Reinforcement learning

Co wstrzymuje rozwój AI?

1. Computing (Moore's law, CPU/GPU, ASICs)

2. Data (complete and tidy, labeled)

Andrej Karpathy

3. Algorithms: research and ideas: backprop, CNNs, transformers, etc.

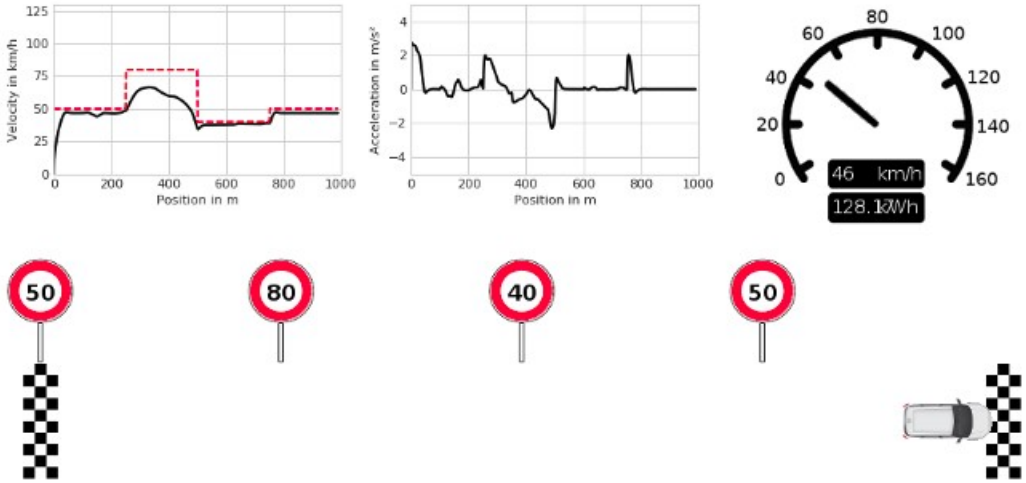
4. Infrastructure (Python, TensorFlow, Cloud Computing, Git)

David Silver

"Expert data sets are often expensive, unreliable or simply unavailable. [...]
Reinforcement learning systems are trained from their own experience,
in principle allowing them to exceed human capabilities,
and to operate in domains where human expertise is lacking."

Uczenie ze wzmocnieniem

reinforcement learning (RL)



- obserwacja aktualnego **STANU**
- wybór określonej **AKCJI**
- uzyskanie **NAGRODY/KARY**

© towardsdatascience.com

Główne wyzwanie:
- adekwatne środowisko symulacyjne

Co zyskujemy?
- rozwiązania, niekonieczne zgodne z naszym oczekiwaniem

RL: agent, interakcja ze środowiskiem

AGENT

AKCJA

NAGRODA

OBSERWACJA/STAN

ŚRODOWISKO

Założenie 1: czas jest dyskretny

Założenie 2: stan zależy tylko od bieżącej obserwacji

Markov Decision Process

Reinforcement vs supervised learning

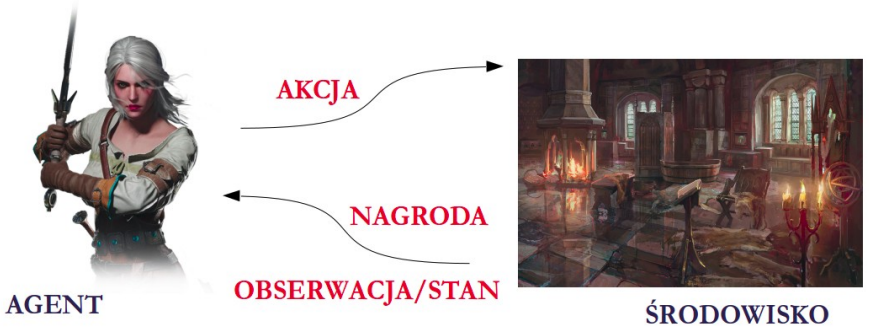
Uczenie ze wzmocnieniem:

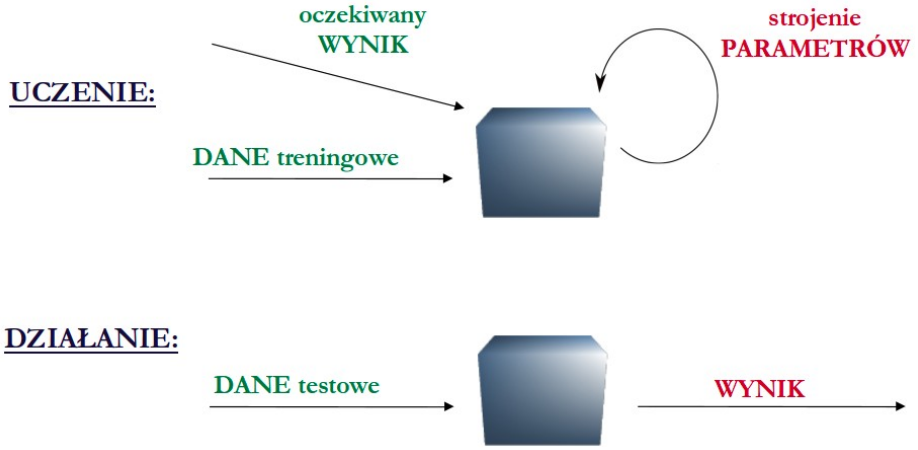
- nie potrzebujemy **danych(!)**
  ale konieczne jest **środowisko**
  i zdefiniowane **nagrody**

- uczymy się w systemie ciągłym,
  wykonując akcje

- kolejne stany zależą od siebie
  (sekwencja: STAN-AKCJA-STAN-AKCJA...)

