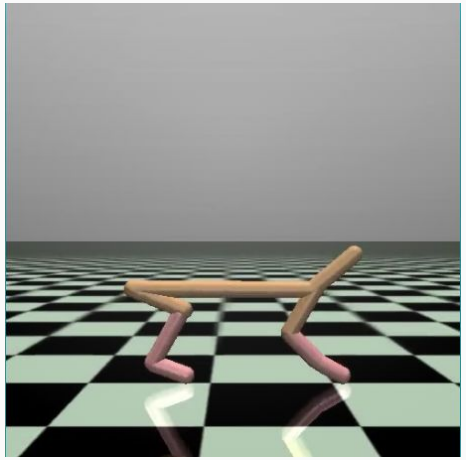


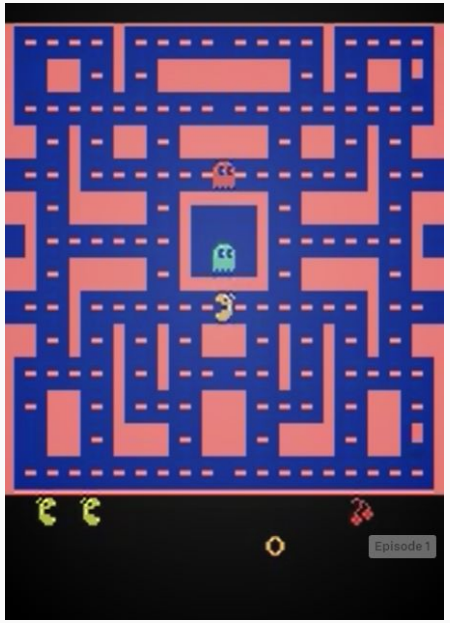
Problemy rozwiązywane przez RL

Podejmowanie decyzji

w reakcji na otoczenie

wpływając na te otoczenie

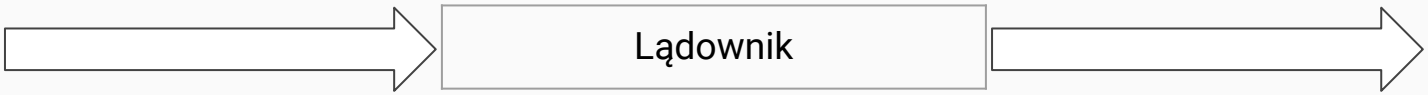




Przykład

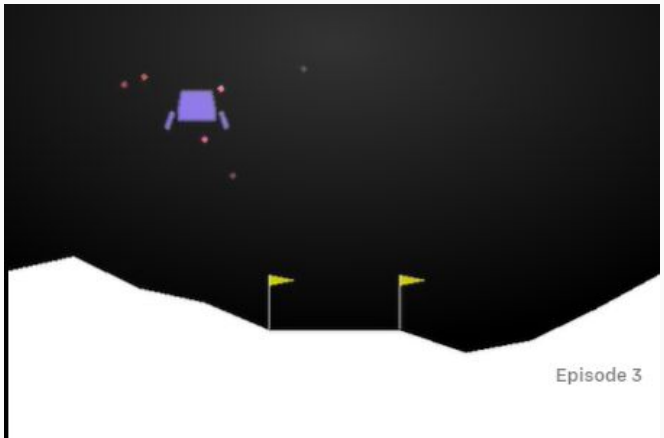
Cel: wylądować lądownikiem zużywając jak najmniej paliwa.

