Inteligentne Systemy Interaktywne

Piotr Duch

pduch@iis.p.lodz.pl
Instytut Informatyki Stosowanej
Politechnika Łódzka

Lato 2020

Plan wykładu

Informacje ogólne:

- Materiały wykładowe oraz laboratoryjne dostępne są na githubie (https://github.com/iis-siium/ISI).
- Literatura podstawowa:
  - Richard S. Sutton, and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
  - Csaba Szepesvári. *Algorithms for reinforcement learning*. Morgan and Claypool. 2009.
- Wykłady uzupełniające:
  - RL Course by David Silver - https://www.youtube.com
  - CS 188: Artificial Intelligence by Pieter Abbeel (wykład 10 i 11)- https://www.youtube.com/watch?v=IXuHxkpO5E8
- Materiały dodatkowe:
  - Practical RL Course by Yandex School of Data Analysis - https://github.com/yandexdataschool/Practical_RL
  - CS 188: Introduction to Artificial Intelligence by Berkeley University of California - https://inst.eecs.berkeley.edu/ cs188/fa19/project3/

Uczenie ze wzmocnieniem
Wprowadzenie

Uczenie pasywne

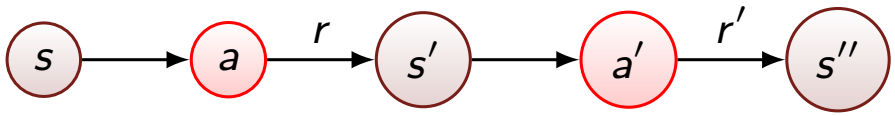
(ang. *model based learning*)

Uczenie aktywne

(ang. *model free learning*)

Uczenie aktywne

Co zrobić, jeżeli nie dysponujemy modelem środowiska?



Sekwencja:

- stany (s),
- akcje (a),
- nagrody (r).

Uczenie aktywne

Algorytmy:

- Monte Carlo.
- Metody różnic tymczasowych (ang. *Temporal Difference learning*):
  - Q-learning,
  - Sarsa.

Uczenie aktywne

Monte Carlo

Cechy algorytmu:

- Algorytm przeznaczony do zadań epizodycznych.
- Nie wymaga modelu środowiska.
- Uczy się na podstawie doświadczenie (ang. *experience*) - sekwencji stan, akcja, nagroda.

Uczenie aktywne

Monte Carlo

Cechy algorytmu:

- Algorytm przeznaczony do zadań epizodycznych.
- Nie wymaga modelu środowiska.
- Uczy się na podstawie doświadczenie (ang. *experience*) - sekwencji stan, akcja, nagroda.

Wersje algorytmu:

- Pierwsza wizyta (ang. *First-visit Monte Carlo*).
- Każda wizyta (ang. *Every-visit Monte Carlo*).

Uczenie aktywne

Monte Carlo

First-visit Monte Carlo method - oszacowanie $V \approx v_\pi$

Wejście: strategia π, która ma być oszacowana.
Inicjalizacja:

- $V(s) \in \mathbb{R}$ - losowe wartości, dla każdego $s \in S$,
- $Returns(s)$ - puste listy, dla każdego $s \in S$.

Nisekończona pętla (dla każdego epizodu):

Wygeneruj sekwencję przejść dla epizodu zgodnie ze strategią π:
$s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_{T-1}, a_{T-1}, r_T$
$G \leftarrow 0$:
Dla każdego kroku w epizodzie, $t = T-1, T-2, \ldots, 0$:
$G \leftarrow \gamma G + r_{t+1}$
Jeżeli stan s_t nie pojawił się wcześniej:
Dodaj G do listy $Returns(s_t)$
$V(s_t) \leftarrow \text{average}(Returns(s_t))$

Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2018.

Uczenie aktywne

Monte Carlo

First-visit Monte Carlo prediciton - for estimating $V \approx v_\pi$

Input: a policy π to be evaluated
Initialize:

- $V(s) \in \mathbb{R}$, arbitrarily, for all $s \in S$,
- $Returns(s) \leftarrow$ an empty list, for all $s \in S$.

Loop forever (for each episode):
Generate an episode following π: $s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_{T-1}, a_{T-1}, r_T$
$G \leftarrow 0$:
Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
$G \leftarrow \gamma G + r_{t+1}$
Unless s_t appears in $s_0, s_1, \ldots, s_{t+1}$:
Append G to $Returns(s_t)$
$V(s_t) \leftarrow$ average$(Returns(s_t))$

Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2018.

Uczenie aktywne

Monte Carlo

Co się bardziej przyda:

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

Uczenie aktywne

Monte Carlo

Metoda Monte Carlo, zmodyfikowana tak, aby wyznaczała $q_\pi(s, a)$ zamiast $v(s)$ będzie wyglądała analogicznie do tej, przedstawionej wcześniej.

Odwiedzony stan będzie określany za pomocą pary stan (s) - akcja wybrana w dany stanie (a).

Metoda *every-visit Monte Carlo* oszacuje wartość w danym stanie jako średnią oczekiwanych nagród ze wszystkich wizyt w danym stanie.

Metoda *first-visit Monte Carlo* oszacuje wartość w danym stanie jako nagrodę otrzymaną przy okazji pierwszej wizyty w danym stanie.

Uczenie aktywne

Monte Carlo

Jak rozwiązać problem nieodwiedzanych stanów:

- eksploracja stanów początkowych (ang. *exploring starts*):
  - wybieramy losowy stan i akcję, dla których rozpoczynamy epizod,
  - nierealistyczne w rzeczywistym świecie, za wyjątkiem symulacji,
- algorytm ϵ-zachłanny (ang. ϵ-*greedy*):
  - wybieramy najlepszą akcję z prawdopodobieństwem $1 - \epsilon + \frac{\epsilon}{|A(s)|}$,
  - wybieramy losową akcję z prawdopodobieństwem $\frac{\epsilon}{|A(s)|}$.

Uczenie aktywne

Monte Carlo

First-visit Monte Carlo method (for ϵ-soft policies) - oszacowanie $\pi \approx \pi_*$

Parametry algorytmu: mała wartość $\epsilon > 0$
Inicjalizacja:

- π losowa ϵ-miękka strategia,
- $Q(s, a) \in \mathbb{R}$ (losowe), dla każdej pary $s \in S$, $a \in A(s)$,
- $Returns(s, a) \leftarrow$ pusta lista, dla każdej pary $s \in S$, $a \in A(s)$.

Pętla nieskończona (dla każdego epizodu):
Wygeneruj sekwencję przejść dla epizodu zgodnie ze strategią π:
$s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_{T-1}, a_{T-1}, r_T$
$G \leftarrow 0$:
Dla każdego kroku w epizodzie, $t = T-1, T-2, \ldots, 0$:
$G \leftarrow \gamma G + r_{t+1}$
Jeżeli para s_t, a_t niepojawiła się wcześniej w sekwencji $s_0, a_0, s_1, a_1, \ldots, s_{t+1}, a_{t+1}$:
Dodaj G do listy $Returns(s_t, a_t)$
$Q(S_t, A_t) \leftarrow \text{average}(Returns(s_t, a_t))$
$a^* \leftarrow \text{argmax}_a Q(s_t, a)$
Dla każdej akcji $a \in A(s_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \epsilon + \frac{\epsilon}{|A(s)|} & \text{if} \quad a = a^* \\ \frac{\epsilon}{|A(s)|} & \text{if} \quad a \neq a^* \end{cases} \tag{1}$$

Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2018.

Uczenie aktywne

Monte Carlo

First-visit Monte Carlo method (for ϵ-soft policies) - estimates $\pi \approx \pi_*$

Algorithm parameter: small $\epsilon > 0$
Initialize:

- π an arbitrary ϵ-soft policy,
- $Q(s, a) \in \mathbb{R}$ (arbitrarily), for all $s \in S$, $a \in A(s)$,
- $Returns(s, a) \leftarrow$ an empty list, for all $s \in S$, $a \in A(s)$.

Loop forever (for each episode):
Generate an episode following π: $s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_{T-1}, a_{T-1}, r_T$
$G \leftarrow 0$:
Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
$G \leftarrow \gamma G + r_{t+1}$
Unless the pair s_t, a_t appears in $s_0, a_0, s_1, a_1, \ldots, s_{t+1}, a_{t+1}$:
Append G to $Returns(s_t, a_t)$
$Q(s_t, a_t) \leftarrow \text{average}(Returns(s_t, a_t))$
$a^* \leftarrow \text{argmax}_a Q(s_t, a)$
For all $a \in A(s_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \epsilon + \frac{\epsilon}{|A(s)|} & \text{if} \quad a = a^* \\ \frac{\epsilon}{|A(s)|} & \text{if} \quad a \neq a^* \end{cases} \tag{2}$$

Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2018.

Uczenie aktywne

Metody różnic tymczasowych (ang. *Temporal-Difference (TD) Learning*)

Metody różnic tymczasowych:

- Kombinacja metody Monte Carlo i Programowania Dynamicznego.
- Nie wymagają znajomości modelu środowiska.
- Uaktualnianie przewidywanych wartości następuje natychmiastowo - nie ma koniczeności oczekiwania na zakończenie epizodu.

Uczenie aktywne

Metody różnic tymczasowych (ang. *Temporal-Difference (TD) Learning*)

Szacowanie funkcji wartości za pomocą metod Monte Carlo:

$$V(s_t) \leftarrow V(s_t) + \alpha[G_t - V(s_t)] \tag{3}$$

Szacowanie funkcji wartości za pomocą metod różnic tymczasowych:

$$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)] \tag{4}$$

Uczenie aktywne

Metody różnic tymczasowych (ang. *Temporal-Difference (TD) Learning*)

Tabelaryczny algorytm różnic tymczasowych z krokiem 1 do oszacowania v_π

Wejście: strategia do oszacowania π
Parametr algorytmu: krok uczenia $\alpha \in (0, 1]$
Inicjalizacja tablicy wartości stanów $V(s)$ losowymi wartościami, za wyjątkiem stanu końcowego, któremu przypisana jest wartość 0.
Pętla dla każdego epizodu:
    Inicjalizacja s
    Dla każdego kroku w epizodzie:
        Wybierz ackję a zgodnie ze strategią π dla stanu s
        Wykonaj akcję a i zaobserwuj r oraz s'
        $V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$
        $s \leftarrow s'$
    Dopóki s nie jest stanem końcowym

Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2018.

Uczenie aktywne

Metody różnic tymczasowych (ang. *Temporal-Difference (TD) Learning*)

Tabular TD(0) for estimating v_π

Input: the policy π to be evaluated
Algorithm parameter: step size $\alpha \in (0, 1]$
Initialize $V(s)$, for all $s \in S^+$, arbitrarily except that $V(terminal) = 0$
Loop for each episode:
    Initialize s
    Loop for each step of episode:
        $a \leftarrow$ action given by π for s
        Take action a, observe r, s'
        $V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$
        $s \leftarrow s'$
    Until s is not terminal

Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2018.

