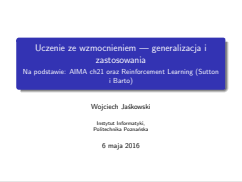
Uczenie ze wzmocnieniem — generalizacja i zastosowania

Na podstawie: AIMA ch21 oraz Reinforcement Learning (Sutton i Barto)

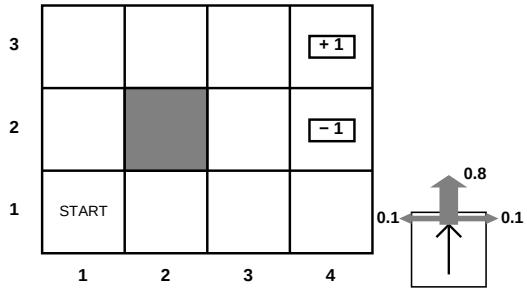
Wojciech Jaśkowski

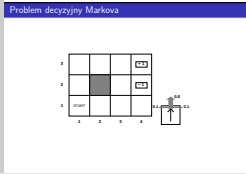
Instytut Informatyki,
Politechnika Poznańska

6 maja 2016

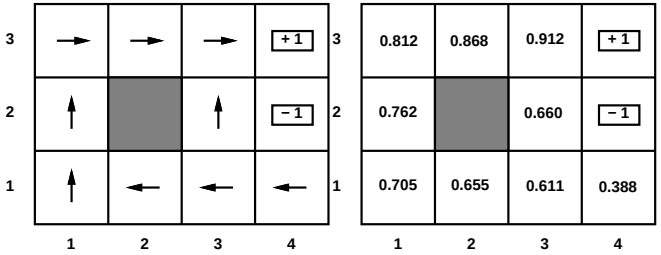


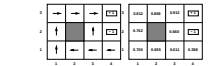
Problem decyzyjny Markova





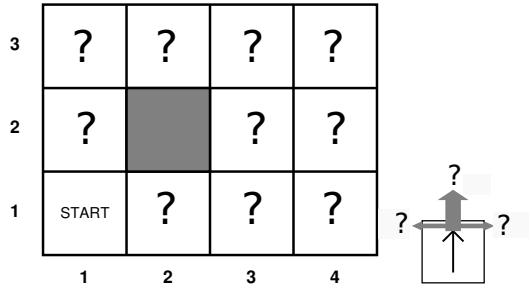
Rozwiązanie problemu decyzyjnego Markova





nieznany MDP

brak f. nagrody i modelu przejść



1. Co jest gorsze? Brak f. nagrody, czy modelu przejść? Raczej: brak modelu przejść.

Uczenie ze wzmocnieniem

Uczenie:

- **pasywne (problem predykcji)** $\rightarrow$ ocena użyteczności danej polityki π
- **aktywne (problem sterowania)** $\rightarrow$ znalezienie optymalnej polityki π
  - eksploracja!

Uczenie ze wzmocnieniem
Uczenie:
pasywne (problem predykcji) → ocena użyteczności danej polityki π
aktywne (problem sterowania) → znalezienie optymalnej polityki π
eksploracja!

Rodzaje uczenia

1. **Uczenie nadzorowane** (nauczyciel)
   1. **klasyfikacja**: *stan → [znana!] klasa decyzyjna*
   2. **regresja**: *stan → [znana!] wartość*
2. **Uczenie ze wzmocnieniem** (krytyk)
   1. stan → [nieznana!] klasa (akcja), ale wzmocnienie.
3. **Uczenie nienadzorowane** (brak nauczyciela)
   1. stan → [*nieznana*!] klasa

Rodzaje uczenia
Uczenie nadzorowane (nauczyciel)
klasyfikacja: stan → [znana!] klasa decyzyjna
regresja: stan → [znana!] wartość
Uczenie ze wzmocnieniem (krytyk)
stan → [nieznana!] klasa (akcja), ale wzmocnienie.
Uczenie nienadzorowane (brak nauczyciela)
stan → [nieznana!] klasa

1. Stan jest zwykle opisywany atrybutami, cechami (np. typowa „tabelka” w problemie klasyfikacji)

Podejścia do uczenia ze wzmocnieniem

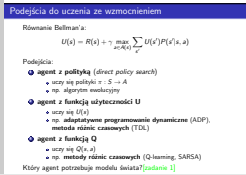
Równanie Bellman'a:

$$U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} U(s')P(s'|s,a)$$

Podejścia:

1. **agent z polityką** (*direct policy search*)
   - uczy się polityki $\pi : S \to A$
   - np. algorytm ewolucyjny
2. **agent z funkcją użyteczności U**
   - uczy się $U(s)$
   - np. **adaptatywne programowanie dynamiczne** (ADP), **metoda różnic czasowych** (TDL)
3. **agent z funkcją Q**
   - uczy się $Q(s,a)$
   - np. **metody różnic czasowych** (Q-learning, SARSA)

Który agent potrzebuje modelu świata?[zadanie 1]



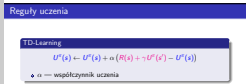
1. Teraźniejszość $R(s)$ + oczekiwana zdyskontowana przyszłość.
2. która mapuje bezpośrednio stany na akcje.
3. musi posiadać model środowiska, żeby podejmować decyzje.
4. nie musi posiadać modelu środowiska

Reguły uczenia

TD-Learning

$$U^\pi(s) \leftarrow U^\pi(s) + \alpha\left(R(s) + \gamma U^\pi(s') - U^\pi(s)\right)$$

- α — współczynnik uczenia



Reguły uczenia

TD-Learning

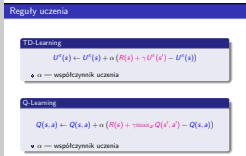
$$U^{\pi}(s) \leftarrow U^{\pi}(s) + \alpha\left(R(s) + \gamma U^{\pi}(s') - U^{\pi}(s)\right)$$

- α — współczynnik uczenia

Q-Learning

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left(R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a)\right)$$

- α — współczynnik uczenia



Approksymator funkcji

Przestrzeń stanów:

- ADP działa rozsądnie dla problemów wielkości rzędu 10000 stanów.
  - tryktrak (backgammon): 10^{20}
  - szachy: 10^{40}
- Nie da się explicite rozważać tylu stanów

Approksymator funkcji
Przestrzeń stanów:
- ADP działa rozsądnie dla problemów wielkości rzędu 10000 stanów.
  - tryktrak (backgammon): 10^{20}
  - szachy: 10^{40}
- Nie da się explicite rozważać tylu stanów

1. (jeśli dobrze się zaimplementuje),
2. Ale wcale nie musi być liniowa

Approksymator funkcji

Przestrzeń stanów:

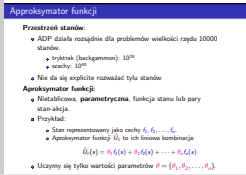
- ADP działa rozsądnie dla problemów wielkości rzędu 10000 stanów.
  - tryktrak (backgammon): 10^{20}
  - szachy: 10^{40}
- Nie da się explicite rozważać tylu stanów

Aproksymator funkcji:

- Nietablicowa, **parametryczna**, funkcja stanu lub pary stan-akcja.
- Przykład:
  - Stan reprezentowany jako cechy $f_1, f_2, \ldots, f_n$.
  - Aproksymator funkcji $\hat{U}_\theta$ to ich liniowa kombinacja:

$$\hat{U}_\theta(s) = \theta_1 f_1(s) + \theta_2 f_2(s) + \cdots + \theta_n f_n(s)$$

- Uczymy się tylko wartości parametrów $\theta = (\theta_1, \theta_2, \ldots, \theta_n)$.