W każdej chwili:



- Prędkość
- Położenie
- Styk z podłożem

- Które silniki są włączone



Przykład

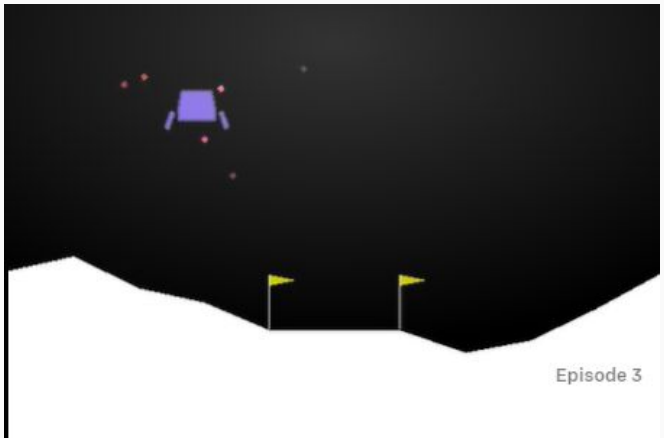
Cel: wylądować lądownikiem zużywając jak najmniej paliwa.

Zmaksymalizować sumę nagród za zbliżanie się do celu i kar za zużycie paliwa i uszkodzenie pojazdu

W każdej chwili:

- Prędkość
- Położenie
- Styk z podłożem

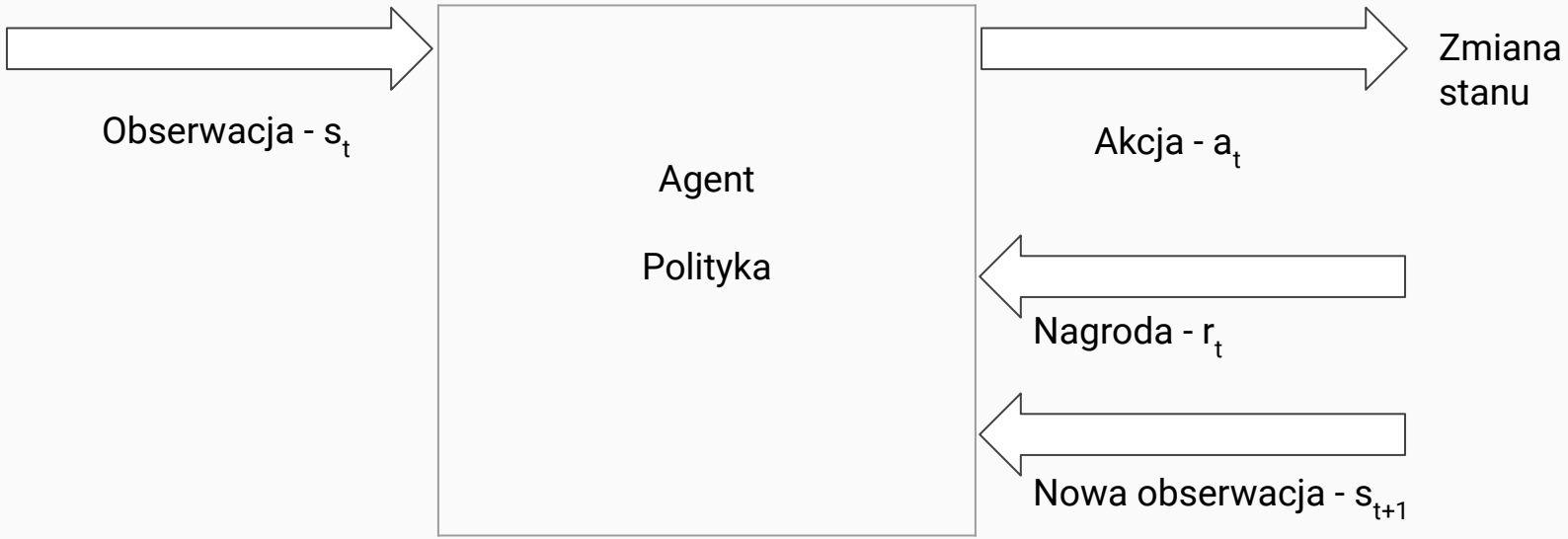
- Które silniki są włączone



Ogólnie

Szukanie polityki decyzyjnej,
która maksymalizuje sumę nagród.

W dyskretnej chwili czasowej t



Suma nagród - problemy

Może nie być zbieżna dla nieskończonych problemów

Trudna do wykorzystania - jaki jest wpływ poszczególnego kroku?