Uczenie aktywne

Metody różnic tymczasowych (ang. *Temporal-Difference (TD) Learning*)

Błąd:

$$\delta_t \doteq r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \tag{5}$$

Uczenie aktywne

Metody różnic tymczasowych (ang. *Temporal-Difference (TD) Learning*)

Metody różnic tymczasowych nie wymagają znajomości modelu środowiska.

Obliczenia wykonywane są online - brak konieczności oczekiwania na koniec epizodu.

Dla dowolnej stałej strategii π, udowodnione zostało, że metody TD(0) są zbieżne do v_π, w przypadku kiedy wartość parametru uczącego (α) jest stała i dostatecznie mała lub gdy wartość tego parametru zmniejsza się.

Uczenie aktywne

Q-Learning

Cechy algorytmu Q-Learning:

- uczy się nie tylko na podstawie swojego doświadczenia, ale także innych ludzi / agentów,
- korzysta z optymalnej strategii nawet w trakcie eksploracji,
- korzysta z wielu strategii podążając tylko jedną.

Uczenie aktywne

Q-Learning

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_{t+1} + \gamma max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

Uczenie aktywne

Q-Learning

Wartość dla strategii π^* - optymalnej strategii

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \, max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

Uczenie aktywne

Q-Learning

Wartość dla strategii π^* - optymalnej strategii

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \, max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

Nagroda otrzymana po wykonaniu akcji a_t w stanie s_t

Uczenie aktywne

Q-Learning

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

$$Q(s_t, a_t) = (1 - \alpha)Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a)]$$

Uczenie aktywne

Q-Learning

Algorytm Q-Learning do wyznaczenia strategii $\pi \approx \pi_*$

Parametry algorytmu: krok uczenia $\alpha \in (0, 1]$, $\epsilon > 0$ o małej wartości
Inicjalizacja tablicy $Q(s, a)$, dla każdego stanu $s \in S$ i akcji w tym stanie $a \in A(s)$, losowymi wartościami oprócz stanu końcowego $Q(terminal, .) = 0$
Pętla po wszystkich epizodach:
    Inicjalizacja s
    Dla każdego kroku w epizodzie:
        Wybierz akcję a w stanie s wykorzystując strategię opartą o tablicę Q (np., ϵ-zachłanną)
        Wykonaj akcję a i zaobserwuj r oraz s'
        $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma max_a Q(s', a) - Q(s, a)]$
        $s \leftarrow s'$
    Dopóki s nie jest stanem końcowym

Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2018.

Uczenie aktywne

Q-Learning

Q-Learning for estimating $\pi \approx \pi_*$

Algorithm parameter: step size $\alpha \in (0, 1]$, small $\epsilon > 0$
Initialize $Q(s, a)$, for all $s \in S$, $a \in A(s)$, arbitrarily except that $Q(terminal, .) = 0$
Loop for each episode:
    Initialize s
    Loop for each step of episode:
        Choose a from s using policy derived from Q (e.g., ϵ-greedy)
        Take action a, observe r, s'
        $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma max_a Q(s', a) - Q(s, a)]$
        $s \leftarrow s'$
    Until s is not terminal

Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2018.

Uczenie aktywne

Q-Learning

Algorytm ϵ-zachłanny:

$$a = \begin{cases} \text{argmax}_a Q(s,.) & \text{z prawdopodobieństwem} \quad 1-\epsilon \ * \\ \text{losowa akcja} & \text{z prawdopodobieństwem} \quad \epsilon \end{cases} \tag{6}$$

* w przypadku kilku akcji z taką samą wartością należy wybierać **losową**

Uczenie aktywne

Q-Learning - przykład liczbowy

Nowe środowisko:

Aktualizowanie funkcji wartości dla pary stan-akcja (s_t, a_t):

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_{t+1} + \gamma max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

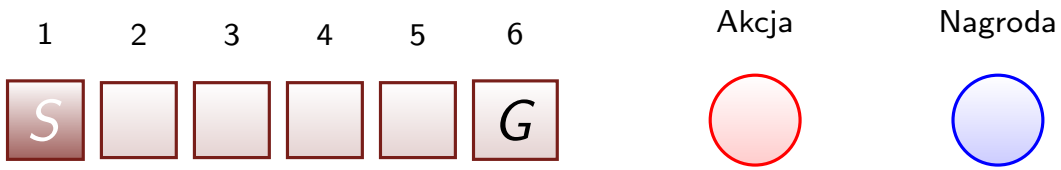
Parametry algorytmu:

- $\alpha = 0.1$,
- $\gamma = 0.9$,
- $r_G = 1$, w pozostałych przypadkach $r = 0$.

Uczenie aktywne

Q-Learning - przykład liczbowy cd.

Epizod 1:



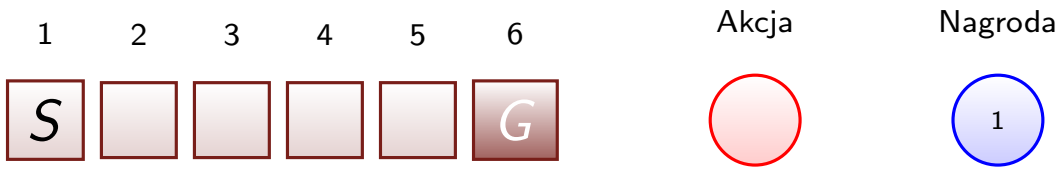
$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_{t+1} + \gamma max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

Stan	L	P
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0

Uczenie aktywne

Q-Learning - przykład liczbowy cd.

Epizod 1:



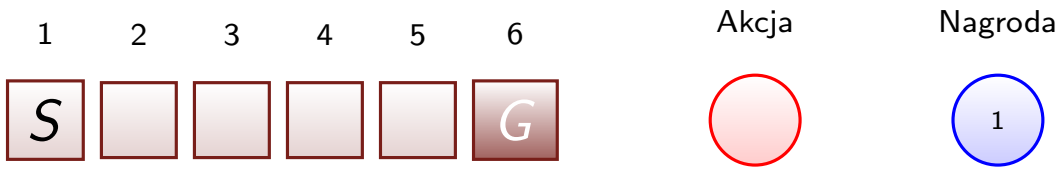
$$Q(5,P) = Q(5,P) + \alpha[r + \gamma max_a Q(6,a) - Q(5,P)]$$

Stan	L	P
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0

Uczenie aktywne

Q-Learning - przykład liczbowy cd.

Epizod 1:



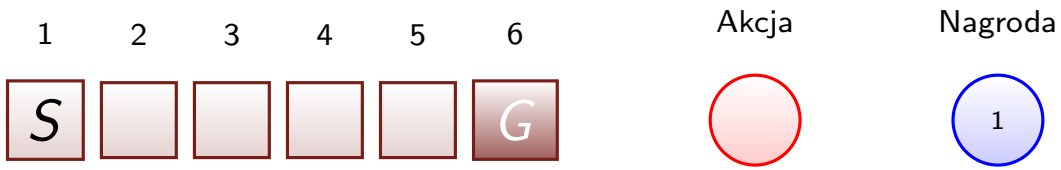
$$Q(5, P) = 0 + 0.1[1 + 0.9 * 0 - 0] = 0.1$$

Stan	L	P
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0

Uczenie aktywne

Q-Learning - przykład liczbowy cd.

Epizod 1:



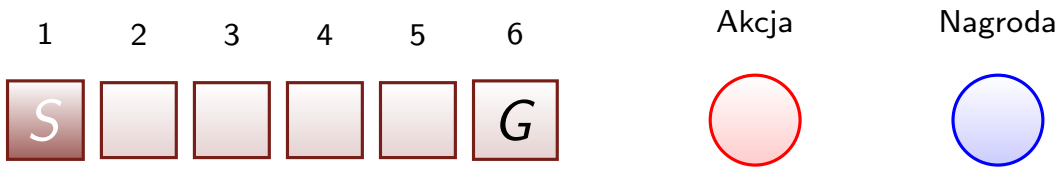
$$Q(5, P) = 0 + 0.1[1 + 0.9 * 0 - 0] = 0.1$$

Stan	L	P
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0.1
6	0	0

Uczenie aktywne

Q-Learning - przykład liczbowy cd.

Epizod 2:



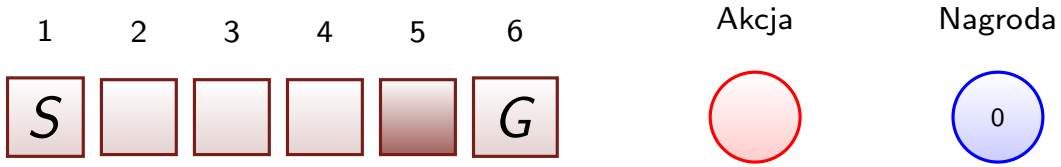
$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_{t+1} + \gamma max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

Stan	L	P
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0.1
6	0	0

Uczenie aktywne

Q-Learning - przykład liczbowy cd.

Epizod 2:



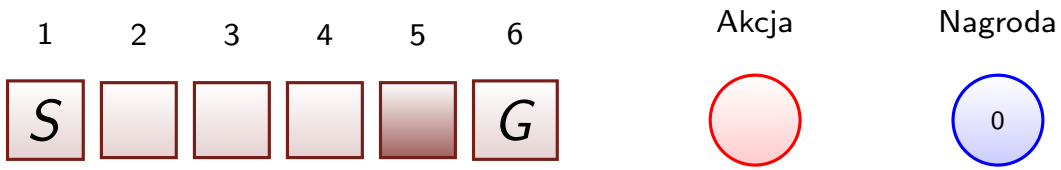
$$Q(4,P) = Q(4,P) + \alpha[r + \gamma max_a Q(5,a) - Q(4,P)]$$

Stan	L	P
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0.1
6	0	0

Uczenie aktywne

Q-Learning - przykład liczbowy cd.

Epizod 2:



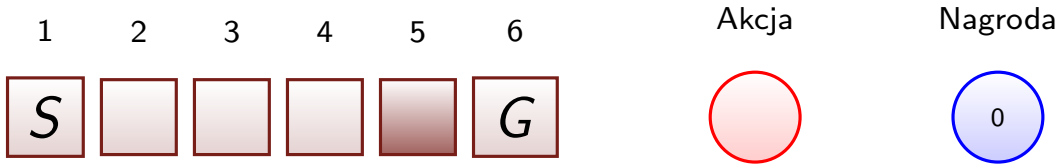
$$Q(4, P) = 0 + 0.1[0 + 0.9 * 0.1 - 0] = 0.009$$

Stan	L	P
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0.1
6	0	0

Uczenie aktywne

Q-Learning - przykład liczbowy cd.

Epizod 2:



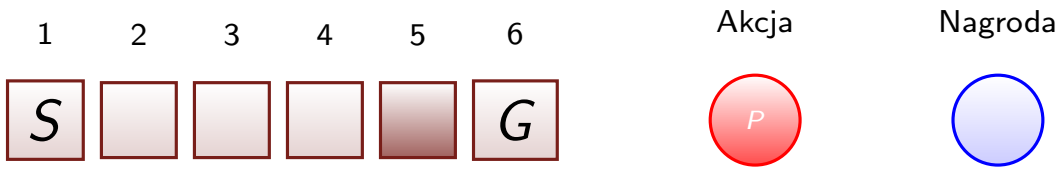
$$Q(4, P) = 0 + 0.1[0 + 0.9 * 0.1 - 0] = 0.009$$

Stan	L	P
1	0	0
2	0	0
3	0	0
4	0	0.009
5	0	0.1
6	0	0

Uczenie aktywne

Q-Learning - przykład liczbowy cd.