1. (jeśli dobrze się zaimplementuje),
2. Ale wcale nie musi być liniowa

Przykład

$$\hat{U}_\theta(s) = \theta_1 \mathit{pionków}(s) + \theta_2 \mathit{figur\_w\_centrum}(s) + \theta_3 \mathit{hetman?}(s) + \theta_4 \mathit{szach?}(s)$$

10^{40} stanów $\rightarrow$ 4 parametry

Przykład
$\hat{U}_\theta(s) = \theta_1 \mathit{pionków}(s) + \theta_2 \mathit{figur\_w\_centrum}(s) + \theta_3 \mathit{hetman?}(s) + \theta_4 \mathit{szach?}(s)$
10^{40} stanów $\rightarrow$ 4 parametry

Aproksymator funkcji

Właściwości:

1. musi być łatwo obliczalny,
   - Cechy oparte na np. przeszukiwaniu Minimax'em są dobre, ale kosztowne obliczeniowo. Potrzebujemy prostych cech.

Aproksymator funkcji
Właściwości:
musi być łatwo obliczalny,
Cechy oparte na np. przeszukiwaniu Minimax'em są dobre, ale kosztowne obliczeniowo. Potrzebujemy prostych cech.

Aproksymator funkcji

Właściwości:

1. musi być łatwo obliczalny,
   - Cechy oparte na np. przeszukiwaniu Minimax'em są dobre, ale kosztowne obliczeniowo. Potrzebujemy prostych cech.
2. „**kompresuje**" (dużą) przestrzeń stanów w (małą) liczbę parametrów,
3. $\Longrightarrow$ **uogólniania wiedzę** (stany odwiedzone vs. nieodwiedzone),
   - Przykład: co 10^{12} stan $\rightarrow$ „mistrzowski" gracz w tryktraka

Aproksymator funkcji
Właściwości:
1. musi być łatwo obliczalny,
   - Cechy oparte na np. przeszukiwaniu Minimax'em są dobre, ale kosztowne obliczeniowo. Potrzebujemy prostych cech.
2. „kompresuje" (dużą) przestrzeń stanów w (małą) liczbę parametrów,
3. $\Longrightarrow$ uogólniania wiedzę (stany odwiedzone vs. nieodwiedzone),
   - Przykład: co 10^{12} stan $\rightarrow$ „mistrzowski" gracz w tryktraka

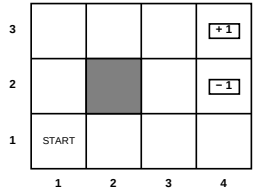
Aproksymator funkcji

Właściwości:

1. musi być łatwo obliczalny,
   - Cechy oparte na np. przeszukiwaniu Minimax'em są dobre, ale kosztowne obliczeniowo. Potrzebujemy prostych cech.
2. „**kompresuje**" (dużą) przestrzeń stanów w (małą) liczbę parametrów,
3. $\Longrightarrow$ **uogólniania wiedzę** (stany odwiedzone vs. nieodwiedzone),
   - Przykład: co 10^{12} stan $\rightarrow$ „mistrzowski" gracz w tryktraka
4. Kompromis: wielkość przestrzeni (jakość aproksymacji) vs. czas nauki / pamięć.

Aproksymator funkcji
Właściwości:
1. musi być łatwo obliczalny,
   - Cechy oparte na np. przeszukiwaniu Minimax'em są dobre, ale kosztowne obliczeniowo. Potrzebujemy prostych cech.
2. „kompresuje" (dużą) przestrzeń stanów w (małą) liczbę parametrów,
3. $\Longrightarrow$ uogólniania wiedzę (stany odwiedzone vs. nieodwiedzone),
   - Przykład: co 10^{12} stan $\rightarrow$ „mistrzowski" gracz w tryktraka
4. Kompromis: wielkość przestrzeni (jakość aproksymacji) vs. czas nauki / pamięć.

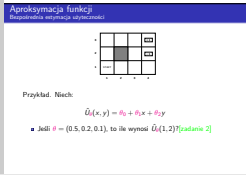
Aproksymacja funkcji
Bezpośrednia estymacja użyteczności



Przykład. Niech:

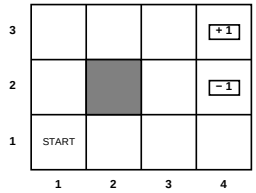
$$\hat{U}_\theta(x, y) = \theta_0 + \theta_1 x + \theta_2 y$$

- Jeśli $\theta = (0.5, 0.2, 0.1)$, to ile wynosi $\hat{U}_\theta(1, 2)$?[zadanie 2]



1. $\hat{U}_\theta(1, 1) = 0.5 + 0.2 + 0.2 = 0.9$
2. $u(1, 2)$ jest sumarycznym zdyskontowanym wzmocnieniem.

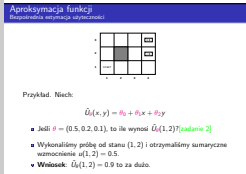
Aproksymacja funkcji
Bezpośrednia estymacja użyteczności



Przykład. Niech:

$$\hat{U}_\theta(x, y) = \theta_0 + \theta_1 x + \theta_2 y$$

- Jeśli $\theta = (0.5, 0.2, 0.1)$, to ile wynosi $\hat{U}_\theta(1, 2)$?[zadanie 2]
- Wykonaliśmy próbę od stanu $(1, 2)$ i otrzymaliśmy sumaryczne wzmocnienie $u(1, 2) = 0.5$.
- **Wniosek**: $\hat{U}_\theta(1, 2) = 0.9$ to za dużo.

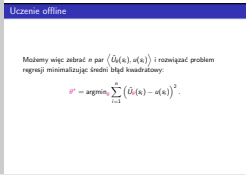


1. $\hat{U}_\theta(1, 1) = 0.5 + 0.2 + 0.2 = 0.9$
2. $u(1, 2)$ jest sumarycznym zdyskontowanym wzmocnieniem.

Uczenie offline

Możemy więc zebrać n par $\left\langle \hat{U}_\theta(s_i), u(s_i) \right\rangle$ i rozwiązać problem regresji minimalizując średni błąd kwadratowy:

$$\theta^* = \text{argmin}_\theta \sum_{i=1}^{n} \left(\hat{U}_\theta(s_i) - u(s_i) \right)^2 .$$



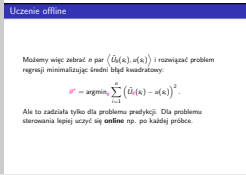
1. Np. regresja liniowa jest prosta obliczeniowo.

Uczenie offline

Możemy więc zebrać n par $\left\langle \hat{U}_\theta(s_i), u(s_i) \right\rangle$ i rozwiązać problem regresji minimalizując średni błąd kwadratowy:

$$\theta^* = \text{argmin}_\theta \sum_{i=1}^{n} \left(\hat{U}_\theta(s_i) - u(s_i) \right)^2 .$$

Ale to zadziała tylko dla problemu predykcji. Dla problemu sterowania lepiej uczyć się **online** np. po każdej próbce.