Rozwiązanie: zdyskontowana suma nagród

Dyskonto - $\gamma \in (0, 1]$

$$r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \gamma^4 r_{t+4} + \dots$$

Rozwiązanie: zdyskontowana suma nagród

Dyskonto - $\gamma \in (0, 1]$

$$r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \gamma^4 r_{t+4} + \ldots = \Sigma_{i=0} \gamma^i r_{t+i}$$

Maksymalizacja zdyskontowanej sumy nagród dla każdego stanu maksymalizuje sumę nagród

Q-learning

Q-Learning

Funkcja Q:

$$Q^{\pi}(s_t, a_t) = \mathbb{E}[\Sigma_{i=0} \gamma^i r_{t+i} \mid s_t, a_t, \pi]$$

Rekurencyjnie:

$$Q^{\pi}(s_t, a_t) = \mathbb{E}[r_t + \gamma Q^{\pi}(s_{t+1}, (a \mid s_{t+1}, \pi))]$$

Q-learning

Optymalna polityka:

$$a_t = \text{argmax}_a \, Q^{\pi}(s_t, a)$$

Funkcja Q:

$$Q^{\pi}(s_t, a_t) = \mathbb{E}[r_t + \gamma \max_a Q^{\pi}(s_{t+1}, a)]$$

Q-learning - uczenie

Uczenie $Q^{\pi}(s_t, a_t)$ tak, aby zminimalizować:

$$\mathbb{E}[(r_t + \gamma \max_a Q^{\pi}(s_{t+1}, a) - Q^{\pi}(s_t, a_t))^2]$$

Q-learning - eksploracja

Wybór akcji powinien

- Faworyzować akcję (akcje) o największych wartościach Q
- Nie być deterministyczny

Przykładowe możliwości:

- Prawdopodobieństwa jako softmax(Q)
- Z pewnym prawdopodobieństwem wybieramy najlepszą, wpr. losową

Implementacja funkcji Q

Oryginalnie tabela:

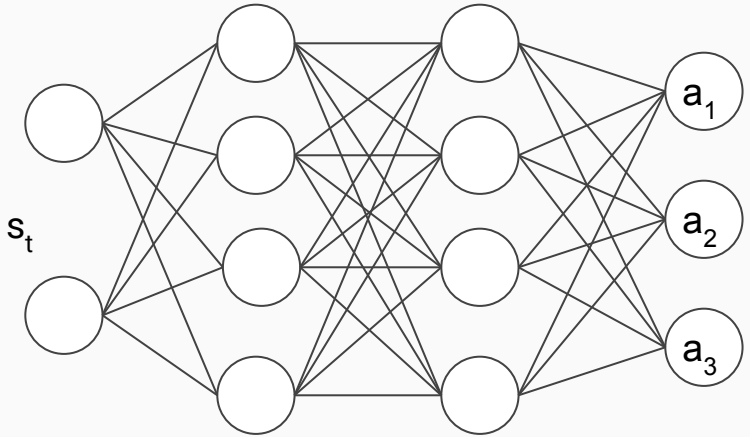
	a_1	a_2	a_2
s_1			
s_2			
s_3			
s_4			

Implementacja funkcji Q

Oryginalnie tabela:

	a_1	a_2	a_2
s_1			
s_2			
s_3			
s_4			

Sieć neuronowa:



Coding time!

DQN

Lepiej jest uczyć sieci na minibatchach nich na pojedynczych próbkach

Kolejne próbki są często do siebie podobne

Rozwiązanie - bufor z próbkami z poprzednich kroków

Coding time!

DQN - problemy

- Uczenie funkcji Q potrafi być niestabilne
- Nie wszystkie próbki są tak samo ważne
- Czasem może się opłacać uczenie na podstawie sekwencji próbek
- Inne…

DQN - problemy

- Uczenie funkcji Q potrafi być niestabilne
- Nie wszystkie próbki są tak samo ważne
- Czasem może się opłacać uczenie na podstawie sekwencji próbek
- Inne…

Algorytm RAINBOW łączący wiele wersji DQN

DQN - przestrzeń akcji

DQN jest algorytmem działającym na problemach o dyskretnej przestrzeni akcji.