Epizod 2:



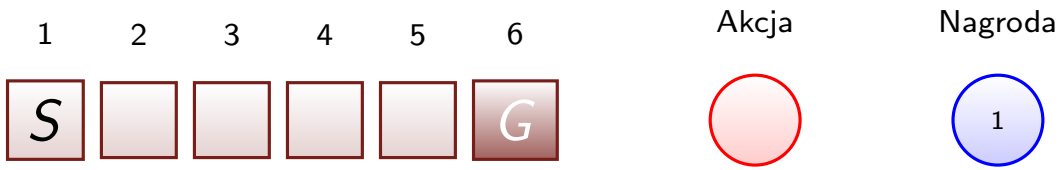
$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_{t+1} + \gamma max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

Stan	L	P
1	0	0
2	0	0
3	0	0
4	0	0.009
5	0	0.1
6	0	0

Uczenie aktywne

Q-Learning - przykład liczbowy cd.

Epizod 2:



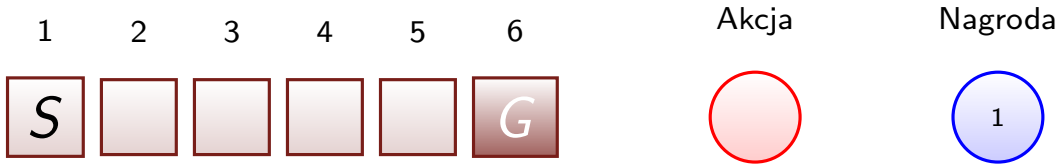
$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_{t+1} + \gamma max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

Stan	L	P
1	0	0
2	0	0
3	0	0
4	0	0.009
5	0	0.1
6	0	0

Uczenie aktywne

Q-Learning - przykład liczbowy cd.

Epizod 2:



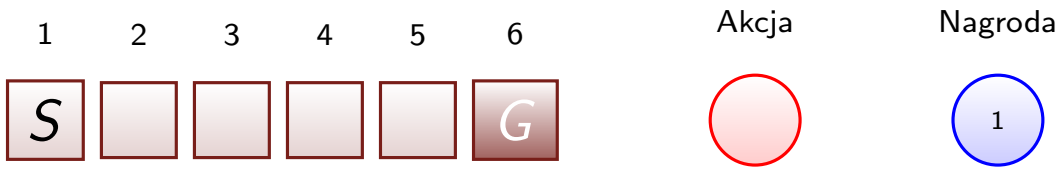
$$Q(5,P) = Q(5,P) + \alpha[r + \gamma max_a Q(6,a) - Q(5,P)]$$

Stan	L	P
1	0	0
2	0	0
3	0	0
4	0	0.009
5	0	0.1
6	0	0

Uczenie aktywne

Q-Learning - przykład liczbowy cd.

Epizod 2:



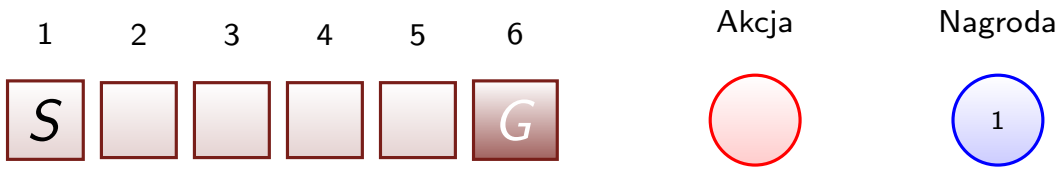
$$Q(5, P) = 0.1 + 0.1[1 + 0.9 * 0 - 0.1] = 0.19$$

Stan	L	P
1	0	0
2	0	0
3	0	0
4	0	0.009
5	0	0.1
6	0	0

Uczenie aktywne

Q-Learning - przykład liczbowy cd.

Epizod 2:

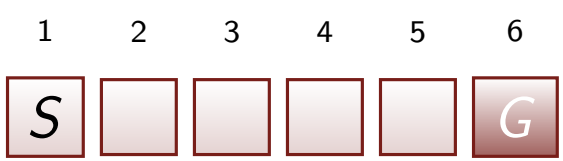


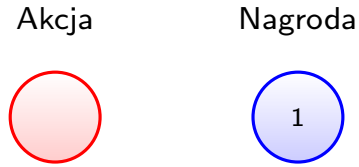
$$Q(5, P) = 0.1 + 0.1[1 + 0.9 * 0 - 0.1] = 0.19$$

Stan	L	P
1	0	0
2	0	0
3	0	0
4	0	0.009
5	0	0.19
6	0	0

Uczenie aktywne

Q-Learning - przykład liczbowy cd.





Epizod 3

Stan	L	P
1	0	0
2	0	0
3	0	0.00081
4	0	0.02520
5	0	0.27100
6	0	0

Epizod 4

Stan	L	P
1	0	0
2	0	0.00007
3	0	0.00300
4	0	0.04707
5	0	0.34390
6	0	0

Uczenie aktywne

SARSA

Przykład algorytmu *On-Policy*.

Do aktualizacji wartości funkcji w stanie (s_t, a_t) używana jest wartość z następnego stanu dla akcji, która później rzeczywiście będzie wykonana (s_{t+1}, a_{t+1}).

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

Uczenie aktywne

SARSA

Algorytm SARSA do wyznaczenia strategii $\pi \approx \pi_*$

Parametry algorytmu: krok uczenia $\alpha \in (0, 1]$, $\epsilon > 0$ o małej wartości
Inicjalizacja tablicy $Q(s, a)$, dla każdego stanu $s \in S$ i akcji w tym stanie $a \in A(s)$, losowymi wartościami oprócz stanu końcowego $Q(terminal, .) = 0$
Pętla po wszystkich epizodach:
    Inicjalizacja s
    Wybierz akcję a w stanie s wykorzystując strategię opartą o tablicę Q (np., ϵ-zachłanną)
    Dla każdego kroku w epizodzie:
        Wykonaj akcję a i zaobserwuj r oraz s'
        Wybierz akcję a' w stanie s' wykorzystując strategię opartą o tablicę Q (np., ϵ-zachłanną)
        $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$
        $s \leftarrow s'$, $a \leftarrow a'$
    Dopóki s nie jest stanem końcowym

Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2018.

Uczenie aktywne

SARSA

SARSA for estimating $\pi \approx \pi_*$

Algorithm parameter: step size $\alpha \in (0, 1]$, small $\epsilon > 0$
Initialize $Q(s, a)$, for all $s \in S$, $a \in A(s)$, arbitrarily except that $Q(terminal, .) = 0$
Loop for each episode:
    Initialize s
    Choose a from s using policy derived from Q (e.g., ϵ-greedy)
    Loop for each step of episode:
        Take action a, observe r, s'
        Choose a' from s' using policy derived from Q (e.g., ϵ-greedy)
        $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$
        $s \leftarrow s'$, $a \leftarrow a'$
    Until s is not terminal

Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2018.

Uczenie aktywne

Expected SARSA

Przykład algorytmu *On-Policy*.

Do aktualizacji wartości funkcji w stanie $(s_t,\ a_t)$ używana jest oczekiwana wartość z następnego stanu obliczona zgodnie z założoną strategią.

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \sum_a \pi(a|s_{t+1}) Q(s_{t+1}, a) - Q(s_t, a_t)]$$

Uczenie aktywne

SARSA(λ)

Połączenie algorytmu Monte Carlo oraz SARSA.

"Śledzenie" odwiedzonych stanów oraz aktualizacja wartości wszystkich odwiedzonych stanów w każdym kroku.

$E_t(s, a)$ - ślad w dla pary stan - akcja w chwili czasowej t.

$Q_{t+1}(s, a) = Q_t(s, a) + \alpha\delta_t E_t(s, a)$.
$\delta_t = r_{t+1}\gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t)$.

Uczenie aktywne

SARSA(λ)

Algorytm SARSA(λ) do wyznaczenia strategii $\pi \approx \pi_*$

Parametry algorytmu: krok uczenia $\alpha \in (0, 1]$, $\epsilon > 0$ o małej wartości, $\lambda \in [0, 1]$
Inicjalizacja tablicy $Q(s, a)$, dla każdego stanu $s \in S$ i akcji w tym stanie $a \in A(s)$, losowymi wartościami oprócz stanu końcowego $Q(terminal, .) = 0$
Pętla po wszystkich epizodach:
    Inicjalizacja s oraz $E(s, a) = 0$, dla każdego stanu $s \in S$ i akcji w tym stanie $a \in A(s)$
    Wybierz akcję a w stanie s wykorzystując strategię opartą o tablicę Q (np., ϵ-zachłanną)
    Dla każdego kroku w epizodzie:
        Wykonaj akcję a i zaobserwuj r oraz s'
        Wybierz akcję a' w stanie s' wykorzystując strategię opartą o tablicę Q (np., ϵ-zachłanną)
        $\delta \leftarrow r + \gamma Q(s', a') - Q(s, a)$
        $E(s, a) = E(s, a) + 1$
        Dla każdeg $s \in S$, $a \in A(s)$:
            $Q(s, a) \leftarrow Q(s, a) + \alpha\delta E(s, a)$
            $E(s, a) \leftarrow \gamma\lambda E(s, a)$
        $s \leftarrow s'$, $a \leftarrow a'$
    Dopóki s nie jest stanem końcowym

Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2015.