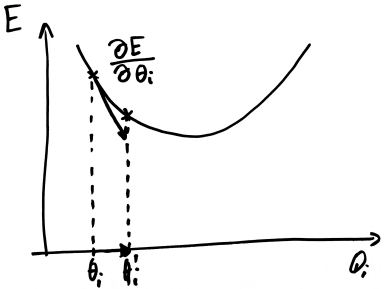
1. Np. regresja liniowa jest prosta obliczeniowo.

Reguła Widrow-Hoff'a

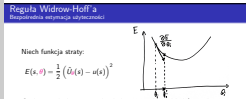
Bezpośrednia estymacja użyteczności

Niech funkcja straty:

$$E(s,\theta) = \frac{1}{2}\left(\hat{U}_\theta(s) - u(s)\right)^2$$



Szukamy takich parametrów, które minimalizują błąd (metoda gradientu prostego):



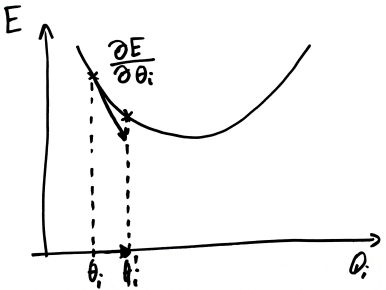
1. Dlaczego $\frac{1}{2}$ w funkcji błędu? Żeby się skróciły stałe przy różniczkowaniu. Generalnie, to nie ma znaczenia, bo jest α
2. α jest współczynnikiem, który mówi, jak bardzo chcemy się zbliżyć do celu. Jeśli α będzie maleć w czasie, zbiegniemy do optimum.

Reguła Widrow-Hoff'a

Bezpośrednia estymacja użyteczności

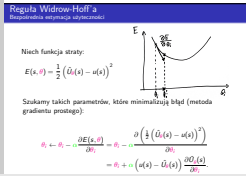
Niech funkcja straty:

$$E(s,\theta) = \frac{1}{2}\left(\hat{U}_\theta(s) - u(s)\right)^2$$



Szukamy takich parametrów, które minimalizują błąd (metoda gradientu prostego):

$$\theta_i \leftarrow \theta_i - \alpha\frac{\partial E(s,\theta)}{\partial\theta_i} = \theta_i - \alpha\frac{\partial\left(\frac{1}{2}\left(\hat{U}_\theta(s) - u(s)\right)^2\right)}{\partial\theta_i}$$
$$= \theta_i + \alpha\left(u(s) - \hat{U}_\theta(s)\right)\frac{\partial\hat{U}_\theta(s)}{\partial\theta_i}.$$



1. Dlaczego $\frac{1}{2}$ w funkcji błędu? Żeby się skróciły stałe przy różniczkowaniu. Generalnie, to nie ma znaczenia, bo jest α
2. α jest współczynnikiem, który mówi, jak bardzo chcemy się zbliżyć do celu. Jeśli α będzie maleć w czasie, zbiegniemy do optimum.

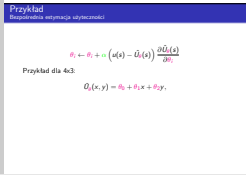
Przykład

Bezpośrednia estymacja użyteczności

$$\theta_i \leftarrow \theta_i + \alpha \left(u(s) - \hat{U}_\theta(s) \right) \frac{\partial \hat{U}_\theta(s)}{\partial \theta_i}$$

Przykład dla 4x3:

$$\hat{U}_\theta(x, y) = \theta_0 + \theta_1 x + \theta_2 y,$$



Przykład

Bezpośrednia estymacja użyteczności

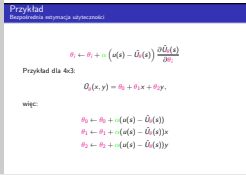
$$\theta_i \leftarrow \theta_i + \alpha \left(u(s) - \hat{U}_\theta(s) \right) \frac{\partial \hat{U}_\theta(s)}{\partial \theta_i}$$

Przykład dla 4x3:

$$\hat{U}_\theta(x, y) = \theta_0 + \theta_1 x + \theta_2 y,$$

więc:

$$\theta_0 \leftarrow \theta_0 + \alpha(u(s) - \hat{U}_\theta(s))$$
$$\theta_1 \leftarrow \theta_1 + \alpha(u(s) - \hat{U}_\theta(s))x$$
$$\theta_2 \leftarrow \theta_2 + \alpha(u(s) - \hat{U}_\theta(s))y$$



Przykład

Bezpośrednia estymacja użyteczności

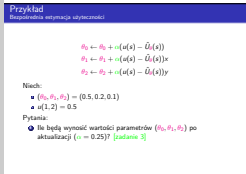
$$\theta_0 \leftarrow \theta_0 + \alpha(u(s) - \hat{U}_\theta(s))$$
$$\theta_1 \leftarrow \theta_1 + \alpha(u(s) - \hat{U}_\theta(s))x$$
$$\theta_2 \leftarrow \theta_2 + \alpha(u(s) - \hat{U}_\theta(s))y$$

Niech:

- $(\theta_0, \theta_1, \theta_2) = (0.5, 0.2, 0.1)$
- $u(1,2) = 0.5$

Pytania:

1. Ile będą wynosić wartości parametrów $(\theta_0, \theta_1, \theta_2)$ po aktualizacji $(\alpha = 0.25)$? [zadanie 3]



1. $\alpha\delta = 0.25 \times 0.4 = 0.1$. więc $\theta = (0.5 - 0.1, 0.2 - 0.1 \times 1, 0.1 - 0.1 \times 2) = (0.4, 0.1, -0.1)$
2. Wtedy $\hat{U}_\theta(1,2) = 0.4 + 0.1 \times 1 - 0.1 \times 2 = 0.3$
3. Zauważmy przy okazji: wartość aktualna wynosiła 0.9 a docelowa 0.5, ale ostatecznie ją przeskoczyliśmy (0.3). Wniosek: zbyt duży współczynnik α.
4. Tak! I to jest istota **generalizowania**!

Przykład

Bezpośrednia estymacja użyteczności

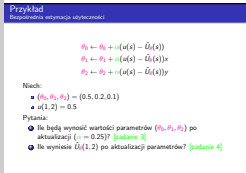
$$\theta_0 \leftarrow \theta_0 + \alpha(u(s) - \hat{U}_\theta(s))$$
$$\theta_1 \leftarrow \theta_1 + \alpha(u(s) - \hat{U}_\theta(s))x$$
$$\theta_2 \leftarrow \theta_2 + \alpha(u(s) - \hat{U}_\theta(s))y$$

Niech:

- $(\theta_0, \theta_1, \theta_2) = (0.5, 0.2, 0.1)$
- $u(1,2) = 0.5$

Pytania:

1. Ile będą wynosić wartości parametrów $(\theta_0, \theta_1, \theta_2)$ po aktualizacji $(\alpha = 0.25)$? [zadanie 3]
2. Ile wyniesie $\hat{U}_\theta(1,2)$ po aktualizacji parametrów? [zadanie 4]



1. $\alpha\delta = 0.25 \times 0.4 = 0.1$. więc $\theta = (0.5 - 0.1, 0.2 - 0.1 \times 1, 0.1 - 0.1 \times 2) = (0.4, 0.1, -0.1)$
2. Wtedy $\hat{U}_\theta(1,2) = 0.4 + 0.1 \times 1 - 0.1 \times 2 = 0.3$
3. Zauważmy przy okazji: wartość aktualna wynosiła 0.9 a docelowa 0.5, ale ostatecznie ją przeskoczyliśmy (0.3). Wniosek: zbyt duży współczynnik α.
4. Tak! I to jest istota **generalizowania**!

