Reinforcement Learning

and DeepMind

Piotr Januszewski

DeepMind

- Kto to?

Londyn 2010

Firmę założyli Demis Hassabis, Shane Legg i Mustafa Suleyman w 2010 roku w Londynie.

> *Solve intelligence. Use it to make the world a better place.*

Ich celem jest "rozwiązać inteligencję" i tym samym stworzyć system, który będzie w stanie nauczyć się rozwiązywać dowolnie skomplikowany problem, bez potrzeby wskazania jak.

DeepMind

- Kto to?

- Debiut

Playing Atari with Deep Reinforcement Learning[1]

Komputer gra w 7 gier Atari 2600, w 6 z nich gra lepiej niż kiedykolwiek. W trzech z nich pokonuje człowieka na poziomie eksperta.

Human-level control through deep reinforcement learning[2]

49 gier Atari 2600. W 29 komputer gra na poziomie człowieka lub lepszym.

[1]Volodymyr Mnih, Koray Kavukcuoglu & David Silver, *Playing Atari with Deep Reinforcement Learning*, Dec 2013

[2]Volodymyr Mnih, Koray Kavukcuoglu & David Silver, *Human-level control through deep reinforcement learning*, "Nature" 518, Feb 2015

DeepMind

- Kto to?
- Debiut
- Okładka Nature

Breakout

Wejściem dla przykładowego modelu są 84x84 piksele aktualnego oraz trzech poprzednich obrazów gry w skali szarości. Chcemy, aby ten model nauczył się ruszać paletką w lewo, w prawo i wystrzelić piłkę na początku gry. *Proste, czyż nie?*

Nagroda jest opóźniona

Moment dotknięcia piłeczki przez paletkę oraz otrzymania za to nagrody dzieli pewien czas. W dodatku często ciężko powiedzieć, jaka dokładnie akcja skutkowała otrzymaniem nagrody.

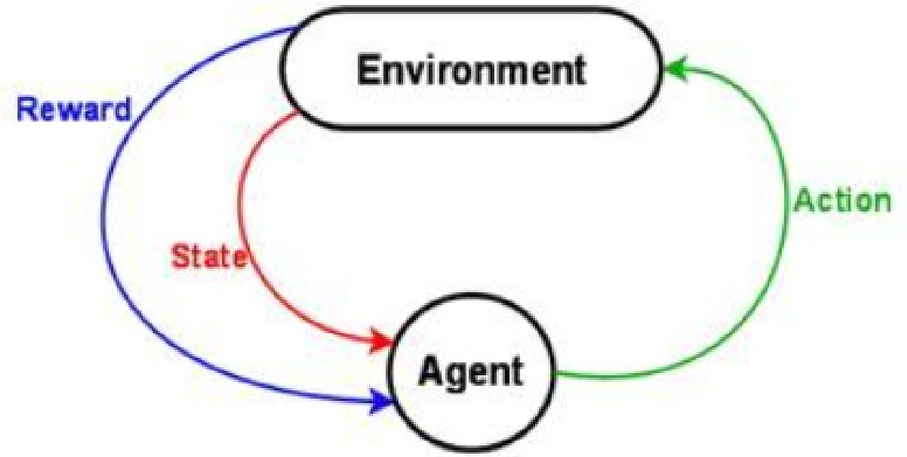
Czy ryzykować?

Model czegoś się nauczył i gra nawet nieźle. Powinien brnąć w tą strategię dalej, czy próbować nowych innych technik?