Uczenie aktywne

SARSA(λ)

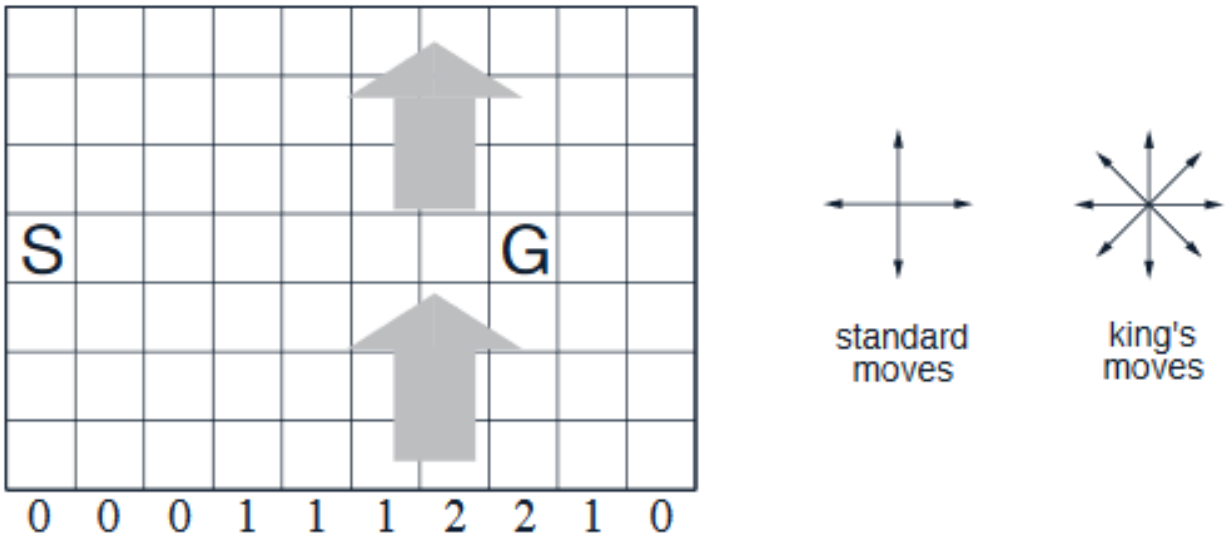
SARSA(λ) for estimating $\pi \approx \pi_*$

Algorithm parameter: step size $\alpha \in (0, 1]$, small $\epsilon > 0$, $\lambda \in [0, 1]$
Initialize $Q(s, a)$, for all $s \in S$, $a \in A(s)$, arbitrarily except that $Q(terminal, .) = 0$
Loop for each episode:
    $E(s, a) = 0$, for all $s \in S$, $a \in A(s)$
    Initialize s
    Choose a from s using policy derived from Q (e.g., ϵ-greedy)
    Loop for each step of episode:
        Take action a, observe r, s'
        Choose a' from s' using policy derived from Q (e.g., ϵ-greedy)
        $\delta \leftarrow r + \gamma Q(s', a') - Q(s, a)$
        $E(s, a) = E(s, a) + 1$
        For all $s \in S$, $a \in A(s)$:
            $Q(s, a) \leftarrow Q(s, a) + \alpha\delta E(s, a)$
            $E(s, a) \leftarrow \gamma\lambda E(s, a)$
        $s \leftarrow s'$, $a \leftarrow a'$
    Until s is not terminal

Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2015.

Uczenie aktywne

Model środowiska

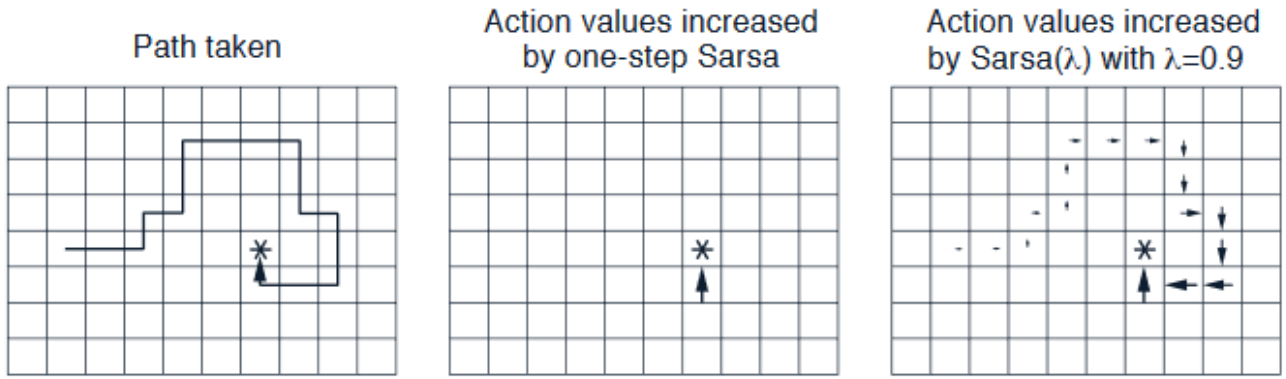


Rysunek 1: Windy Gridworld

Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2015.

Uczenie aktywne

SARSA(λ)



Rysunek 2: Porównanie działania algorytmów SARSA i SARSA(λ)

Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2015.

Uczenie aktywne

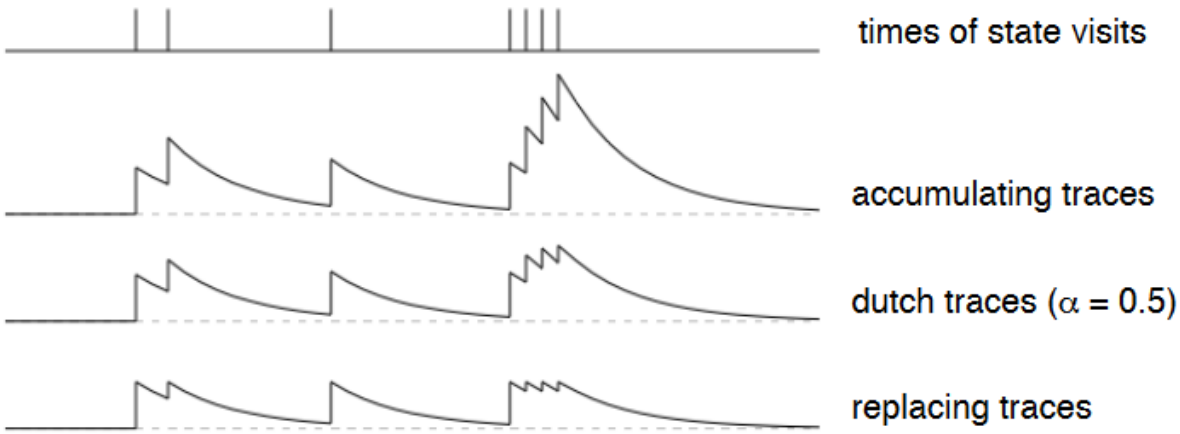
SARSA(λ)

Strategie aktualizacji śladu:

- $E_t(s,a) = \gamma\lambda E_{t-1}(s,a) + 1$ - ang. *accumulating traces*,
- $E_t(s,a) = 1$ - ang. *replacing traces*,
- $E_t(s,a) = (1-\alpha)\gamma\lambda E_{t-1}(s,a) + 1$ - ang. *dutch traces*.

Uczenie aktywne

SARSA(λ)

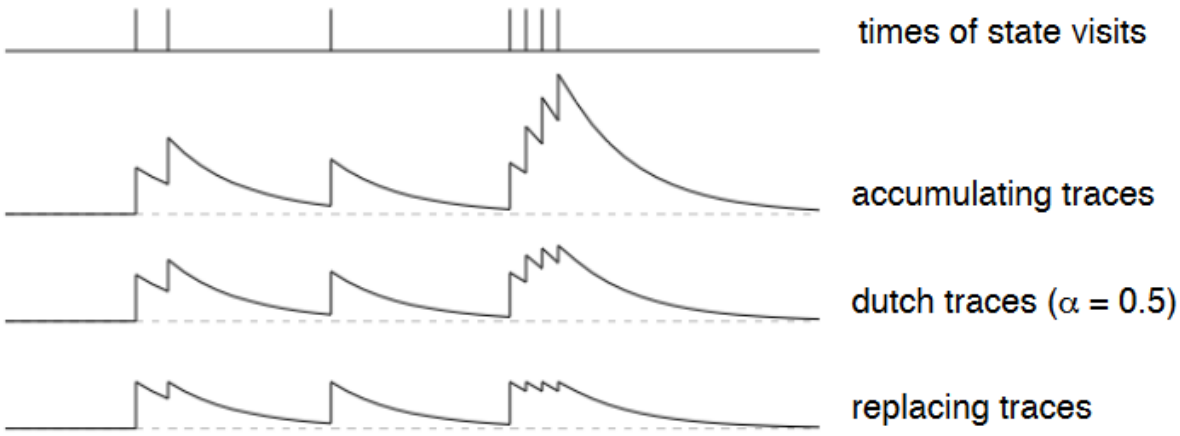


Rysunek 3: Porównanie strategii aktualizacji śladu

Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2015.

Uczenie aktywne

SARSA(λ)



Rysunek 4: Porównanie strategii aktualizacji śladu

Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2015.

Uczenie aktywne

Maximization Bias

Aktualizowanie funkcji wartości dla pary stan-akcja (s_t, a_t):

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_{t+1} + \gamma max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

- Użycie maksimum dla kolejnego stanu do aktualizacji wartości funkcji może prowadzić do nadmiernie optymistycznego przeszacowania wartości.
- $\mathbb{E}_{s'}(max_{a'}(Q(s_{t+1}, a'))) \geqslant max_{a'}(\mathbb{E}_{s'}(Q(s_{t+1}, a')))$
- Problem ten jest nazywany *Maximization Bias*.

Uczenie aktywne

Double Q-Learning

Rozwiązanie problemu:

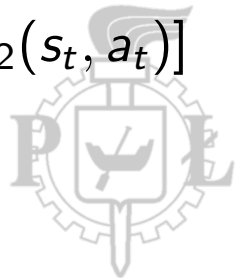
Uczenie oddzielnie dwóch funkcji Q - Q_1 i Q_2.

Aktualizacja wartości funkcji Q_1 na podstawie wartości funkcji Q_2:

$$Q_1(s_t, a_t) = Q_1(s_t, a_t) + \alpha[r_{t+1} + \gamma Q_2(s_{t+1}, argmax_a(Q_1(s_{t+1}, a))) - Q_1(s_t, a_t)]$$

Aktualizacja wartości funkcji Q_2 na podstawie wartości funkcji Q_1:

$$Q_2(s_t, a_t) = Q_2(s_t, a_t) + \alpha[r_{t+1} + \gamma Q_1(s_{t+1}, argmax_a(Q_2(s_{t+1}, a))) - Q_2(s_t, a_t)]$$



Uczenie aktywne

Double Q-Learning

Algorytm Double Q-Learning do szacowania $Q_1 \approx Q_2 \approx q_*$

Parametry algorytmu: krok uczenia $\alpha \in (0, 1]$, $\epsilon > 0$ o małej wartości
Inicjalizacja tablic $Q_1(s, a)$ i $Q_2(s, a)$, dla każdego stanu $s \in S$ i akcji w tym stanie $a \in A(s)$, losowymi wartościami oprócz stanu końcowego $Q(terminal, .) = 0$
Pętla po wszystkich epizodach:
    Inicjalizacja s
    Dla każdego kroku w epizodzie:
        Wybierz akcję a w stanie s wykorzystując strategię ϵ-zachłanną dla $Q_1 + Q_2$
        Wykonaj akcję a i zaobserwuj r oraz s'
        Z prawdopodobieństwem 0.5 aktualizuj:

$$Q_1(s, a) = Q_1(s, a) + \alpha[r + \gamma Q_2(s', argmax_a(Q_1(s', a))) - Q_1(s, a)]$$

        lub:

$$Q_2(s, a) = Q_2(s, a) + \alpha[r + \gamma Q_1(s', argmax_a(Q_2(s', a))) - Q_2(s, a)]$$

        $s \leftarrow s'$
    Dopóki s nie jest stanem końcowym

Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2018.