Przykład

Bezpośrednia estymacja użyteczności

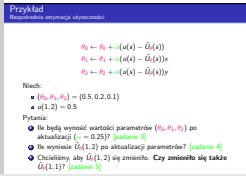
$$\theta_0 \leftarrow \theta_0 + \alpha(u(s) - \hat{U}_\theta(s))$$
$$\theta_1 \leftarrow \theta_1 + \alpha(u(s) - \hat{U}_\theta(s))x$$
$$\theta_2 \leftarrow \theta_2 + \alpha(u(s) - \hat{U}_\theta(s))y$$

Niech:

- $(\theta_0, \theta_1, \theta_2) = (0.5, 0.2, 0.1)$
- $u(1,2) = 0.5$

Pytania:

1. Ile będą wynosić wartości parametrów $(\theta_0, \theta_1, \theta_2)$ po aktualizacji ($\alpha = 0.25$)? [zadanie 3]
2. Ile wyniesie $\hat{U}_\theta(1,2)$ po aktualizacji parametrów? [zadanie 4]
3. Chcieliśmy, aby $\hat{U}_\theta(1,2)$ się zmieniło. **Czy zmieniło się także** $\hat{U}_\theta(1,1)$? [zadanie 5]

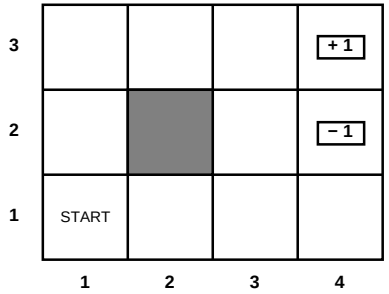


1. $\alpha\delta = 0.25 \times 0.4 = 0.1$. więc $\theta = (0.5 - 0.1, 0.2 - 0.1 \times 1, 0.1 - 0.1 \times 2) = (0.4, 0.1, -0.1)$
2. Wtedy $\hat{U}_\theta(1,2) = 0.4 + 0.1 \times 1 - 0.1 \times 2 = 0.3$
3. Zauważmy przy okazji: wartość aktualna wynosiła 0.9 a docelowa 0.5, ale ostatecznie ją przeskoczyliśmy (0.3). Wniosek: zbyt duży współczynnik α.
4. Tak! I to jest istota **generalizowania**!

Wybór aproksymatora — wiedza dziedzinowa

Generalizacja

Agent uczy się szybciej z aproksymatorem funkcji, bo może **generalizować**.

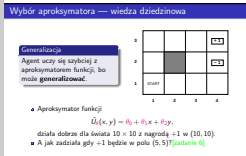


- Aproksymator funkcji

$$\hat{U}_\theta(x, y) = \theta_0 + \theta_1 x + \theta_2 y,$$

działa dobrze dla świata 10×10 z nagrodą $+1$ w $(10, 10)$.

- A jak zadziała gdy $+1$ będzie w polu $(5, 5)$?[zadanie 6]

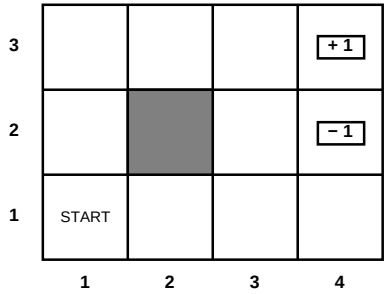


1. Porażka. Dla świata 10×10 funkcja użyteczności jest liniowa względem wymiarów, ale gdy $+1$ jest w polu $(5, 5)$ przypomina piramidę.
2. Zauważmy: cechy mogą być dowolnymi nieliniowymi elementami (np. liczba pionków, hetmanów, etc.)

Wybór aproksymatora — wiedza dziedzinowa

Generalizacja

Agent uczy się szybciej z aproksymatorem funkcji, bo może **generalizować**.



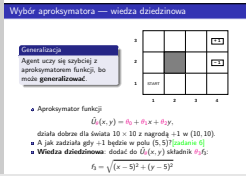
- Aproksymator funkcji

$$\hat{U}_\theta(x, y) = \theta_0 + \theta_1 x + \theta_2 y,$$

działa dobrze dla świata 10×10 z nagrodą $+1$ w $(10, 10)$.

- A jak zadziała gdy $+1$ będzie w polu $(5, 5)$?[zadanie 6]
- **Wiedza dziedzinowa**: dodać do $\hat{U}_\theta(x, y)$ składnik $\theta_3 f_3$:

$$f_3 = \sqrt{(x-5)^2 + (y-5)^2}$$



1. Porażka. Dla świata 10×10 funkcja użyteczności jest liniowa względem wymiarów, ale gdy $+1$ jest w polu $(5, 5)$ przypomina piramidę.
2. Zauważmy: cechy mogą być dowolnymi nieliniowymi elementami (np. liczba pionków, hetmanów, etc.)

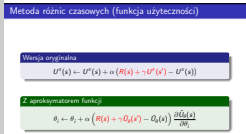
Metoda różnic czasowych (funkcja użyteczności)

Wersja oryginalna

$$U^\pi(s) \leftarrow U^\pi(s) + \alpha\left(R(s) + \gamma U^\pi(s') - U^\pi(s)\right)$$

Z aproksymatorem funkcji

$$\theta_i \leftarrow \theta_i + \alpha\left(R(s) + \gamma\hat{U}_\theta(s') - \hat{U}_\theta(s)\right)\frac{\partial\hat{U}_\theta(s)}{\partial\theta_i}$$



1. Zwróćmy uwagę: tutaj wartością **docelową** (target) jest zaznaczone na czerwono wyrażenie (wartość po wykonaniu jednego kroku), a nie $u(1,2)$ — wartość po całym epizodzie tak jak w metodzie Monte Carlo.

Q-learning

Wersja oryginalna

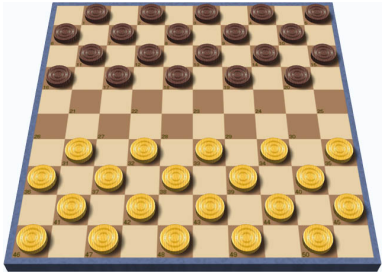
$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a)\right)$$

Z aproksymatorem funkcji

$$\theta_i \leftarrow \theta_i + \alpha \left(R(s) + \gamma \max_{a'} \hat{Q}_\theta(s',a') - \hat{Q}_\theta(s,a)\right) \frac{\partial \hat{Q}_\theta(s,a)}{\partial \theta_i}$$

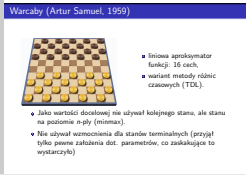
Q-learning
Wersja oryginalna
$Q(s,a) \leftarrow Q(s,a) + \alpha \left(R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a)\right)$
Z aproksymatorem funkcji
$\theta_i \leftarrow \theta_i + \alpha \left(R(s) + \gamma \max_{a'} \hat{Q}_\theta(s',a') - \hat{Q}_\theta(s,a)\right) \frac{\partial \hat{Q}_\theta(s,a)}{\partial \theta_i}$

Warcaby (Artur Samuel, 1959)



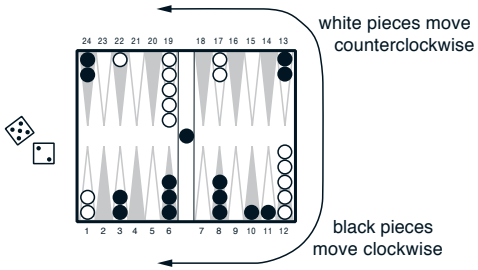
- liniowa aproksymator funkcji: 16 cech,
- wariant metody różnic czasowych (TDL).