RL

- Cel
- Problemy
- MDP

Sformalizować uczenie ze wzmocnieniem

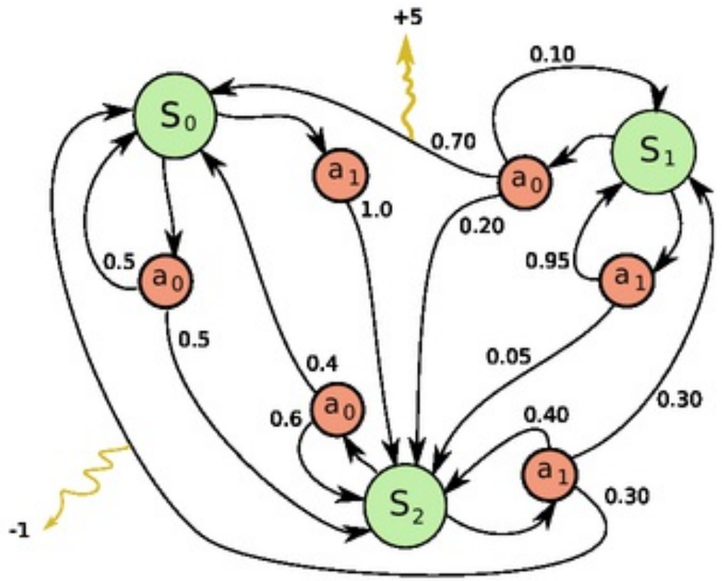


Agent obserwuje środowisko. Podejmuje akcje na podstawie stanu środowiska w którym się znalazł oraz swoich zasad. Podjęta akcja wpływa na środowisko (nie koniecznie w pełni przewidywalny sposób), a agent dostaje odpowiedź w postaci nagrody (lub jej braku).

RL

- Cel
- Problemy
- MDP

Sformalizować uczenie ze wzmocnieniem



Zbiór stanów i akcji wraz z zasadami jak je podejmować oraz nagrodami za pewne akcje tworzą proces decyzyjny Markov-a.

RL

- Cel
- Problemy
- MDP
- Q-learning

Całkowita przyszła nagroda

$$R_t = r_t + r_{t+1} + r_{t+2} + \ldots + r_n$$

Całkowita przyszła nagroda jest sumą nagród od momentu 't' aż do moment 'n', który jest końcem gry.

Całkowita obniżona przyszła nagroda

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} \ldots + \gamma^{n-t} r_n$$

Ponieważ nasze środowisko nie jest w pełni przewidywalne, powinniśmy jakoś to uwzględnić. Zatem wielkość każdej nagrody im dalej w przyszłość tym bardziej będzie obniżona o współczynik γ ($0 < \gamma < 1$).

...tak samo, ale inaczej

$$R_t = r_t + \gamma(r_{t+1} + \gamma(r_{t+2} + \ldots)) = r_t + \gamma R_{t+1}$$

RL

- Cel
- Problemy
- MDP
- Q-learning

Funkcja wartości akcji

$$Q(s_t, a_t) = max\ R_t$$

Funkcja ta mówi nam, jaka jest w stanie 's' po podjęciu akcji 'a' maksymalna całkowita przyszła obniżona nagroda.

Teraz możemy zdefiniować naszą optymalną politykę (zasady podejmowania akcji)

$$\pi(s) = argmax_a\ Q(s, a)$$

W stanie 's' zawsze podejmujemy taką akcję 'a', która da nam największą maksymalną całkowitą przyszłą obniżoną nagrodę.

Ostatnie pytanie... jak znaleźć taką funkcję?

W najprostrzy sposób, można przedstawić ją tablicą w której każda para stan-akcja mają swój wpis. Na początku wypełnić ją losowo i w miarę grania przez naszego agenta, uaktualniać ją w następujący sposób:

$$Q(s,a) = r + \gamma max_{a'} Q(s',a')$$

Istnieje matematyczny dowód, że takie uaktualnianie tablicy Q dąży do optymalnej funkcji wartości akcji.

Ale czy to jest na pewno wystarczająca metoda?

Cztery obrazy 84x84 pikseli w skali szarości dają nam $256^{4\times84\times84} \approx 10^{67960}$ stanów, czyli znacznie więcej niż cząsteczek w widzialnym Wszechświecie.

Rozwiązanie problemów

Nagroda jest opóźniona

Nasza funkcja Q w momencie kiedy musimy podjąć akcje bierze pod uwagę, że nagroda pojawi się w przyszłości.

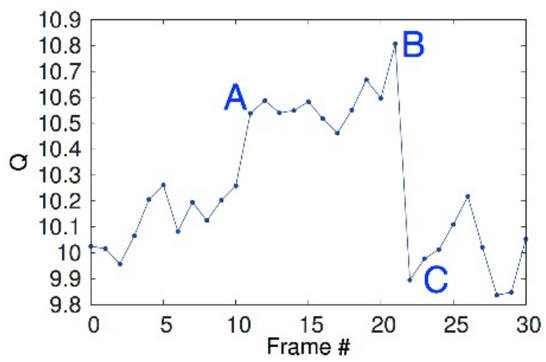
Czy ryzykować?

W trakcie uczenia do naszej optymalnej polityki możemy dodać pewne prawdopodobieństwo podjęcia losowej akcji. Tym samym złamiemy naszą politykę, ale pozwalamy naszemy modelowi na eksplorację środowiska.