Uczenie aktywne

Double Q-Learning

Double Q-Learning, for estimating $Q_1 \approx Q_2 \approx q_*$

Algorithm parameter: step size $\alpha \in (0, 1]$, small $\epsilon > 0$
Initialize $Q_1(s, a)$ and $Q_2(s, a)$, for all $s \in S$, $a \in A(s)$, arbitrarily except that $Q(terminal, .) = 0$
Loop for each episode:
Initialize s
Loop for each step of episode:
Choose a from s using the policy ϵ-greedy in $Q_1 + Q_2$
Take action a, observe r, s'
With 0.5 probability:

$$Q_1(s, a) = Q_1(s, a) + \alpha[r + \gamma Q_2(s', argmax_a(Q_1(s', a))) - Q_1(s, a)]$$

else:

$$Q_2(s, a) = Q_2(s, a) + \alpha[r + \gamma Q_1(s', argmax_a(Q_2(s', a))) - Q_2(s, a)]$$

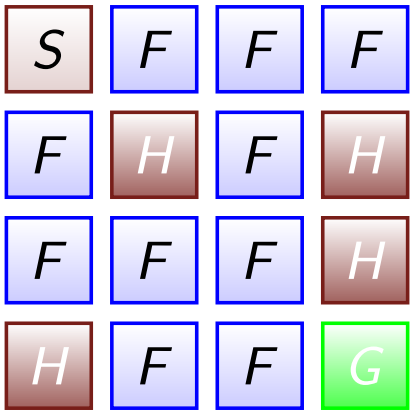
$s \leftarrow s'$
Until s is not terminal

Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2018.

Uczenie aktywne

Model środowiska

Frozen Lake:



Oznaczenia:

- S - stan początkowy,
- F - zamrożone pole,
- H - dziura (stan końcowy),
- G - cel (stan końcowy).

Nagrody:

- 1 - po dotarciu do pola G,
- 0 - w pozostałych przypadkach.

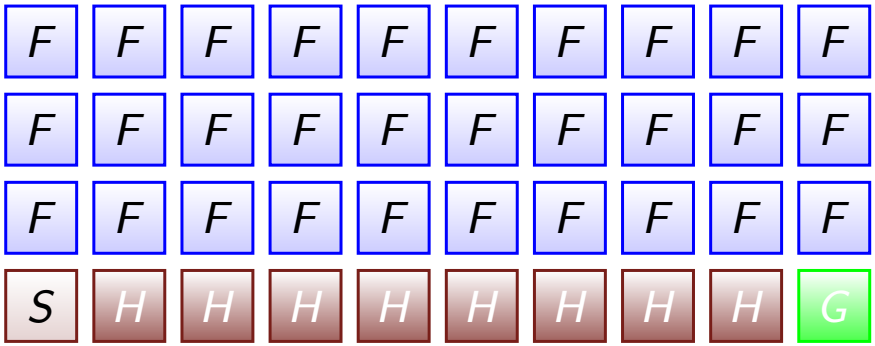
Akcje:

- lewo,
- prawo,
- góra,
- dół.

Uczenie aktywne

Model środowiska

Cliff World:



Akcje:

- lewo,
- prawo,
- góra,
- dół.

Oznaczenia:

- S - stan początkowy,
- F - wolne pole,
- H - dziura (stan końcowy),
- G - cel (stan końcowy).

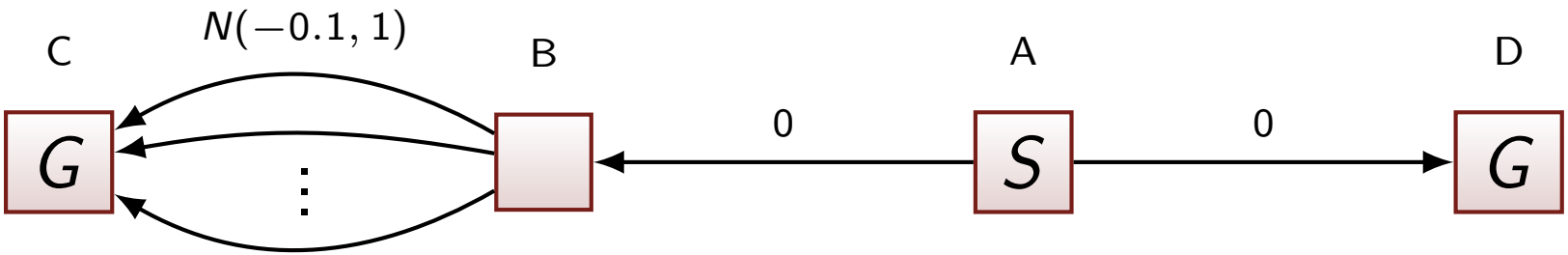
Nagrody:

- 1 - po dotarciu do pola G,
- -100 - po dotarciu do pola H,
- -1 - w pozostałych przypadkach.

Uczenie aktywne

Model środowiska

Double Q-Learning:



Oznaczenia:

- S - stan początkowy,
- G - cel (stan końcowy).

Akcje:

- lewo,
- prawo.

Uczenie aktywne

Materiały uzupełniające

- Książka *Reinforcement Learning: An Introduction*, Richard S. Sutton and Andrew G. Barto, wydanie drugie, 2018.
  - Rozdziały 5.1, 5.2, 5.3 i 5.4 - *Monte Carlo Methods*.
  - Rozdziały 6.1, 6.2, 6.3 i 6.5 - *TD Learning and Q-Learning*.
  - Rozdziały 6.4 i 6.6 - *SARSA* i *Expected SARSA*.
  - Rozdziały 6.7 - *Double Q-Learning*.
- Książka *Reinforcement Learning: An Introduction*, Richard S. Sutton and Andrew G. Barto, 2015.
  - Rozdziały 7.1 - 7.5 - *Eligibility traces* i *SARSA(λ)*.
- Video *Artificial Intelligence Course by Pieter Abbeel - Lecture 10: Reinforcement Learning – od 0:38:00*.
- Video *RL Course by David Silver - Lecture 5: Model Free Control – od 1:00:00*.

Exploration vs. exploitation

Aproksymacja funkcji wartości stanu

Aproksymacja funkcji wartości stanu

Podsumowanie dotychczasowo zdobytej wiedzy:

- opracowanie idealnej strategii na podstawie znajomości modelu środowiska (uczenie pasywne),
- opracowanie strategii na podstawie doświadczenia (uczenie aktywne),
- do tej pory wartości opisujące stan lub parę stan-akcja przechowywane były w tablicy, co może okazać się problematyczne:
  - w przypadku próby rozwiązania rzeczywistych problemów liczba stanów lub akcji może być zbyt duża, do przechowywania ich wartości w tablicy,
  - Backgammon - 10^{20} stanów,
  - Szachy - 10^{40} stanów,
  - Go - 10^{70} stanów.

Aproksymacja funkcji wartości stanu

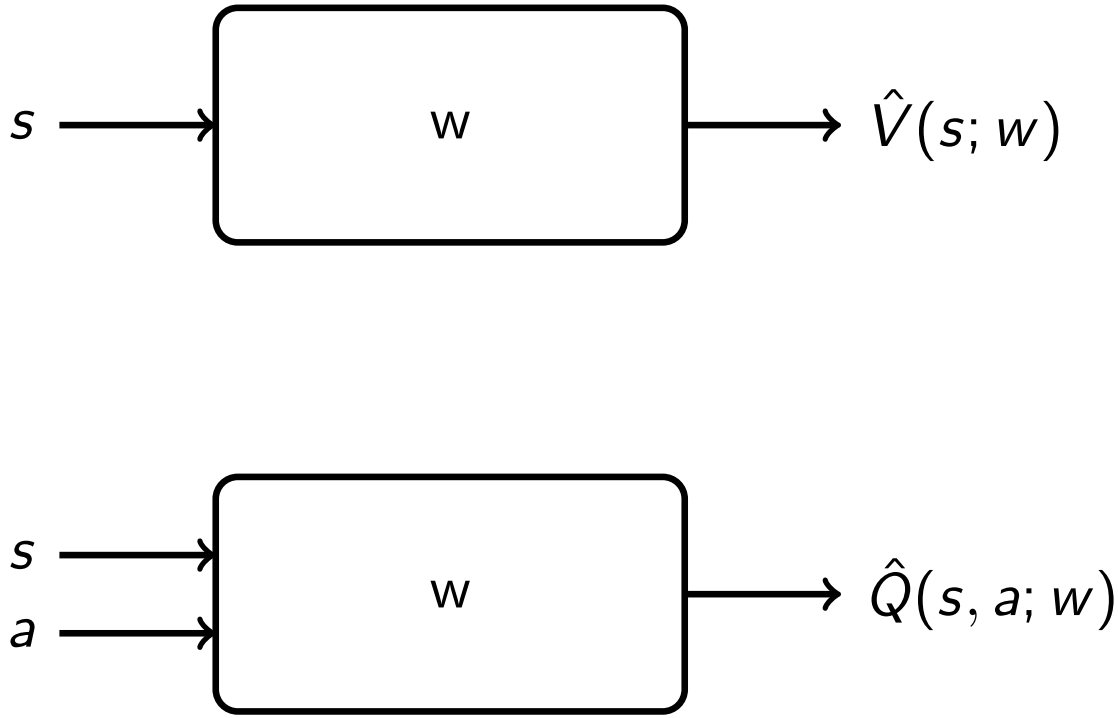
Podsumowanie dotychczasowo zdobytej wiedzy:

- opracowanie idealnej strategii na podstawie znajomości modelu środowiska (uczenie pasywne),
- opracowanie strategii na podstawie doświadczenia (uczenie aktywne),
- do tej pory wartości opisujące stan lub parę stan-akcja przechowywane były w tablicy, co może okazać się problematyczne:
  - w przypadku próby rozwiązania rzeczywistych problemów liczba stanów lub akcji może być zbyt duża, do przechowywania ich wartości w tablicy,
  - Backgammon - 10^{20} stanów,
  - Szachy - 10^{40} stanów,
  - Go - 10^{70} stanów.

Reprezentacja tabelaryczna jest niewystarczająca!

Aproksymacja funkcji wartości stanu

Reprezentacja stanu (stanu-akcji) za pomocą sparametryzowanej funkcji:



Aproksymacja funkcji wartości stanu

Wiele możliwości aproksymacji funkcji wartości stanu:

- Liniowa kombinacja cech.
- Sieci neuronowe.
- Drzewa decyzyjne.

Funkcja powinna być różniczkowalna.
Dwie najpopularniejsze klasy różniczkowalnych funkcji aproksymacyjnych:

- Liniowa kombinacja cech.
- Sieci neuronowe.

Aproksymacja funkcji wartości stanu

Liniowa kombinacja cech

- Określenie optymalnej strategii poprzez wyznaczenie wartości funkcji V(s) lub Q(s, a).
- Przechowywanie wartości funkcji w tablicy.
- Aktualizacja po każdym epizodzie (metody Monte Carlo) lub po każdym kroku (metody różnic tymczasowych).

Aproksymacja funkcji wartości stanu

Liniowa kombinacja cech

- Określenie optymalnej strategii poprzez wyznaczenie wartości funkcji V(s) lub Q(s, a).
- Przechowywanie wartości funkcji w tablicy.
- Aktualizacja po każdym epizodzie (metody Monte Carlo) lub po każdym kroku (metody różnic tymczasowych).