- Jako wartości docelowej nie używał kolejnego stanu, ale stanu na poziomie *n*-ply (minmax).
- Nie używał wzmocnienia dla stanów terminalnych (przyjął tylko pewne założenia dot. parametrów, co zaskakujące to wystarczyło)

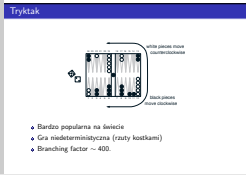


1. Mogło to doprowadzić do nauczenia się np. przegrywania.
2. Gra rozwiązana w 2007 (Schaeffer). Obaj gracze mają gwarantowany remis. Liczba stanów 5×10^{20}.

Tryktak



- Bardzo popularna na świecie
- Gra niedeterministyczna (rzuty kostkami)
- Branching factor ~ 400.



1. [Sutton & Barto, 1998] twierdzą, że bardziej popularna niż szachy (ale wydaje się, że nie jest to prawda jeśli chodzi o Polskę).

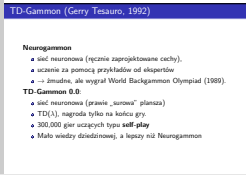
TD-Gammon (Gerry Tesauro, 1992)

Neurogammon

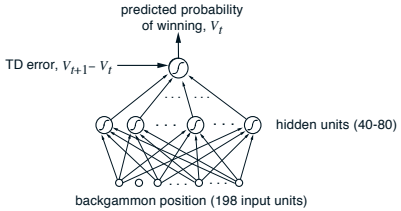
- sieć neuronowa (ręcznie zaprojektowane cechy),
- uczenie za pomocą przykładów od ekspertów
- $\rightarrow$ żmudne, ale wygrał World Backgammon Olympiad (1989).

TD-Gammon 0.0:

- sieć neuronowa (prawie „surowa" plansza)
- TD(λ), nagroda tylko na końcu gry.
- 300,000 gier uczących typu **self-play**
- Mało wiedzy dziedzinowej, a lepszy niż Neurogammon



TD-Gammon (Gerry Tesauro, 1992)



Cechy (198 wejść):

- dla każdego pola i koloru 4 wejścia ($2 \times 4 \times 24 = 192$):
  - 0 pionów $\rightarrow 0, 0, 0, 0$; 1 pion $\rightarrow 1, 0, 0, 0$
  - 2 piony $\rightarrow 1, 1, 0, 0$; 3 piony $\rightarrow 1, 1, 1, 0$
  - $n > 3$ pionów $\rightarrow 1, 1, 1, (n-3)/2$
- 2 wejścia: liczba pionów danego koloru na bandzie,
- 2 wejścia: liczba pionów w domu,
- 2 wejścia: czyj ruch ($0, 1$ i $1, 0$).

1. Klasyczna sigmoidalna sieć neuronowa.

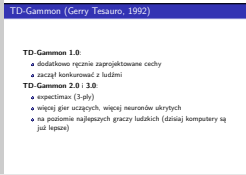
TD-Gammon (Gerry Tesauro, 1992)

TD-Gammon 1.0:

- dodatkowo ręcznie zaprojektowane cechy
- zaczął konkurować z ludźmi

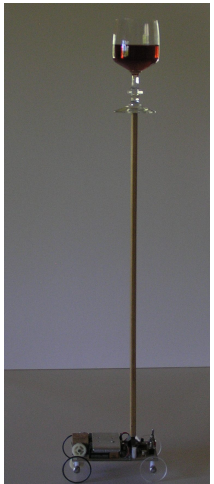
TD-Gammon 2.0 i **3.0**:

- expectimax (3-ply)
- więcej gier uczących, więcej neuronów ukrytych
- na poziomie najlepszych graczy ludzkich (dzisiaj komputery są już lepsze)

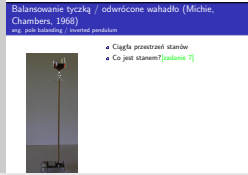


Balansowanie tyczką / odwrócone wahadło (Michie, Chambers, 1968)

ang. pole balanding / inverted pendulum

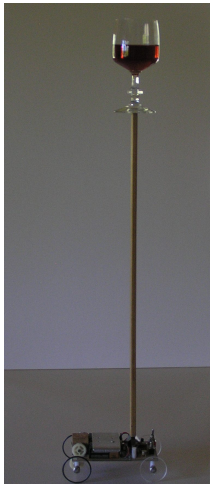


- Ciągła przestrzeń stanów
- Co jest stanem?[zadanie 7]



Balansowanie tyczką / odwrócone wahadło (Michie, Chambers, 1968)

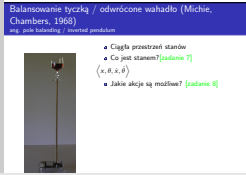
ang. pole balanding / inverted pendulum



- Ciągła przestrzeń stanów
- Co jest stanem?[zadanie 7]

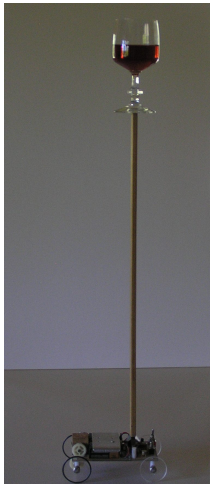
$\langle x, \theta, \dot{x}, \dot{\theta} \rangle$

- Jakie akcje są możliwe? [zadanie 8]



Balansowanie tyczką / odwrócone wahadło (Michie, Chambers, 1968)

ang. pole balanding / inverted pendulum

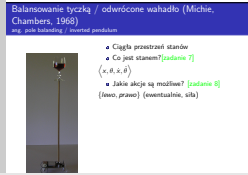


- Ciągła przestrzeń stanów
- Co jest stanem?[zadanie 7]

$\langle x, \theta, \dot{x}, \dot{\theta} \rangle$

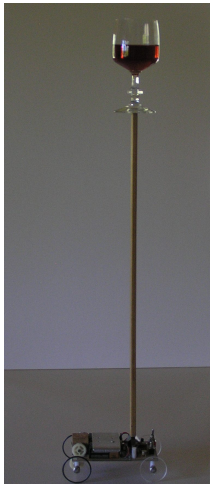
- Jakie akcje są możliwe? [zadanie 8]

$\{lewo, prawo\}$ (ewentualnie, siła)



Balansowanie tyczką / odwrócone wahadło (Michie, Chambers, 1968)

ang. pole balanding / inverted pendulum



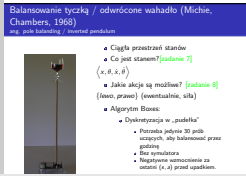
- Ciągła przestrzeń stanów
- Co jest stanem?[zadanie 7]

$\langle x, \theta, \dot{x}, \dot{\theta} \rangle$

- Jakie akcje są możliwe? [zadanie 8]

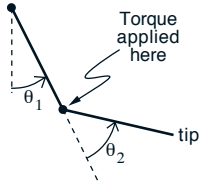
$\{lewo, prawo\}$ (ewentualnie, siła)

- Algorytm Boxes:
  - Dyskretyzacja w „pudełka”
    - Potrzeba jedynie 30 prób uczących, aby balansować przez godzinę
    - Bez symulatora
    - Negatywne wzmocnienie za ostatni (s, a) przed upadkiem.