- nagroda (feedback) może przyjść
  z opóźnieniem



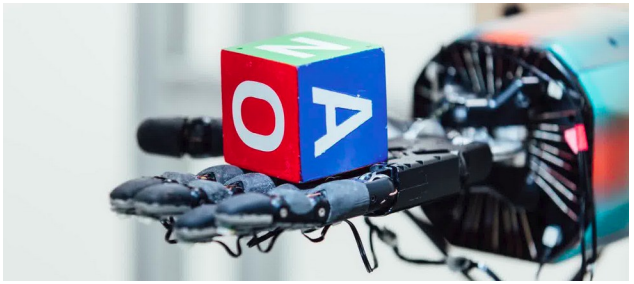


Gdzie stosować RL?

1. Autonomiczna nawigacja/prowadzenie pojazdów (samochody, drony, roboty, itd)

2. AI dla gier

3. Robotyka

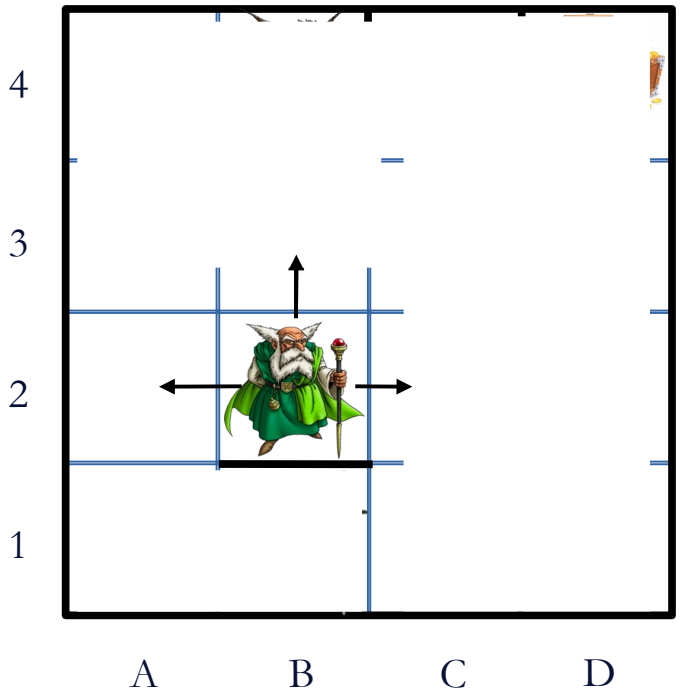


4. Inwestycje finansowe, transakcje giełdowe

W każdym wypadku potrzebujemy odpowiedniego środowiska dla RL!

Q-learning

Przykład z dziedziny gier: **Wędrówka przez labirynt**



STAN (s): B2

AKCJA (a): PRAWO

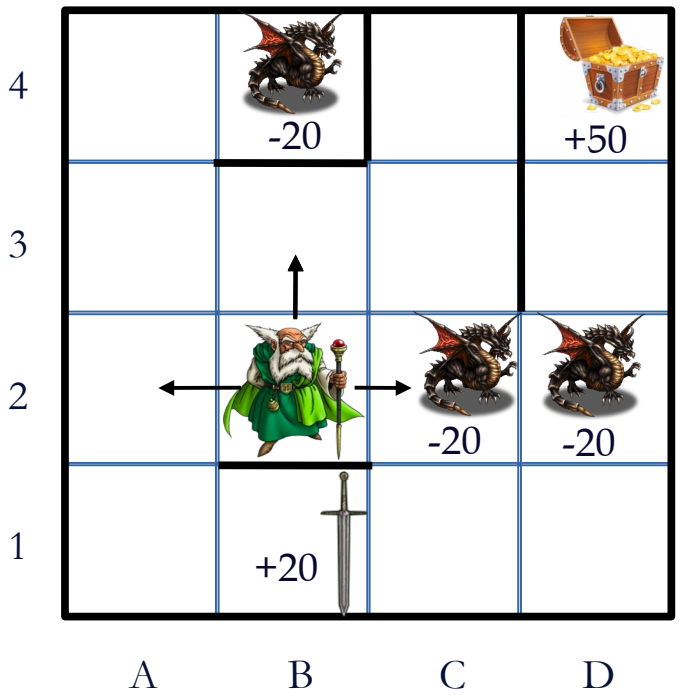
NAGRODA (r): -20

STAN (s): C2

AKCJA (a): ...

Kolejne gry: gromadzenie doświadczenia: (s, a, r, s)

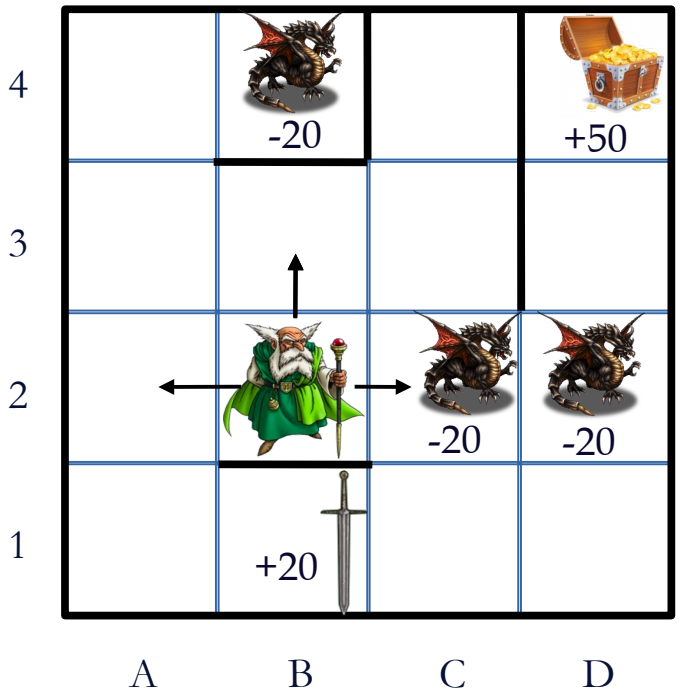
Q-learning: podejmowanie decyzji



Konstruowanie tablicy Q-table:

	↑	↓	←	→	
A1					
A2					
A3					
A4					
B1					
B2	0	x	0	-20	???
...					
D4					

Q-learning: wartości akcji



Jak ocenić wartość akcji **a** w stanie **s** (Q-value)?

Równanie Bellmana:

$$Q(s,a) = r + \gamma \max(Q(s',a'))$$

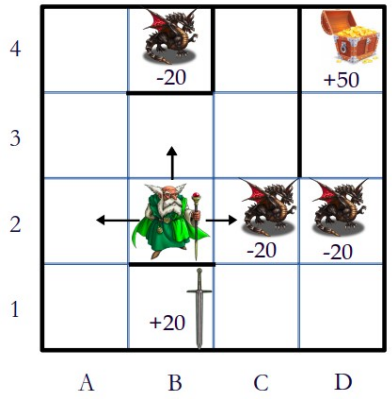
stan aktualny

nagroda

współczynnik
discount factor

stan kolejny
(po wykonaniu akcji a)

Q-learning: proces uczenia



Równanie Bellmana:

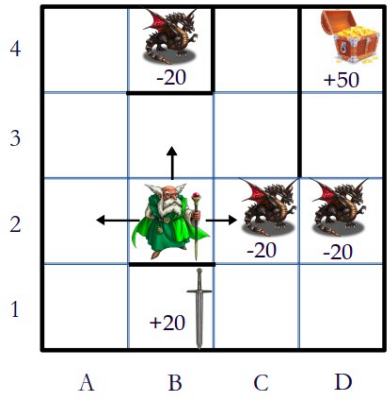
$$Q(s,a) = r + \gamma \max(Q(s',a'))$$

Aktualizacja wartości po zdobyciu nowego doświadczenia:

$$Q_{new}(s,a) = Q(s,a) + \alpha [r + \gamma \max(Q(s',a')) - Q(s,a)]$$

szybkość uczenia

Q-learning: to explore or to exploit?



Exploitation:

→ wybór najlepszej akcji wg aktualnej wiedzy (Q-table)

Exploration (np. z p-stwem ε):

→ wybór losowej akcji

Opinie?

Q-table (γ=0.9)

	↑	↓	←	→
A1	0	x	x	20
A2	0	18	x	0
A3	0	16.2	x	0
A4	x	0	x	-20
B1	x	x	18	18
B2	0	x	16.2	-20
...	...	...	...	...
D4	x	45	x	x

Podejścia zaawansowane #1: policy-based

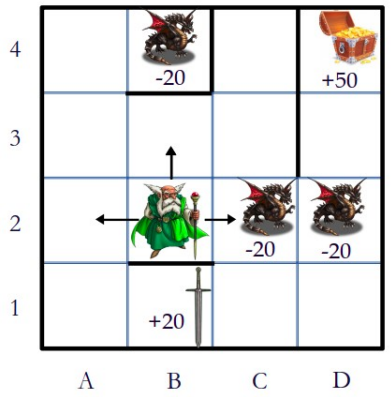
Policy $\pi(s)$

→ strategia 'co robić' (jaką akcję **a** wybrać) w stanie **s**

Strategie deterministyczne: $\pi(s_i) = a_i$

Strategie stochastyczne/probabilistyczne: $\pi(a_i | s_i) = P(a_i | s_i)$

Np. dla wcześniejszego labiryntu:
P(s=B2) = a: ↑(16%), ← (73%), → (11%)

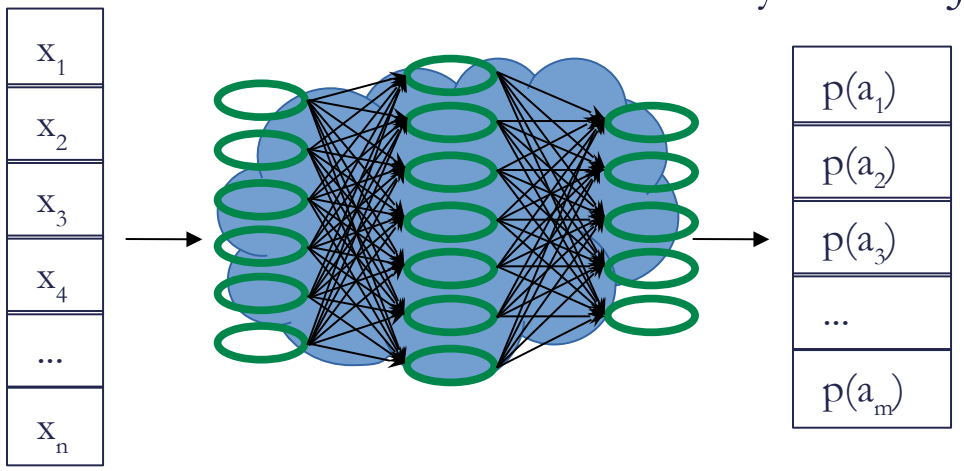


→ wybierając akcję próbkujemy rozkład **P**
→ problem exploit/explore automatycznie rozwiązany