W przypadku funkcji aproksymujących po każdym kroku następuje zmiana parametrów tych funkcji (dopasowanie).

Aproksymacja funkcji wartości stanu

Liniowa kombinacja cech

Funkcja $f(s, a)$ zwraca wektor cech dla stanu s i akcji a. Wartość dla pary stan-akcja będzie obliczana zgodnie ze wzorem:

$$Q(s, a) = \sum_{i=1}^{n} f_i(s, a) w_i \tag{7}$$

Błąd tymczasowy:

$$\delta = (r + \gamma \max_{a'} Q(s', a')) - Q(s, a) \tag{8}$$

Aproksymacja funkcji wartości stanu

Liniowa kombinacja cech

Aktualizacja wartości wag:

$$w_i = w_i + \alpha\delta f_i(s, a) \tag{9}$$

Minimalizacja błędu:

$$J(w) = ||(r + \gamma \max_{a'} Q(s', a')) - Q(s, a)||^2 \tag{10}$$

Aproksymacja funkcji wartości stanu

Pacman

Cechami są funkcje przekształcające stan na liczbę rzeczywistą (najczęściej z zakresu $< 0, 1 >$) w taki sposób, żeby uchwycić najważniejsze właściwości stanu.

Przykładowe cechy w grze Pacman:

- odległość od najbliższego duszka,
- odległość od najbliższego jedzenia ...

Uczenie aktywne

Materiały uzupełniające

- Książka *Reinforcement Learning: An Introduction*, Richard S. Sutton and Andrew G. Barto, wydanie drugie, 2018.
  - Rozdziały 9.4 - *Linear Methods*.
- Video *Artificial Intelligence Course by Pieter Abbeel - Lecture 11: Reinforcement Learning II – od 0:31:00*.
- Video *RL Course by David Silver - Lecture 6: Value Function Approximation*.

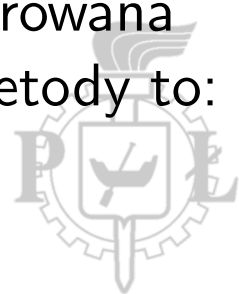
Głębokie uczenie ze wzmocnieniem

Głębokie uczenie ze wzmocnieniem

Keras

Przydatne funkcje:

- `Sequential` - funkcja tworząca model sieci w postaci liniowej serii warstw.
- `add` - funkcja dodaje warstwę do modelu.
- `Dense(units, [input_dim, activation])` - funkcja tworzy warstwę składającą się z *units* neuronów, dodatkowo można podać m.in. liczbę wejść (*inputs_dim*) i funkcję aktywacji (*activation*). Sugerowane funkcje aktywacji to *relu* i *linear*.
- `compile(loss, optimizer)` - funkcja konfiguruje model do treningu, jako parametry przyjmuje funkcję kosztu (*loss*, sugerowana *mse*) oraz metodę aktualizacji wag (*optimizer*, sugerowane metody to: *sgd* oraz *Adam*).



Głębokie uczenie ze wzmocnieniem

Keras

Przydatne funkcje cd:

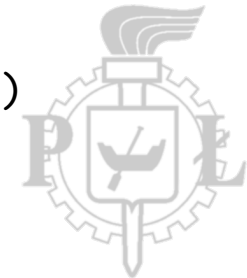
- `predict(x)` - funkcja zwraca odpowiedź sieci dla danych wejściowych (x).
- `fit(x, y, epochs, verbose)` - trenuje model przez podaną liczbę epok:
  - x - dane wejściowe,
  - y - dane oczekiwane,
  - *epochs* - liczba epok trenowania modelu,
  - *verbose* - poziom szczegółowości wyświetlanych informacji - sugerowana wartość 0.

Głębokie uczenie ze wzmocnieniem

Keras - przykład

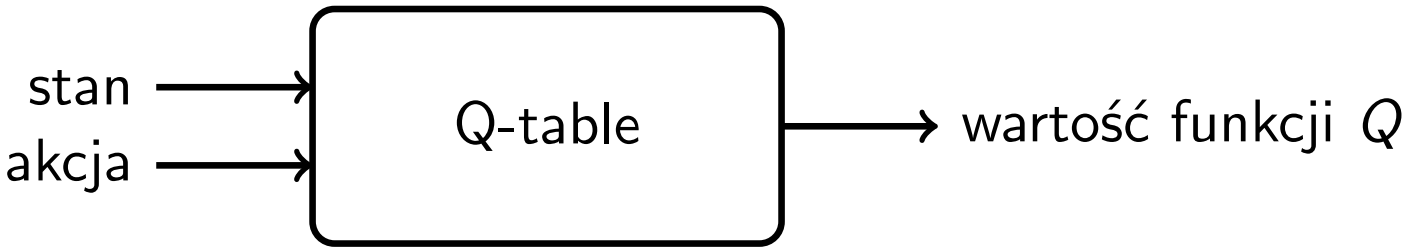
```
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import Adam, SGD

model = Sequential()
model.add(Dense(32, input_dim=16, activation = 'relu'))
#now the model will take as input arrays of shape (*,
    16)
#and output arrays of shape (*, 32) and uses ReLU
    activation function
#after the first layer, you don't need to specify
#the size of the input anymore:
model.add(Dense(32))
model.compile(loss='mse', optimizer=SGD(lr=0.001))
```

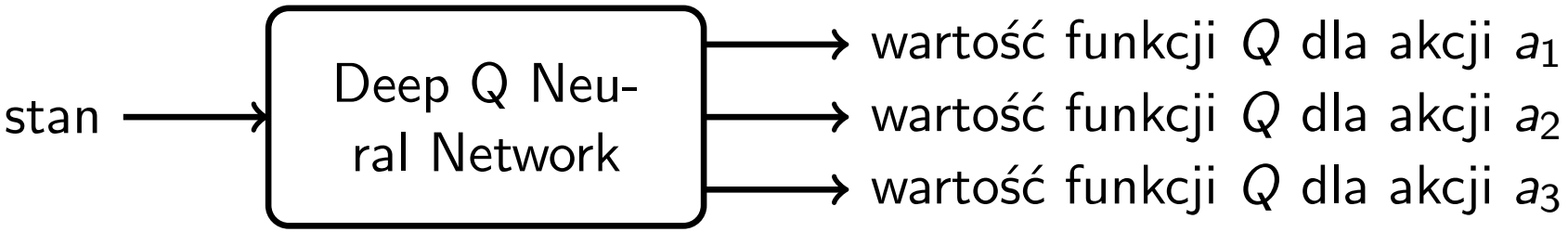


Głębokie uczenie ze wzmocnieniem

Deep Q Neural Network



Rysunek 5: Q-Learning



Rysunek 6: Deep Q Neural Network

Głębokie uczenie ze wzmocnieniem

Deep Q Neural Network

Sposób działania:

- na wejście sieci podawany jest aktualny stan,
- następna akcja jest wybierana na podstawie odpowiedzi sieci (akcja o największej wartości),
- funkcją kosztu jest błąd średniokwadratowy z wartości przewidzianej przez sieć, a tej docelowej (wyznaczonej za pomocą równania Bellmana) - $R + \gamma max_a(Q(s', a))$,
- jako wartość oczekiwaną funkcji *predict* podajemy wektor wartości Q zwrócony przez sieć dla danego stanu, z zaktualizowaną wartością dla wybranej akcji.

Głębokie uczenie ze wzmocnieniem

Deep Q Neural Network

Problemy z podejściem DQN:

- sekwencyjnie skorelowane dane - może wpływać na zbieżność i wydajność sieci,
- niestabilność dystrybucji danych z powodu zmian strategii - strategia może nie być zbieżna lub oscylować.

Głębokie uczenie ze wzmocnieniem

Deep Q Neural Network - Experience Replay

Rozwiązanie problemu sekwencyjnie skorelowanych danych - zapamiętanie całej serii wykonanych działań, to znaczy stanu (s), akcji podjętej w tym stanie (a), otrzymanej nagrody (r) oraz kolejnego stanu (s'). Uczenie sieci na podstawie losowo dobranych próbek z zebranych danych.

Głębokie uczenie ze wzmocnieniem

Deep Q Neural Network - Experience Replay

Sposób działania:

1. zebranie zbioru danych w postaci krotek - (s, a, r, s') na podstawie akcji podjętych przez agenta (sieć), korzystając z algorytmu ϵ-zachłannego,
2. wybór losowych krotek z zebranego zbioru,
3. trenowanie modelu sieci na podstawie wybranych danych,
4. podejmowanie nowych akcji na podstawie zaktualizowanego modelu i algorytmu ϵ-zachłannego,
5. powrót do punktu 2.

Głębokie uczenie ze wzmocnieniem

Double Deep Q Neural Network

Rozwiązanie problemu niestabilności dystrybucji danych z powodu zmian strategii, ocena wybranej akcji odbywa się za pomocą innej sieci, niż sam wybór akcji.

Głębokie uczenie ze wzmocnieniem

Double Deep Q Neural Network

Korzystanie z dwóch sieci:

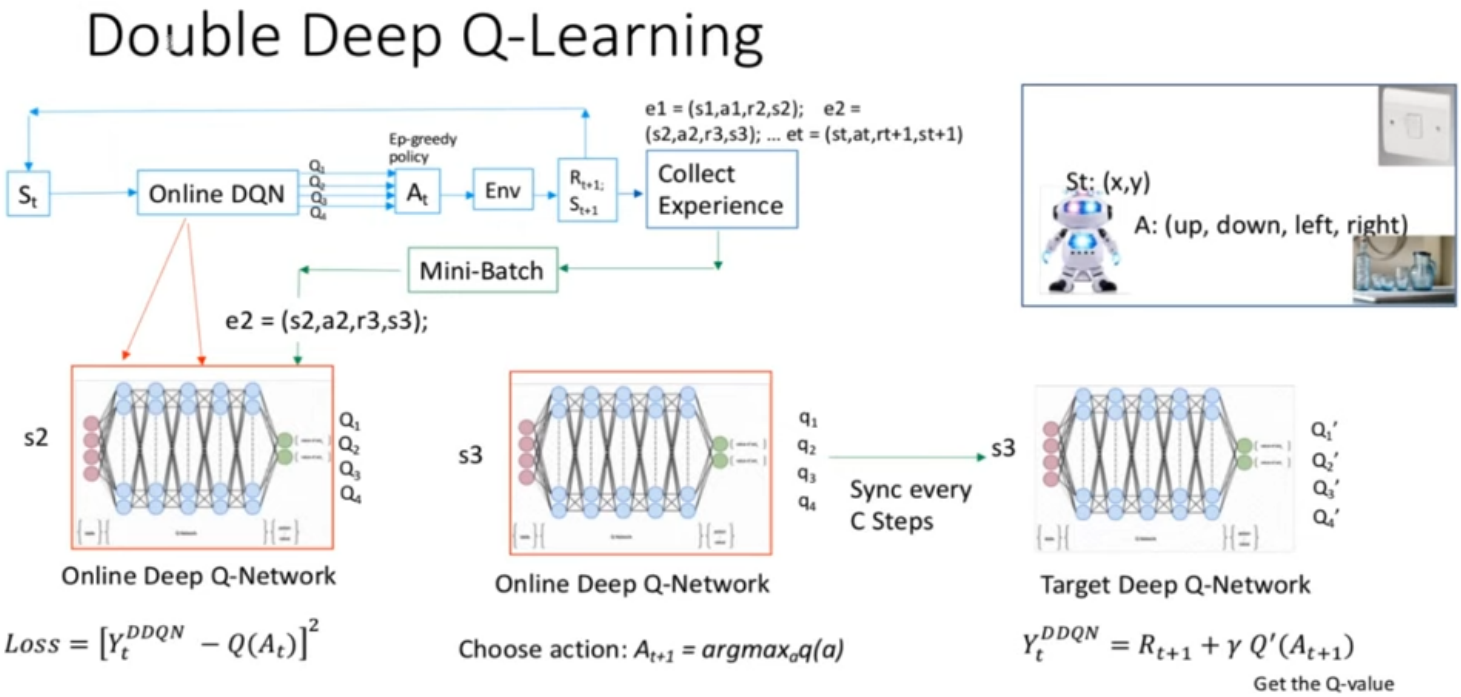
- sieć Q' - wykorzystywana do wyboru akcji podejmowanych przez agenta,
- sieć Q - wykorzystywana do oszacowania wybranej akcji.