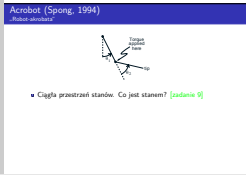


Acrobot (Spong, 1994)

„Robot-akrobata"

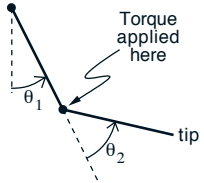


- Ciągła przestrzeń stanów. Co jest stanem? [zadanie 9]



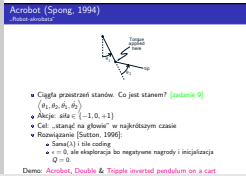
Acrobot (Spong, 1994)

„Robot-akrobata"



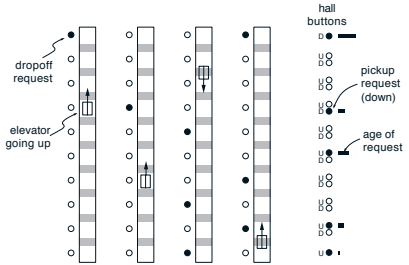
- Ciągła przestrzeń stanów. Co jest stanem? [zadanie 9]
  $\left\langle \theta_1, \theta_2, \dot{\theta_1}, \dot{\theta_2} \right\rangle$
- Akcje: *siła* $\in \{-1, 0, +1\}$
- Cel: „stanąć na głowie" w najkrótszym czasie
- Rozwiązanie [Sutton, 1996]:
  - Sarsa(λ) i tile coding
  - $\epsilon = 0$, ale eksploracja bo negatywne nagrody i inicjalizacja $Q = 0$.

Demo: Acrobot, Double & Tripple inverted pendulum on a cart



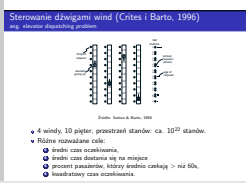
Sterowanie dźwigami wind (Crites i Barto, 1996)

ang. elevator dispatching problem



Źródło: Sutton & Barto, 1998

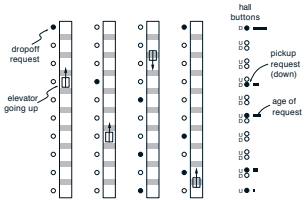
- 4 windy, 10 pięter, przestrzeń stanów: ca. 10^{22} stanów.
- Różne rozważane cele:
  1. średni czas oczekiwania,
  2. średni czas dostania się na miejsce
  3. procent pasażerów, którzy średnio czekają $>$ niż 60s,
  4. kwadratowy czas oczekiwania.



1. A stochastic optimal control problem
2. Przestrzeń stanów jest ciągła (np. pozycja windy), więc podana tutaj liczba stanów jest po (arbitralnej) kwantyzacji.

Sterowanie dźwigami wind (Crites i Barto, 1996)

ang. elevator dispatching problem



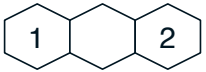
- Przestrzeń akcji?
  - Pewne uproszczenia: każda winda osobno (**multi agent reinforcement learning**)
  - Zdroworozsądkowe ograniczenia
  - Ograniczenia → jedyne akcje: czy zatrzymać się czy nie.

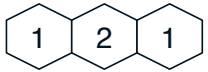
Stan z ciągłym czasem → stan z dyskretnym czasem (**semi-markovski** proces decyzyjny)
Sieć neuronowa: 47 wejść, 20 węzłów ukrytych i 2 wyjścia

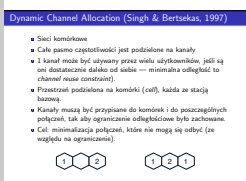
1. 1) winda nie może nie zatrzymać się na piętrze koło, którego przejeżdża, jeśli ktoś chce tam wysiąść.
2. 2) winda nie może zmienić kierunku, jeśli wszyscy pasażerowie, których ma na pokładzie i jadą w tę stronę jeszcze nie wysiedli
3. 3) winda nie może zatrzymać się na piętrze, jeśli nikt tam nie wysiada
4. 4) domyślnie zawsze jedzie do góry

Dynamic Channel Allocation (Singh & Bertsekas, 1997)

- Sieci komórkowe
- Całe pasmo częstotliwości jest podzielone na kanały
- 1 kanał może być używany przez wielu użytkowników, jeśli są oni dostatecznie daleko od siebie — minimalna odległość to *channel reuse constraint*).
- Przestrzeń podzielona na komórki (*cell*), każda ze stacją bazową.
- Kanały muszą być przypisane do komórek i do poszczególnych połączeń, tak aby ograniczenie odległościowe było zachowane.
- Cel: minimalizacja połączeń, które nie mogą się odbyć (ze względu na ograniczenie).







Dynamic Channel Allocation (Singh & Bertsekas, 1997)

- Fixed assignment vs. dynamic assignment
- Stan: 49 komórek i 70 kanałów, to $2^{49\times 70}$ stanów sieci (kanał zajęty/wolny)
- Proces semi-markowski: tranzycje: nowe połączenie oraz (koniec połączenia)
- Akcje: który kanał przypisać przy nowym połączeniu

Dynamic Channel Allocation (Singh & Bertsekas, 1997)
- Fixed assignment vs. dynamic assignment
- Stan: 49 komórek i 70 kanałów, to $2^{49\times 70}$ stanów sieci (kanał zajęty/wolny)
- Proces semi-markowski: tranzycje: nowe połączenie oraz (koniec połączenia)
- Akcje: który kanał przypisać przy nowym połączeniu

1. [Sutton & Barto, 1998] piszą, że mamy tutaj 70^{49} stanów, ale wydaje się, że się mylą.