Policy gradient + deep neural network

Policy π (s)

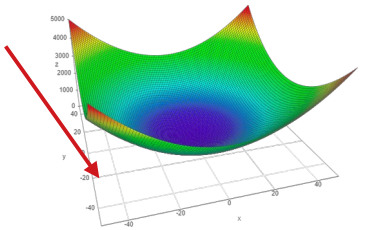
Wektor danych opisujący STAN s_i:



Prawdopodobieństwa możliwych AKCJI $a_1, \ldots, a_m$:

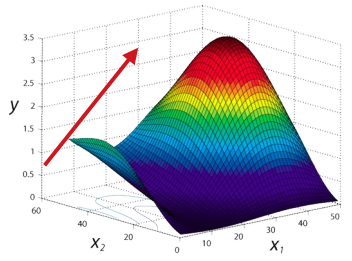
→ DNN: uniwersalny aproksymator :-)

→ w propagacji wstecznej liczymy gradient... ascent



Klasycznie, minimalizujemy **funkcję kosztu** ⇒ gradient descent

Tutaj maksymalizujemy **sumaryczną nagrodę** ⇒ gradient ascent



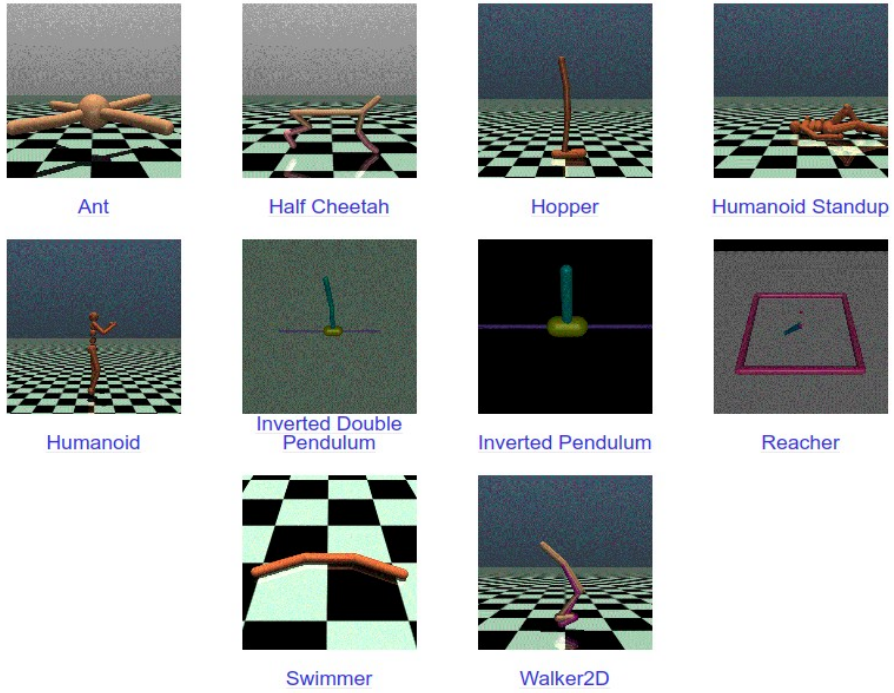
https://torres.ai/deep-reinforcement-learning-explained-series/

Narzędzia na dobry początek

→ **Gymnasium (OpenAI Gym):**
gry video, fizyczne modele ruchu obiektów, i inne

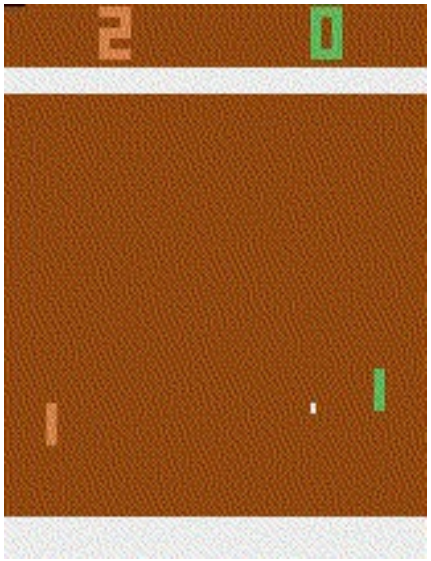
→ **Stable Baseline3 (OpenAI)**
implementacja szeregu algorytmów typu RL

→ **KerasRL**
→ **Pyqlearning**
→ **Tensorforce**

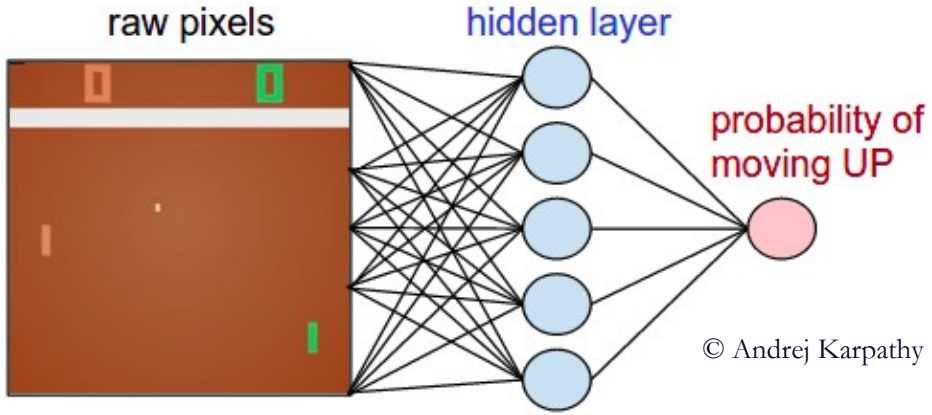


https://gymnasium.farama.org/ https://github.com/DLR-RM/stable-baselines3
https://github.com/keras-rl/keras-rl https://pypi.org/project/pyqlearning/
https://github.com/tensorforce/tensorforce