Funkcją kosztu jest błąd średniokwadratowy z wartości przewidzianej przez sieć Q', a tej oszacowanej prze sieć Q (wyznaczonej za pomocą równania Bellmana):

$$Q^*(s_t, a_t) \approx r_t + \gamma argmax_{a'} Q'(s_{t+1}, a')$$

Głębokie uczenie ze wzmocnieniem

Double Deep Q Neural Network



Rysunek 7: Double Deep Q-Learning

[DDQN] Deep Reinforcement Learning with Double Q-learning — TDLS Foundational)

Uczenie aktywne

Double Deep Q Neural Network

Double Q-Learning

Initialize primary network Q_θ, target network Q'_θ, replay buffer D, $\tau << 1$.
For each iteration do:
For each environment step do:
Observe state s and select action a using network Q_θ:
Take action a, observe r, s'
Store (s, a, r, s') in replay buffer D
For each update step do:
Sample $e = (s, a, r, s)$ from D
Compute target Q value:

$$Q^*(s,a) \approx r + \gamma argmax_{a'} Q'_\theta(s', a')$$

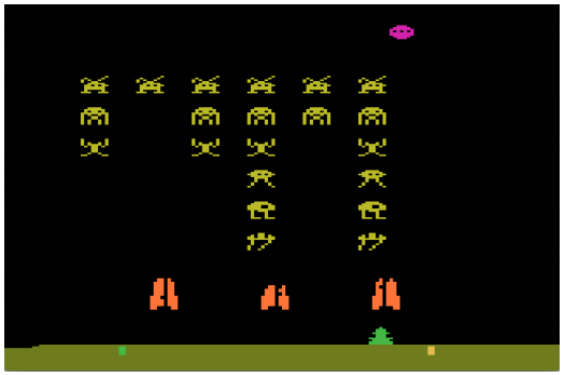
Perform gradient descent step on $(Q^*(s,a) - Q_\theta(s,a))^2$
Update target network parameters:

$$\theta' \leftarrow \tau * \theta + (1 - \tau) * \theta'$$

Mnih, V., Kavukcuoglu, K., Silver, D. et al. Human-level control through deep reinforcement learning. Nature 518, 529–533 (2015).

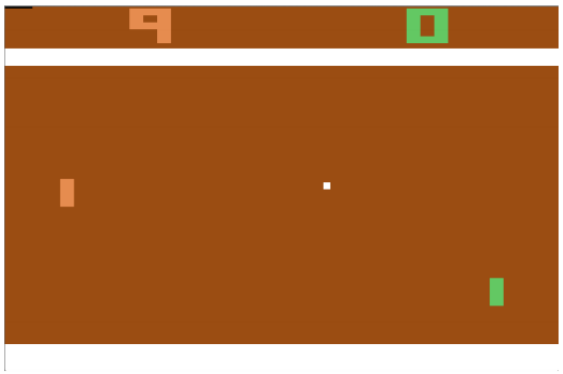
Głębokie uczenie ze wzmocnieniem

Double Deep Q Neural Network



(a) Space Invaders

(b) Beam Rider



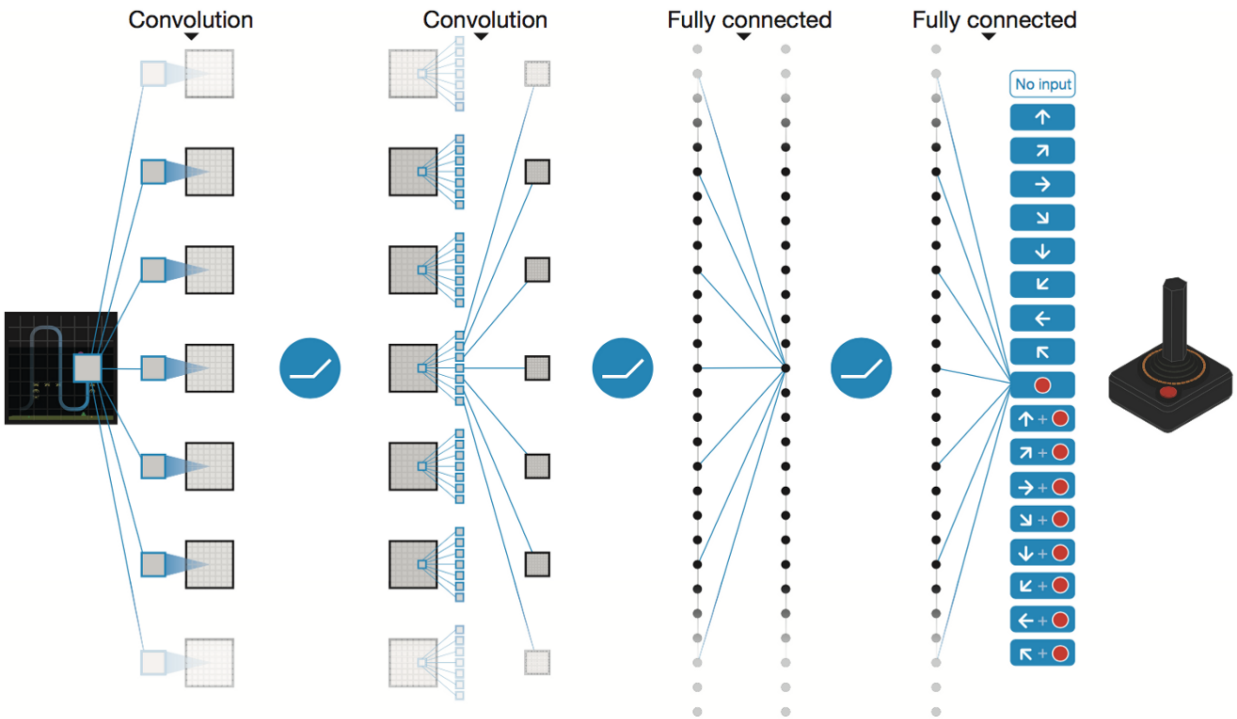
(c) Pong

(d) Breakout

Rysunek 8: Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." (2013).

Głębokie uczenie ze wzmocnieniem

Double Deep Q Neural Network

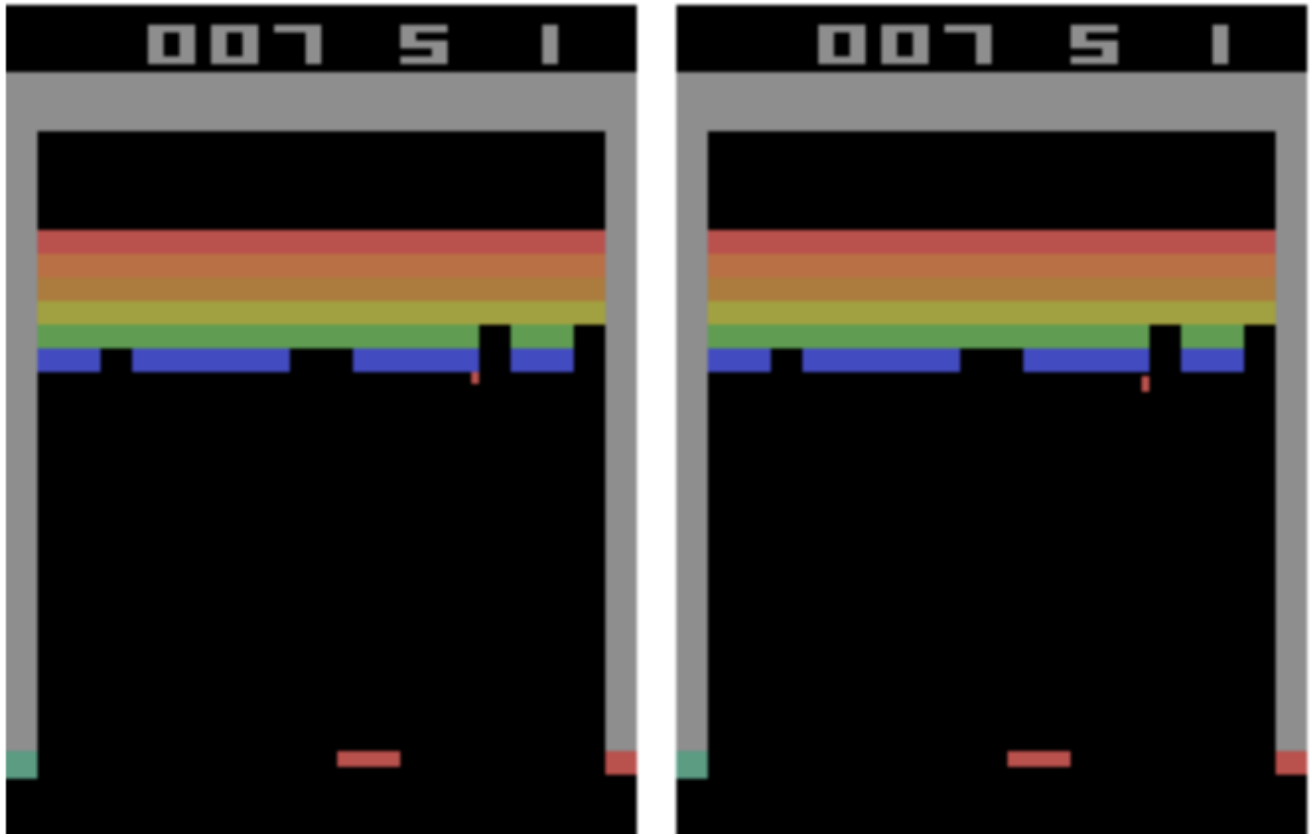


Rysunek 9: Network Architecture

Mnih, V., Kavukcuoglu, K., Silver, D. et al. Human-level control through deep reinforcement learning. Nature 518, 529–533 (2015).