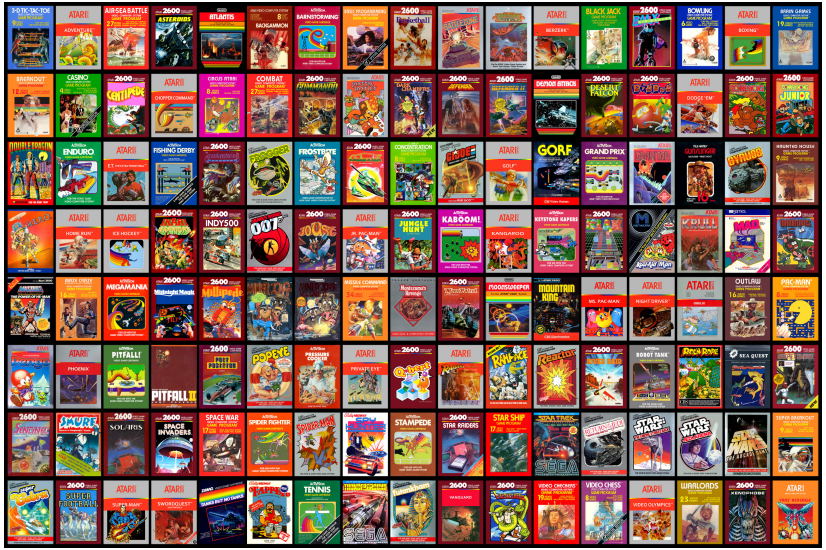
2048 (Szubert & Jaśkowski, 2014)

Prezentacja

2048 (Szubert & Jaśkowski, 2014)
Prezentacja

Atari (Mnih et al., 2015)

Atari (Mnih et al., 2015)



Atari (Mnih et al., 2015)

Atari (Mnih et al., 2015)

Motywacja:

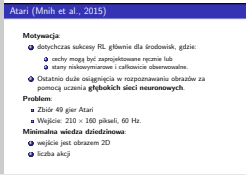
1. dotychczas sukcesy RL głównie dla środowisk, gdzie:
   1. cechy mogą być zaprojektowane ręcznie lub
   2. stany niskowymiarowe i całkowicie obserwowalne.
2. Ostatnio duże osiągnięcia w rozpoznawaniu obrazów za pomocą uczenia **głębokich sieci neuronowych**.

Problem:

- Zbiór 49 gier Atari
- Wejście: 210×160 pikseli, 60 Hz.

Minimalna wiedza dziedzinowa:

1. wejście jest obrazem 2D
2. liczba akcji



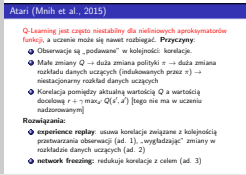
Atari (Mnih et al., 2015)

Q-Learning jest często niestabilny dla nieliniowych aproksymatorów funkcji, a uczenie może się nawet rozbiegać. **Przyczyny**:

1. Obserwacje są „podawane" w kolejności: korelacje.
2. Małe zmiany $Q \rightarrow$ duża zmiana polityki $\pi \rightarrow$ duża zmiana rozkładu danych uczących (indukowanych przez π) $\rightarrow$ niestacjonarny rozkład danych uczących
3. Korelacja pomiędzy aktualną wartością Q a wartością docelową $r + \gamma \max_{a'} Q(s', a')$ [tego nie ma w uczeniu nadzorowanym]

Rozwiązania:

1. **experience replay**: usuwa korelacje związane z kolejnością przetwarzania obserwacji (ad. 1), „wygładzając" zmiany w rozkładzie danych uczących (ad. 2)
2. **network freezing:** redukuje korelacje z celem (ad. 3)



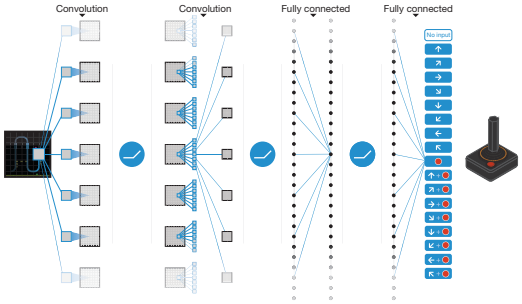
1. Zmiana wartości Q powoduje zmiany wartości docelowej $r + \gamma \max_{a'} Q(s', a')$.

Atari (Mnih et al., 2015)

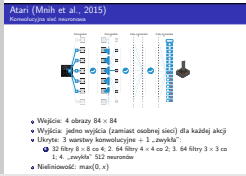
Demo

Atari (Mnih et al., 2015)

Konwolucyjna sieć neuronowa



- Wejście: 4 obrazy 84×84
- Wyjścia: jedno wyjścia (zamiast osobnej sieci) dla każdej akcji
- Ukryte: 3 warstwy konwolucyjne + 1 „zwykła”:
  1. 32 filtry 8×8 co 4; 2. 64 filtry 4×4 co 2; 3. 64 filtry 3×3 co 1; 4. „zwykła” 512 neuronów
- Nieliniowość: $\max(0, x)$



1. W pierwszej warstwie obraz 84×84. Wag: $1 \times 32 \times (8 \times 8) = 2048$.
2. W drugiej, mamy więc 32 kanały 20×20. Wag: $32 \times 64 \times (4 \times 4) = 32768$.
3. W trzeciej: 64 kanały po 9×9. Wag: $64 \times 64 \times (3 \times 3) = 36864$
4. W kolejnej 64 kanały po 7×7, czyli $64 \times 7 \times 7 = 3136$ neuronów wejściowych, które są połączone z 512 neuronami. Czyli $3136 \times 512 = 1,605,632$ wag
5. W ostatniej: $512 \times 18 = 9216$
6. $+$ jeszcze biasy

Atari (Mnih et al., 2015)

Przetwarzanie wstępne:

- Oryginalnie: 210×160 pikseli, 128 wartościowa paleta kolorów
- Max z dwóch kolejnych klatek (redukcja efektu mrugania)
- Decymacja do 84×84.
- Tylko luminancja
- Stan $=$ 4 ostatnie ramki (brak częściowej obserwowalności)

Atari (Mnih et al., 2015)
Przetwarzanie wstępne:
- Oryginalnie: 210×160 pikseli, 128 wartościowa paleta kolorów
- Max z dwóch kolejnych klatek (redukcja efektu mrugania)
- Decymacja do 84×84.
- Tylko luminancja
- Stan $=$ 4 ostatnie ramki (brak częściowej obserwowalności)

Atari (Mnih et al., 2015)

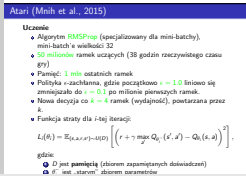
Uczenie

- Algorytm RMSProp (specjalizowany dla mini-batchy), mini-batch'e wielkości 32
- 50 milionów ramek uczących (38 godzin rzeczywistego czasu gry)
- Pamięć: 1 mln ostatnich ramek
- Polityka ϵ-zachłanna, gdzie początkowo $\epsilon = 1.0$ liniowo się zmniejszało do $\epsilon = 0.1$ po milionie pierwszych ramek.
- Nowa decyzja co $k = 4$ ramek (wydajność), powtarzana przez k.
- Funkcja straty dla i-tej iteracji:

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim U(D)}\left[\left(r + \gamma \max_{a'} Q_{\theta_i^-}(s', a') - Q_{\theta_i}(s, a)\right)^2\right],$$

gdzie:

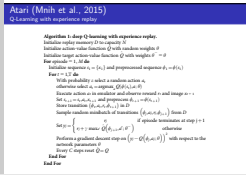
1. D jest **pamięcią** (zbiorem zapamiętanych doświadczeń)
2. θ_i^- jest „starym" zbiorem parametrów



Atari (Mnih et al., 2015)

Q-Learning with experience replay

Algorithm 1: deep Q-learning with experience replay.
Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$
For episode = 1, M **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  For $t = 1$,T **do**
    With probability ε select a random action a_t
    otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^{-}) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $\left(y_j - Q(\phi_j, a_j; \theta)\right)^2$ with respect to the network parameters θ
    Every C steps reset $\hat{Q} = Q$
  End For
End For



Atari (Mnih et al., 2015)

Dodatkowe techniki:

Atari (Mnih et al., 2015)
Dodatkowe techniki:

1. Problem z nagrodami o różnej wielkości

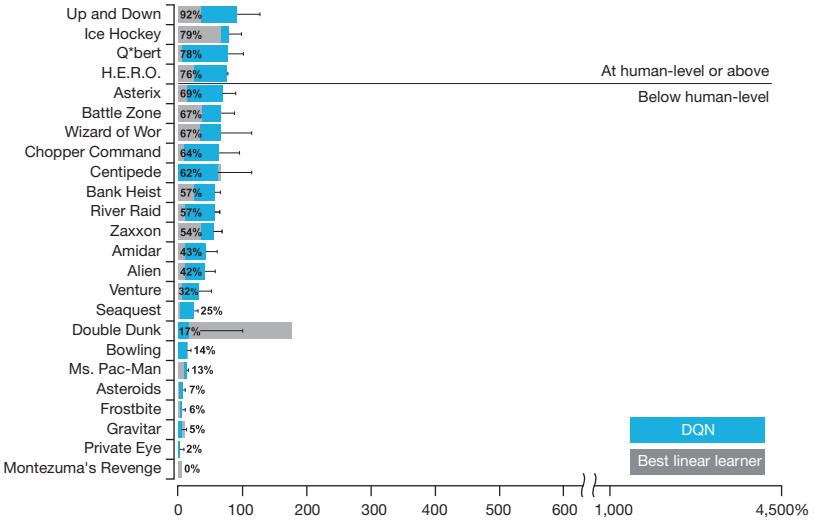
2. Dodatkowa stabilność: funkcja straty ma $abs(x)$ poza obszarem $[-1, 1]$.

Atari (Mnih et al., 2015)

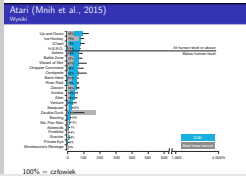
Dodatkowe techniki:

Atari (Mnih et al., 2015)

Wyniki

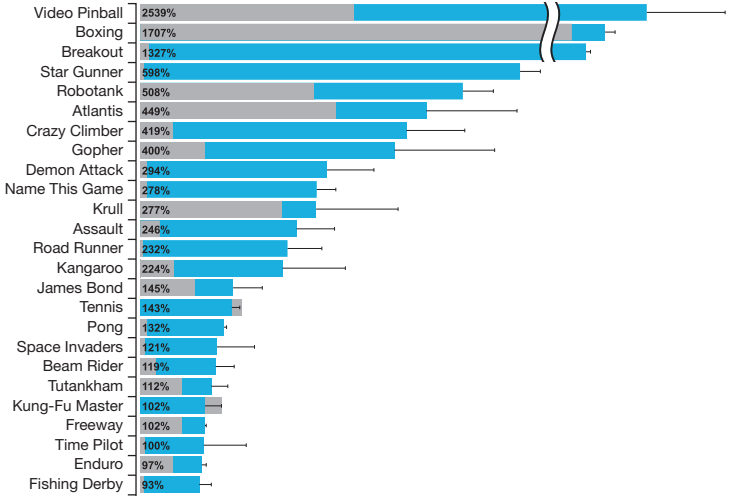


100% = człowiek

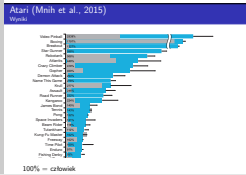


Atari (Mnih et al., 2015)

Wyniki



100% = człowiek



Atari (Mnih et al., 2015)

RMSProp
Parametry: η, $\rho = 0.95$, $\epsilon = 10^{-6}$.

Atari (Mnih et al., 2015)
RMSProp
Parametry: η, $\rho = 0.95$, $\epsilon = 10^{-6}$.

$$g_i^{(t)} = \rho g_i^{(t-1)} + (1-\rho)(\frac{\partial L}{\partial \theta_i}^{(t)})^2$$

$$h_i^{(t)} = \rho h_i^{(t-1)} + (1-\rho)\frac{\partial L}{\partial \theta_i}^{(t)}$$

$$\eta_i^{(t)} = \frac{\eta}{\sqrt{g_i^{(t)} - (h_i^{(t)})^2 + \varepsilon}}$$

$$\theta_i^{(t)} = \theta_i^{(t-1)} - \eta_i^{(t)}\frac{\partial L}{\partial \theta_i}^{(t)}$$

Atari (Mnih et al., 2015)

RMSProp
Parametry: η, $\rho = 0.95$, $\epsilon = 10^{-6}$.

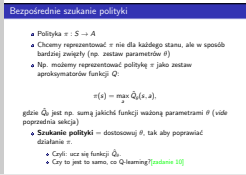
Bezpośrednie szukanie polityki

- Polityka $\pi : S \to A$
- Chcemy reprezentować π nie dla każdego stanu, ale w sposób bardziej zwięzły (np. zestaw parametrów θ)
- Np. możemy reprezentować politykę π jako zestaw aproksymatorów funkcji Q:

$$\pi(s) = \max_a \hat{Q}_\theta(s, a),$$

gdzie $\hat{Q}_\theta$ jest np. sumą jakichś funkcji ważoną parametrami θ (*vide* poprzednia sekcja)

- **Szukanie polityki** = dostosowuj θ, tak aby poprawiać działanie π.
  - Czyli: ucz się funkcji $\hat{Q}_\theta$.
  - Czy to jest to samo, co Q-learning?[zadanie 10]

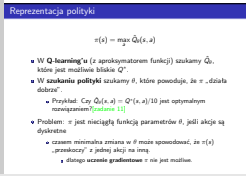


1. Nie

Reprezentacja polityki

$$\pi(s) = \max_a \hat{Q}_\theta(s, a)$$

- W **Q-learning'u** (z aproksymatorem funkcji) szukamy $\hat{Q}_\theta$, które jest możliwie bliskie Q^*.
- W **szukaniu polityki** szukamy θ, które powoduje, że π „działa dobrze".
  - Przykład: Czy $\hat{Q}_\theta(s, a) = Q^*(s, a)/10$ jest optymalnym rozwiązaniem?[zadanie 11]
- Problem: π jest nieciągłą funkcją parametrów θ, jeśli akcje są dyskretne
  - czasem minimalna zmiana w θ może spowodować, że $\pi(s)$ „przeskoczy" z jednej akcji na inną.
    - dlatego **uczenie gradientowe** π nie jest możliwe.

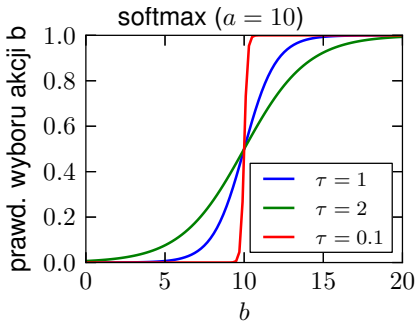


1. Tak. Mimo, że $\hat{Q}_\theta(s, a) \neq Q^*(s, a)$, bo liczy się $\max_a$

Polityka stochastyczna

- Dlatego używa się **polityki stochastycznej** $\pi_\theta(s, a)$, reprezentującej prawd. wybrania akcji a w stanie s.
- Reprezentacja z użyciem **funkcji softmax**:

$$\pi_\theta(s, a) = e^{\hat{Q}_\theta(s,a)/\tau} / \sum_{a'} e^{\hat{Q}_\theta(s,a')/\tau}$$



- Zaleta: różniczkowalna