Przykłady

"Wartość" przykładowych stanów

- Seaquest

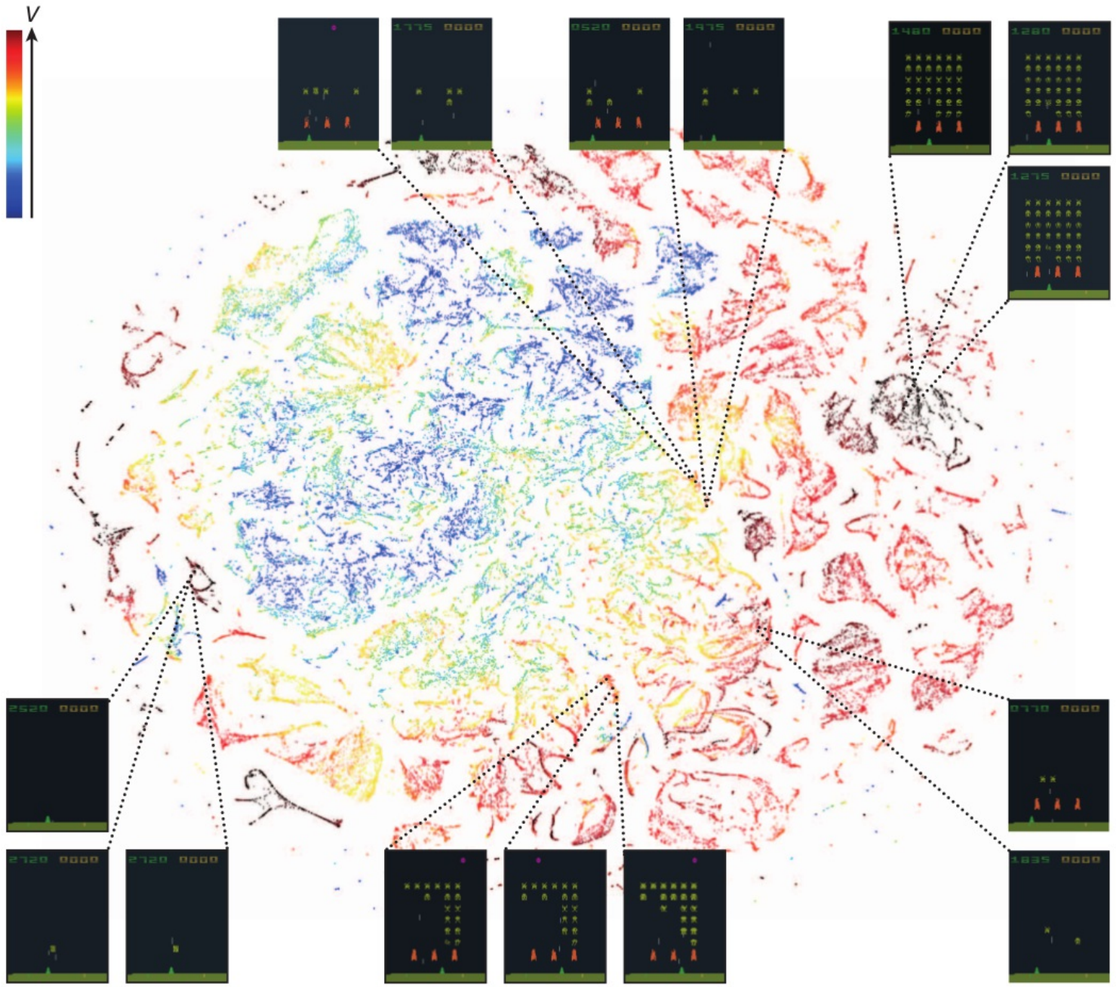


1. Kiedy przeciwnik pojawia się z lewej strony ekranu, funkcja wartosci zwraca większe wartości.
2. Zaraz przed uderzeniem torpedy w przeciwnika, funkcja wartości ma największą wartość.
3. Po zniszczeniu przeciwnika wartość spada.