Przykład gry Pong z podejściem policy-based



→ wektor stanu: 100 800 elementów
(różnica dwóch kolejnych klatek, każda 210×160 pikseli RGB)
→ 2 możliwe akcje: ruch paletki góra/dół
→ nagrody: +1 wygrana, -1 porażka



Trening:

- nagroda pozwala na aktualizację parametrów sieci dopiero po ukończonej grze
- człowiek korzysta z intuicji dot. praw fizyki, RL startuje od zera
- RL by Karpathy: 3 noce treningu, wynik trochę lepszy niż bazowe AI
- czy można lepiej?

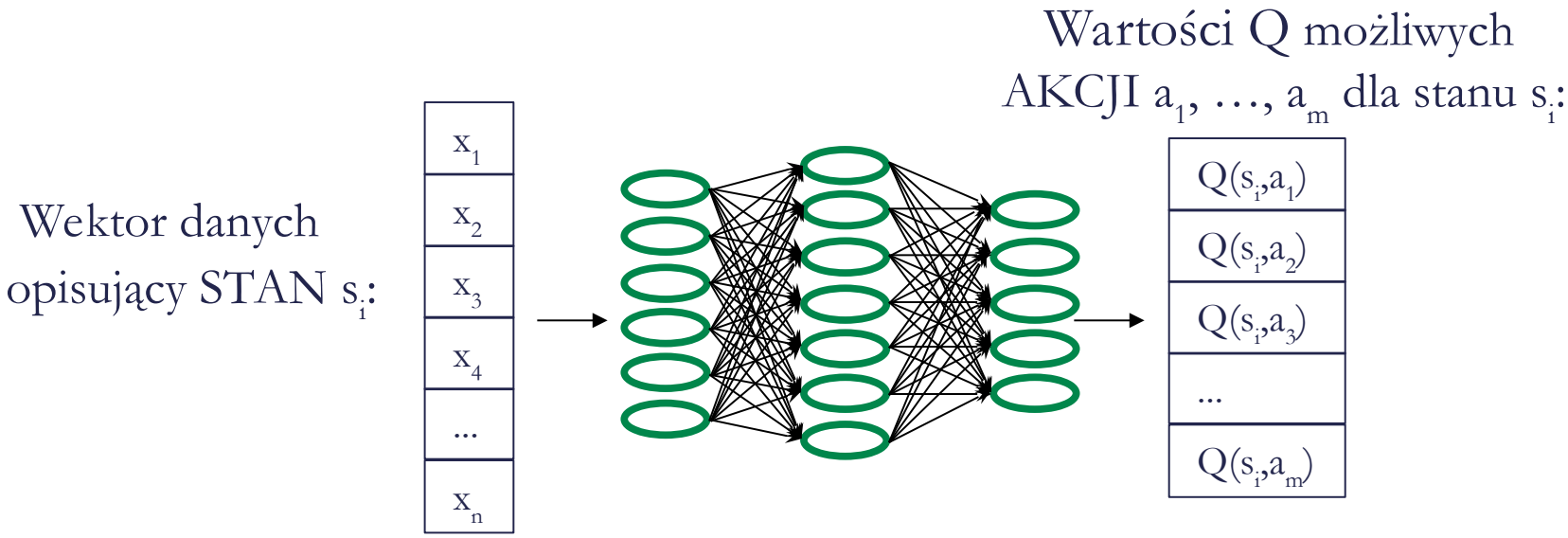
https://karpathy.github.io/2016/05/31/rl/

Podejścia zaawansowane #2: value-based

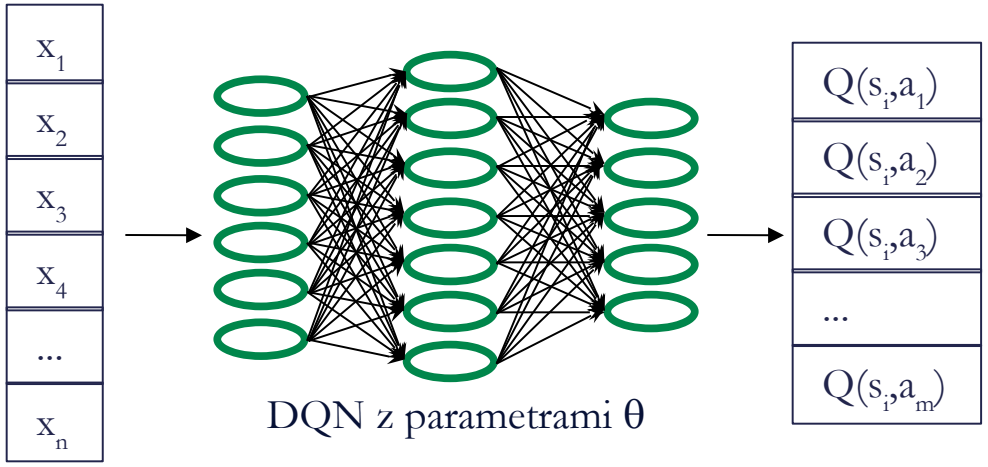
Celem jest obliczenie wartości Q(s,a)

Opcja 1. Podstawowe Q-learning

Opcja 2. Sieć neuronowa przewiduje Q(s,a):



Deep Q-Network (DQN)