Aktor-krytyk

Aktor-Krytyk

Dwa modele:

- Aktor - określa politykę decyzyjną na podstawie stanu
- Krytyk - estymuje zdyskontowaną sumę nagród

Przykładowa architektura

Aktor:

Sieć neuronowa,
która na wejściu przyjmuje obserwację,
a na wyjściu zwraca wektor akcji.

Krytyk:

Sieć neuronowa,
która na wejściu przyjmuje obserwację,
a na wyjściu zwraca estymację funkcji V:

$$V^{\pi}(s_t) = \mathbb{E}[\Sigma_{i=0} \gamma^i r_{t+i} \mid s_t, \pi]$$

Aktor a polityka

Podejmowanie akcji nie powinno być deterministyczne

Przykładowa polityka:
na wyjście aktora nakładany jest szum losowy

$a \sim N(A(s), [\sigma, \sigma, \ldots]^T)$

Uczenie krytyka

$$V^{\pi}(s_t) = \mathbb{E}[Q^{\pi}(s_t, a) \mid \pi] = \mathbb{E}[r_t + \gamma V^{\pi}(s_{t+1}) \mid \pi]$$

Uczenie krytyka

$$V^{\pi}(s_t) = \mathbb{E}[Q^{\pi}(s_t, a_t) \mid \pi] = \mathbb{E}[r_t + \gamma V^{\pi}(s_{t+1}, a) \mid \pi]$$

Problem: próbki z bufora nie mają zawsze akcji wybranych zgodnie z obecną polityką

Próbkowanie ważności (Importance sampling)

$\mathbb{E}X = \mathbb{E}[y * p_X(y) / p_Y(y) \mid y \sim Y]$

Waga próbek w buforze:

$IS(s_t, a_t) = \pi(a_t \mid s_t) / \pi_t(a_t \mid s_t)$

Uczenie krytyka

Uczenie $V(s_t)$ aby minimalizowało

$$(r_t + \gamma V(s_t) - V(s_t))^2 \, IS(s_t, a_t)$$

Uczenie aktora

Dopasowanie prawdopodobieństwa akcji tak,
aby akcje dające lepsze wyniki
miały większe prawdopodobieństwo

Określanie jakości akcji

Funkcja przewagi (advantage):

$$Q(s, a) - V(s)$$

Estymacja:

$$r_t + V(s_{t+1}) - V(s_t)$$

Uczenie aktora

Maksymalizacja

$$\ln \pi(a_t \mid s_t) \, (r_t + V(s_{t+1}) - V(s_t)) \, IS(s_t, a_t)$$

Ograniczenia

Często przestrzeń akcji jest ograniczona

Ograniczenia

Często przestrzeń akcji jest ograniczona

Typowym rozwiązaniem jest dodatkowy składnik optymalizacji:
kara za przekroczenie ograniczeń

Coding time!

Dalsze możliwości poprawiania algorytmów

- Ograniczanie maksymalnej zmiany w jednym kroku
- Adaptacja eksploracji
- Szum w kolejnych krokach podobny do siebie
- Inne…

Podział algorytmów uczenia ze wzmocnieniem

Model-free:

- Policy-based/**Value-based**
- **Online/Offline**

Model-based

(Imitation learning,
model-based data generation,
inverse reinforcement learning,
…)

Przydatne biblioteki

- Stable baselines
- RLlib
- SpinningUp
- Acme
- garage
- ...

Praca domowa

- Wytrenować agenta na środowisku CarRacing-v0
- Spróbować sił na tym środowisku
  z gotową implementacją algorytmu SAC
  z wybranego pakietu

Dziękuję
za uwagę

Bibliografia:

https://gym.openai.com/

V. Mnih et al, Playing Atari with Deep Reinforcement Learning

M. Hessel et al, Rainbow: Combining Improvements in Deep Reinforcement Learning

P. Wawrzyński, Real-time reinforcement learning by sequential Actor-Critics and experience replay