Przykłady

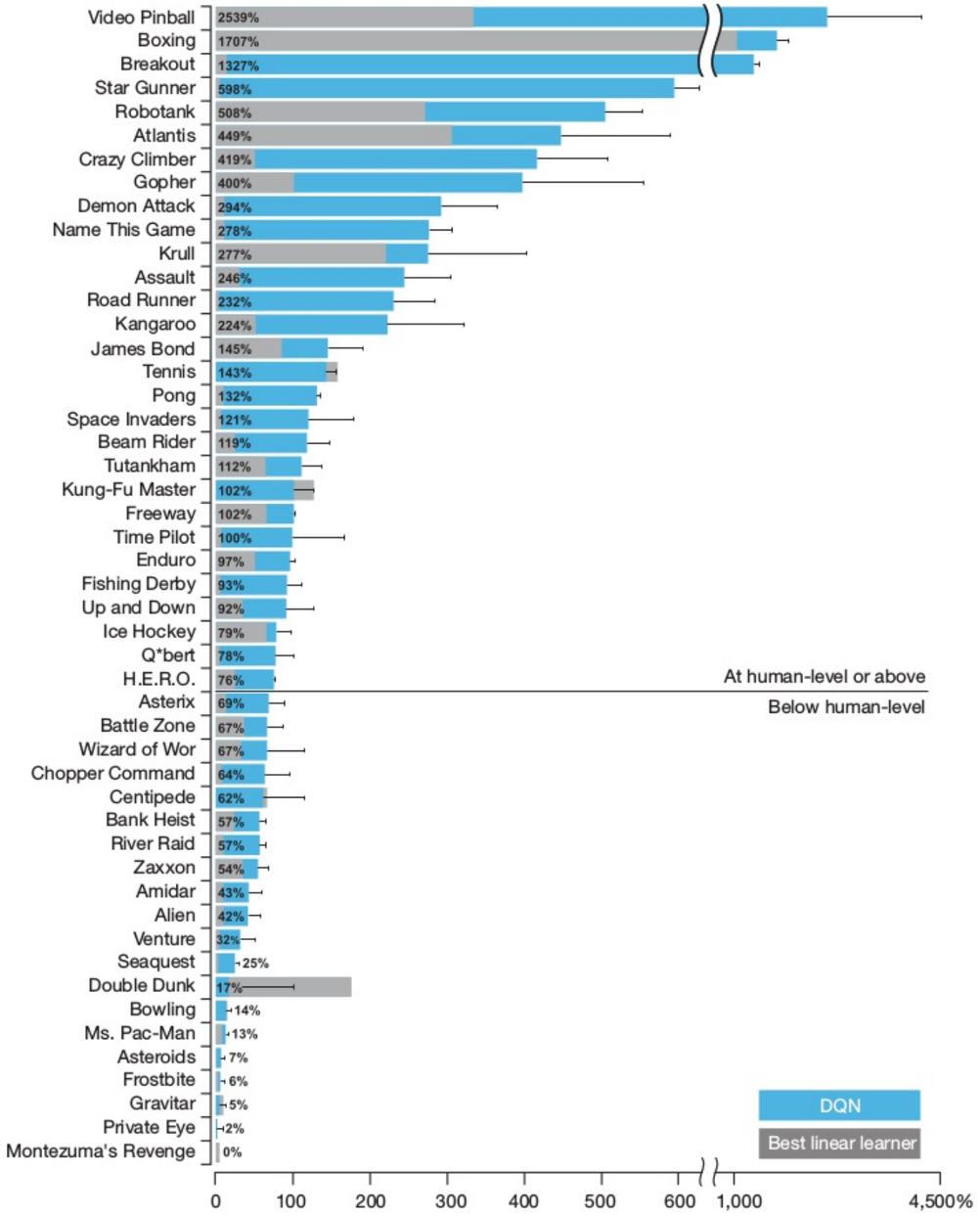
- Seaquest

- t-SNE



Przykłady

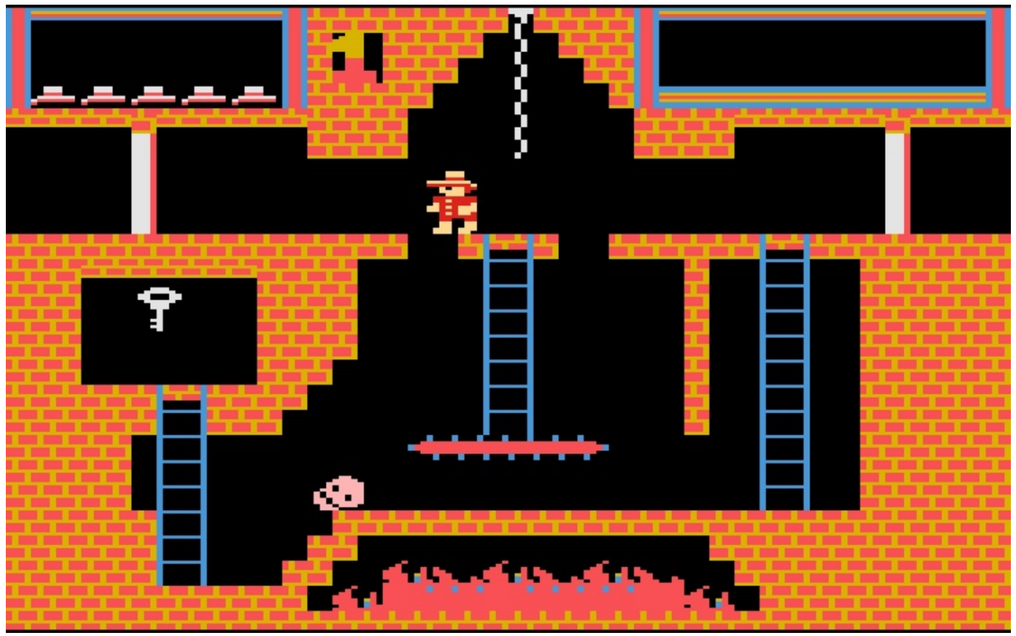
- Seaquest
- t-SNE
- Porównanie



Przykłady

- Seaquest
- t-SNE
- Porównanie

Montezuma's Revenge

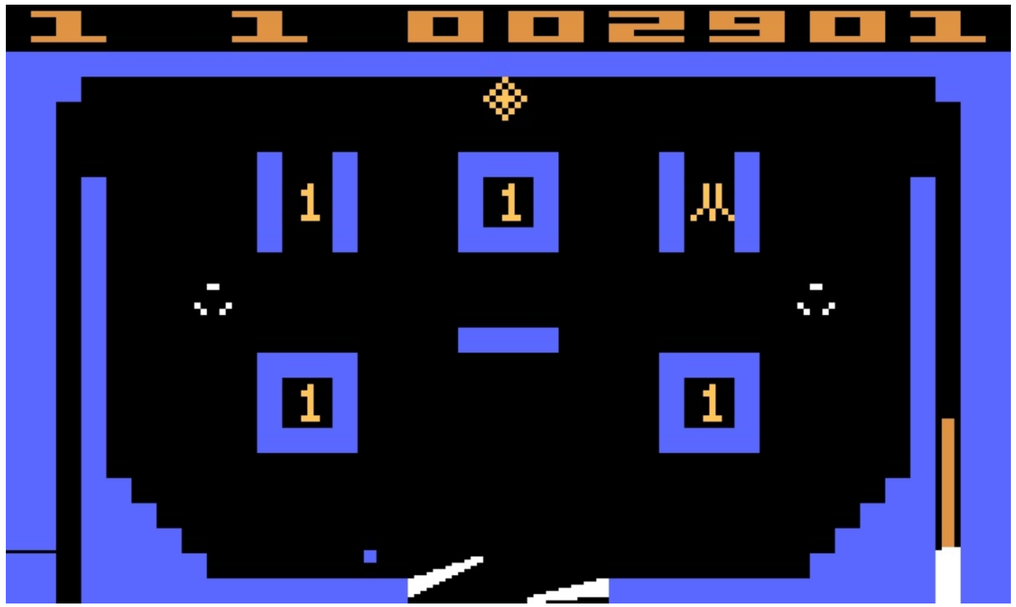


Środowisko w tej grze jest bardzo złożone i różnorodne, a cele nie są trywialne.

Przykłady

- Seaquest
- t-SNE
- Porównanie

Video Pinball



Proste środowisko i jasno zdefiniowane nagrody.

Przykłady

- Seaquest
- t-SNE
- Porównanie
- **Filmiki**

Przykładowe rozgrywki

Film przedstawia dwie przykładowe rozgrywki z dwóch gier w różnych stadiach rozwoju sieci. Gry to Breakout oraz Space Invaders.

Podsumowanie

Jeden model, wiele problemów

DeepMind przedstawiło model, który otrzymując "gołe" piksele oraz nagrody jako wejścia, z użyciem jednej i tej samej metody uczenia jest w stanie zrozumieć wiele zróżnicowanych środowisk.

Neurobiologiczne poszlaki

Model ten musiał sam nauczyć się widzieć, a następnie zrozumieć co widzi. Zrobił to otrzymując jedynie nagrody za podjęte przez siebie akcje. Sugeruje to, że sygnały o nagrodach podczas rozwijania percepcji, mogą mieć również wpływ na rozwój kory wzrokowej.

Dziękuję :)

Pytania?

Źródła

https://www.nervanasys.com/demystifying-deep-reinforcement-learning/
https://arxiv.org/abs/1312.5602
https://www.nature.com/nature/journal/v518/n7540/full/nature14236.html
https://en.wikipedia.org/wiki/Markov_decision_process
https://en.wikipedia.org/wiki/Reinforcement_learning