Równanie Bellmana:

$$Q(s,a) = r + \gamma \max(Q(s',a'))$$

Funkcja kosztu (mse*) do propagacji wstecznej:

$$J = [r + \gamma \max(Q(s',a',\theta)) - Q(s,a,\theta)]^2$$

Szacowane przez DQN

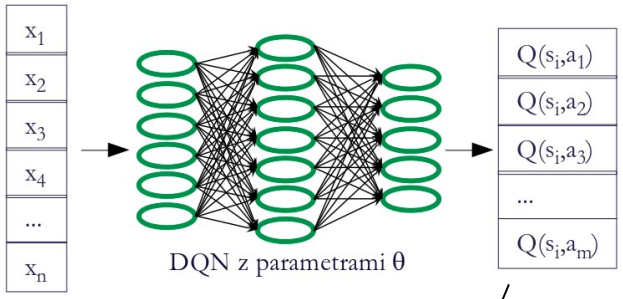
Exploitation/exploration
→ kontrolowane przez parametr ε

* Mean Square Error. Gdy koszt (błąd przewidywania sieci) jest bardzo duży, to stosowany jest Mean Absolute Error.
Dobry tutorial z MIT: https://cbmm.mit.edu/video/tutorial-reinforcement-learning-10733

DQN tricks and hacks

#1. Użycie dwóch kopii sieci:

$$J = [r + \gamma \max (Q (s',a',\theta)) - Q(s,a,\theta)]^2$$

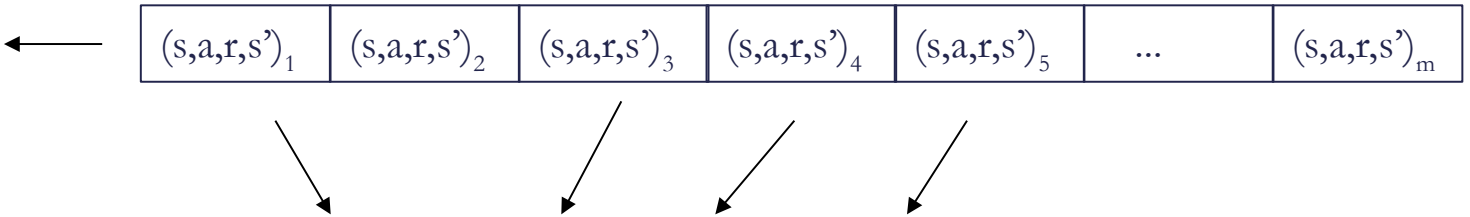


Szacowane przez tzw. 'target network' o zamrożonych parametrach, aktualizowanych tylko co pewien czas

Szacowane przez DQN, o parametrach aktualizowanych podczas propagacji wstecznej

#2. Experience replay

doświadczenia zbierane do bufora przesuwnego...



... i losowane na potrzebę treningu sieci DQN

Efekt: AI dla 49 gier na tej samej sieci i tych samych hiperparametrach*

* V. Mnih et al. "Human-level control through deep reinforcement learning", Nature 2015

Everything together... multiplied → A3C

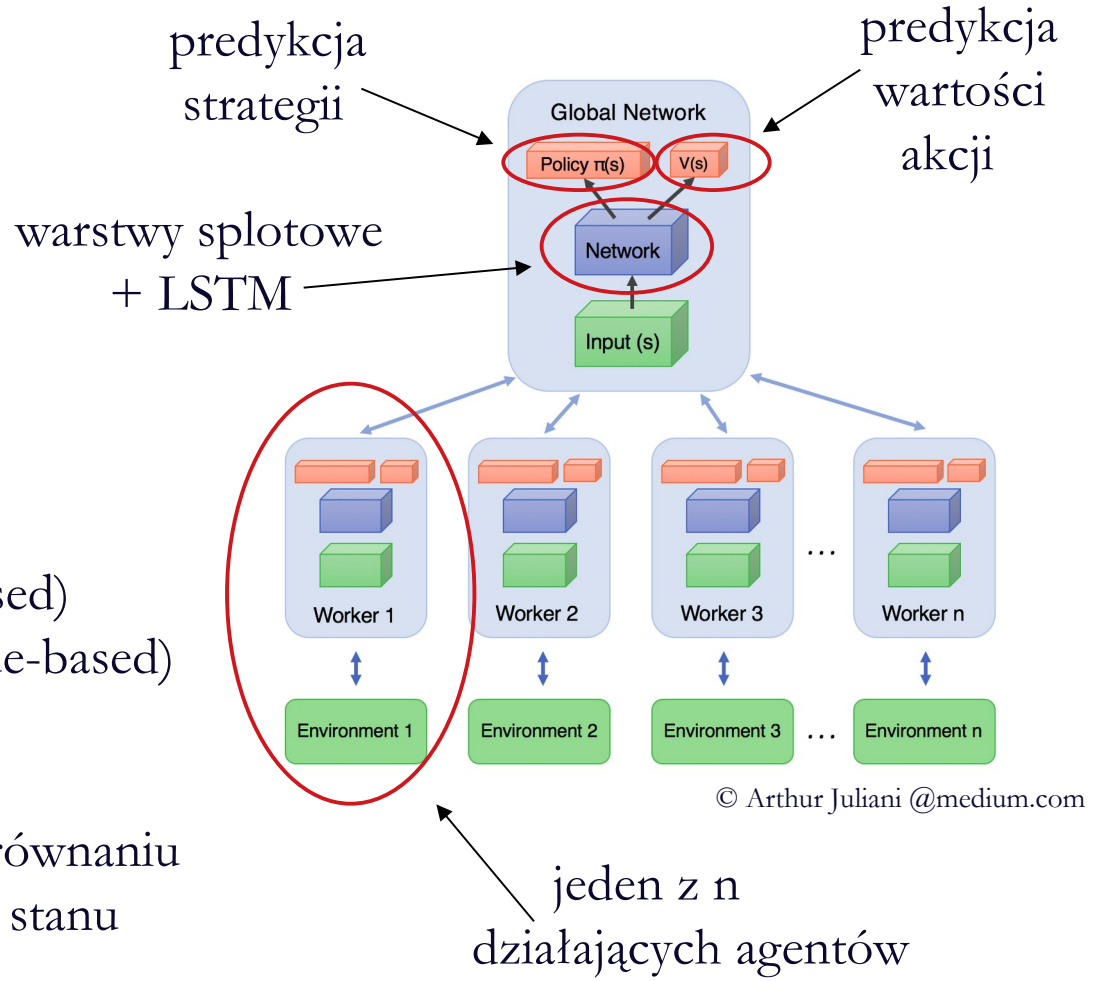
Asynchronous Advantage Actor-Critic algorithm

Mechanizmy:

1. Asynchronous:
wielu agentów jednocześnie zbiera doświadczenie na swoich kopiach środowiska

2. Actor-Critic:
oba podejścia: sieć Actor (policy-based) wspierana danymi z sieci Critic (value-based)

3. Advantage:
obliczanie wartości danej akcji w porównaniu ze średnią wartością strategii danego stanu

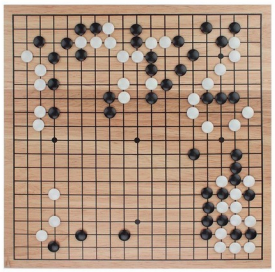


V. Mnih et al. "Asynchronous Methods for Deep Reinforcement Learning", Proc. of Machine Learning Research 2016

A. Juliani: https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-8-asynchronous-actor-critic-agents-a3c-c88f72a5e9f2

AI dla gier (DeepMind)

AlphaGo ZERO



→ Policy network + value network
→ Monte Carlo Tree Search

Gra typu:
- discrete
- deterministic
- with perfect information

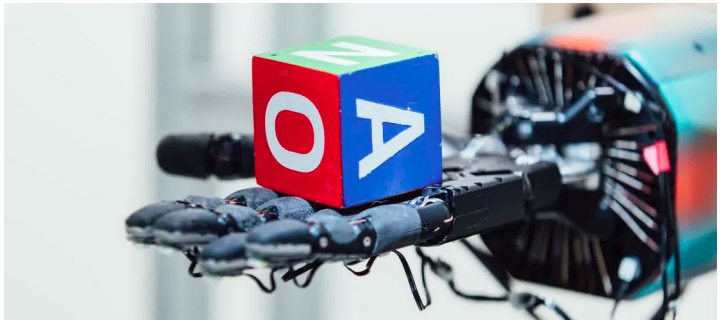
StarCraft II

→ Supervised (~milion gier)
+ multi-agent RL
→ Actor-Critic
→ Obserwacja gry przez sieć LSTM

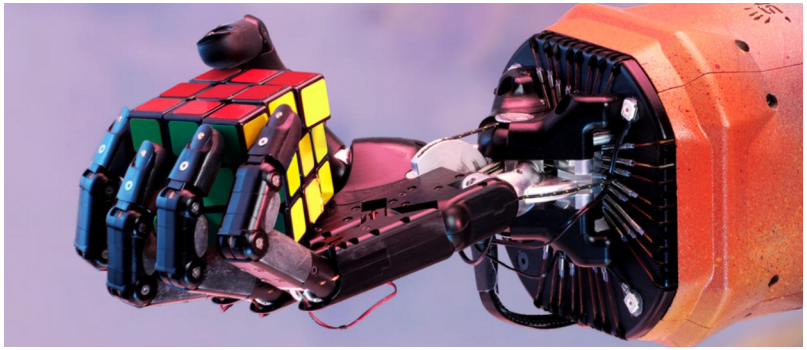
Seria zwycięstw z topowymi graczami (m.in Grzegorz 'MaNa' Komincz)

D. Silver et al. "Mastering the game of Go without human knowledge", Nature 2017
O. Vinyals et al. "Grandmaster level in Starcraft II using multi-agent reinforcement learning", Nature 2019

Robot z kostką Rubika (OpenAI)



→ Algorytm RL: Proximal Policy Optimization
(Advantage Actor-Critic z kontrolą tempa zmian parametrów sieci)
→ sieci splotowe, LSTM
→ symulacja z dużą losowością (domain randomization)
+ trening na rzeczywistym robocie



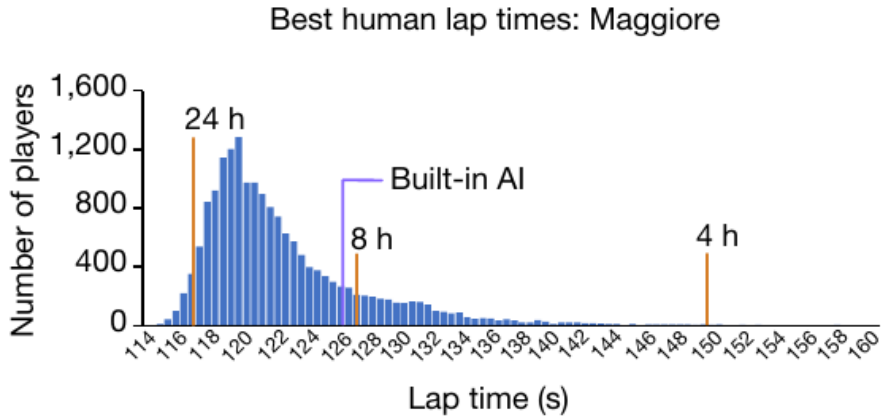
https://openai.com/blog/learning-dexterity/ https://openai.com/blog/solving-rubiks-cube/

I. Akkaya et al. "Solving Rubik's Cube with a Robot Hand", https://arxiv.org/pdf/1910.07113.pdf, 2019

Nawigacja/wyścigi samochodowe (Sony AI)

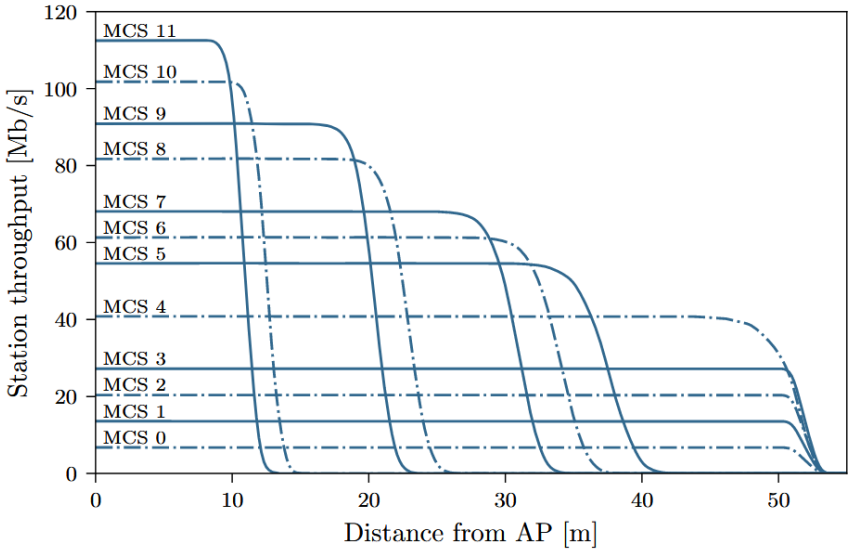
→ Algorytm QR-SAC
(wersja Actor-Critic)
→ experience replay buffers
→ aspekty: kontrola pojazdu, taktyka na torze, etykieta
→ długa lista nagród i kar (dot. środka trasy, ścian, wyprzedzania, kolizje, itp.)

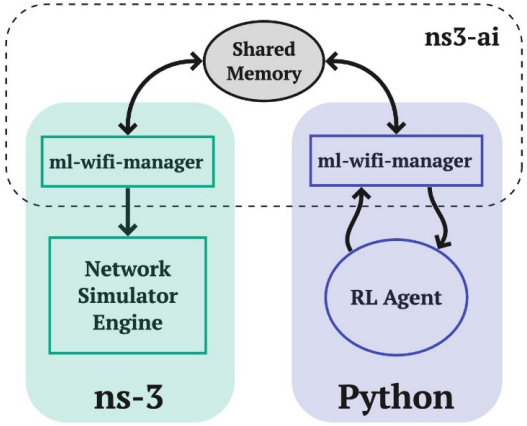
Wygrane wyścigi
z najlepszymi zawodnikami Gran Turismo

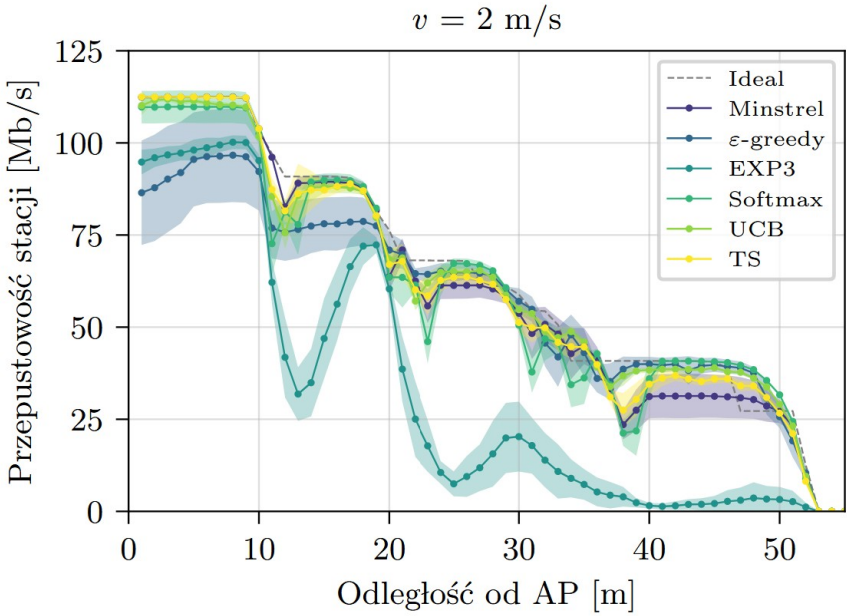


P. Wurman et al. "Outracing champion Gran Turismo drivers with deep reinforcement learning", Nature 2022
https://sonyai.github.io/gt_sophy_public/

RL w 802.11 (projekt Sz. Szotta)







RL: co zapamiętać

1. RL: bez danych, ale ze środowiskiem symulacyjnym

2. Uczenie: stan, akcja, nagroda, kolejny stan

3. Uczymy się strategii i/lub wartości (opłacalności) danych akcji

4. Głębokie sieci neuronowe zawsze się przydają

5. Nowoczesne algorytmy: everything combined :-)

Dziękuję!

Pytania?

Komentarze?