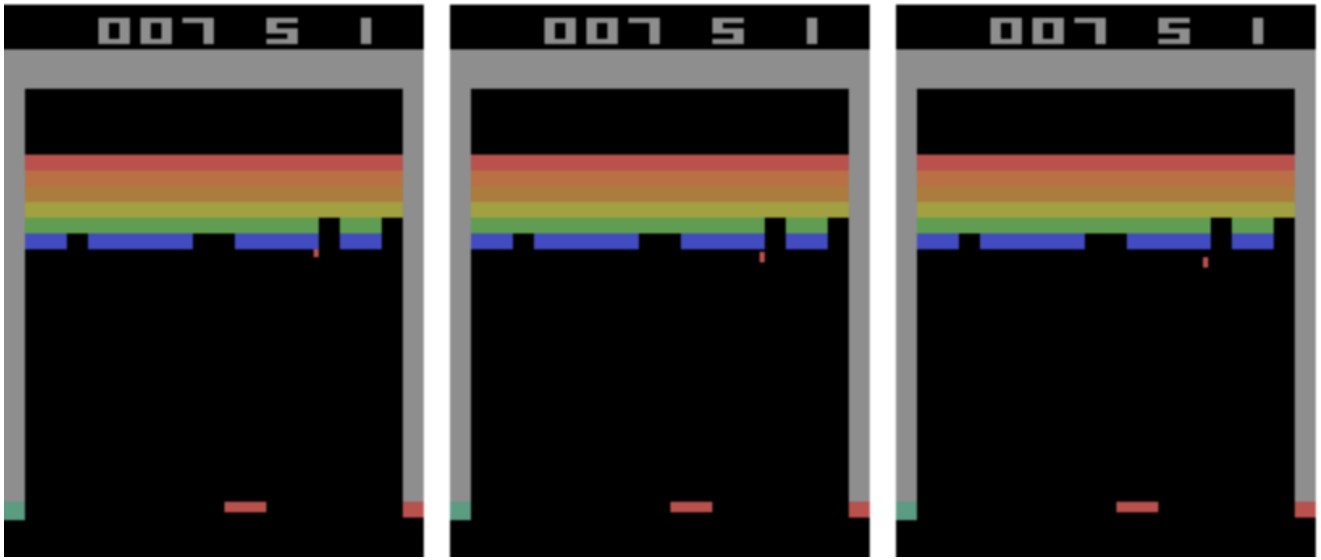
Głębokie uczenie ze wzmocnieniem

Double Deep Q Neural Network



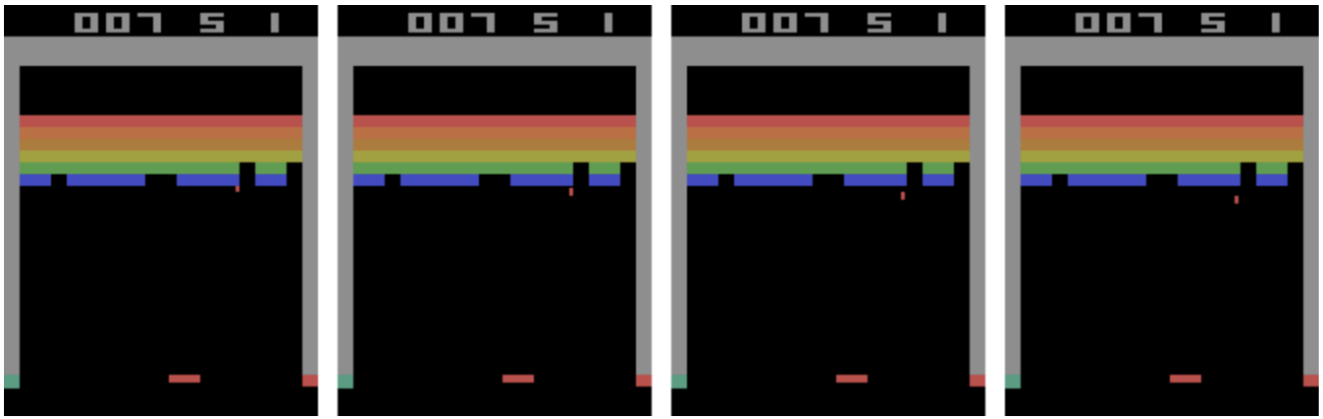
Głębokie uczenie ze wzmocnieniem

Double Deep Q Neural Network



Głębokie uczenie ze wzmocnieniem

Double Deep Q Neural Network



Głębokie uczenie ze wzmocnieniem

Double Deep Q Neural Network

Przetwarzania wstępne obrazów:

- wycięcie istotnego fragmentu obrazu,
- zamiana na odcienie szarości,
- zmniejszenie obrazu.

Głębokie uczenie ze wzmocnieniem

Double Deep Q Neural Network

Hyperparameter	Value	Description
minibatch size	32	Number of training cases over which each stochastic gradient descent (SGD) update is computed.
replay memory size	1000000	SGD updates are sampled from this number of most recent frames.
agent history length	4	The number of most recent frames experienced by the agent that are given as input to the Q network.
discount factor	0.99	Discount factor gamma used in the Q-learning update.
action repeat	4	Repeat each action selected by the agent this many times. Using a value of 4 results in the agent seeing only every 4th input frame.
update frequency	4	The number of actions selected by the agent between successive SGD updates. Using a value of 4 results in the agent selecting 4 actions between each pair of successive updates.
learning rate	0.00025	The learning rate used by RMSProp.
gradient momentum	0.95	Gradient momentum used by RMSProp.
squared gradient momentum	0.95	Squared gradient (denominator) momentum used by RMSProp.
min squared gradient	0.01	Constant added to the squared gradient in the denominator of the RMSProp update.
initial exploration	1	Initial value of ε in ε-greedy exploration.
final exploration	0.1	Final value of ε in ε-greedy exploration.
final exploration frame	1000000	The number of frames over which the initial value of ε is linearly annealed to its final value.
replay start size	50000	A uniform random policy is run for this number of frames before learning starts and the resulting experience is used to populate the replay memory.
no-op max	30	Maximum number of "do nothing" actions to be performed by the agent at the start of an episode.

Rysunek 10: Przykładowe parametry sieci

Mnih, V., Kavukcuoglu, K., Silver, D. et al. Human-level control through deep reinforcement learning. Nature 518, 529–533 (2015).

Głębokie uczenie ze wzmocnieniem

Double Deep Q Neural Network

Dla chętnych:
Yandex School of Data Analysis - Practical RL

Głębokie uczenie ze wzmocnieniem

Krótkie podsumowanie

- Uczenie pasywne (ang. *model-based learning*):
  - Iteracyjne doskonalenie strategii (ang. *Policy Iteration*).
  - Iteracyjne obliczanie funkcji wartości (ang. *Value Iteration*).

Głębokie uczenie ze wzmocnieniem

Krótkie podsumowanie

- Uczenie pasywne (ang. *model-based learning*):
  - Iteracyjne doskonalenie strategii (ang. *Policy Iteration*).
  - Iteracyjne obliczanie funkcji wartości (ang. *Value Iteration*).
- Uczenie aktywne (ang. *model-free learning*):
  - Algorytmy bazujące na wyznaczaniu funkcji wartości:
    - Q-Learning (*off-policy*).
    - SARSA (*on-policy*).
    - Metody aproksymacyjne (Liniowa kombinacja cech, *Deep Q Networks*).

Głębokie uczenie ze wzmocnieniem

Krótkie podsumowanie

- Uczenie pasywne (ang. *model-based learning*):
  - Iteracyjne doskonalenie strategii (ang. *Policy Iteration*).
  - Iteracyjne obliczanie funkcji wartości (ang. *Value Iteration*).
- Uczenie aktywne (ang. *model-free learning*):
  - Algorytmy bazujące na wyznaczaniu funkcji wartości:
    - Q-Learning (*off-policy*).
    - SARSA (*on-policy*).
    - Metody aproksymacyjne (Liniowa kombinacja cech, *Deep Q Networks*).
  - Algorytmy bazujące na wyznaczaniu strategii (ang. *REINFORCE*).

Głębokie uczenie ze wzmocnieniem

Policy Gradients

Dotychczasowe podejście:

- Aproksymacja funkcji wartości:

$$v_*(s) \approx v_\pi(s)$$

$$q_*(s,a) \approx q_\pi(s,a)$$

- Wyznaczanie strategii na podstawie tych funkcji (algorytm zachłanny, ϵ-zachłanny).

Nowe podejście:

- Wyznaczanie strategii bezpośrednio:

$$\pi_\theta(s) = p(a|s,\theta)$$

Głębokie uczenie ze wzmocnieniem

Policy Gradients

Zalety optymalizacji strategii:

- Strategia wyznaczana przez metody Q-Learning jest zawsze deterministyczna. Metody te nie mogą opracować strategii stochastycznej, co może być przydatne w niektórych środowiskach. W takim przypadku wykorzystywane są algorytmy, np. ϵ-zachłanne, ale to podejście nie zawsze jest wystarczające i optymalne.
- Metody należące do grupy optymalizujących strategie mogą być zastosowane w przypadku środowisk, gdzie akcje mają charakter ciągły.
- W sposób ciągły poprawiana jest strategia, a przypadku metod Q-Learning aproksymowana jest funkcja wartości, a dopiero na jej podstawie wyznaczana jest strategia.

Głębokie uczenie ze wzmocnieniem

Policy Gradients

- Algorytmy optymalizacji strategii należą do dziedziny algorytmów optymalizacyjnych.
- Celem jest znalezienie takich wartości parametrów θ, które będą dawały najwyższy wynik funkcji $J(\theta)$.
- Metody nie wykorzystujące gradientów (np. algorytmy genetyczne).
- Metody wykorzystujące gradient (np. metoda gradientu prostego).

Głębokie uczenie ze wzmocnieniem

Policy Gradients

$$\theta_{t+1} = \theta_t + \alpha \nabla \pi_{\theta_t}(s, a^*)$$

Głębokie uczenie ze wzmocnieniem

Policy Gradients

$$\theta_{t+1} = \theta_t + \alpha \nabla \pi_{\theta_t}(s, a^*)$$

$$\theta_{t+1} = \theta_t + \alpha \hat{Q}(s, a) \nabla \pi_{\theta_t}(s, a)$$

Głębokie uczenie ze wzmocnieniem

Policy Gradients

$$\theta_{t+1} = \theta_t + \alpha \nabla \pi_{\theta_t}(s, a^*)$$

$$\theta_{t+1} = \theta_t + \alpha \hat{Q}(s, a) \nabla \pi_{\theta_t}(s, a)$$

$$\theta_{t+1} = \theta_t + \alpha \frac{\hat{Q}(s, a) \nabla \pi_{\theta_t}(s, a)}{\pi_\theta(s, a)}$$

Głębokie uczenie ze wzmocnieniem

Policy Gradients

$$\theta_{t+1} = \theta_t + \alpha \nabla \pi_{\theta_t}(s, a^*)$$

$$\theta_{t+1} = \theta_t + \alpha \hat{Q}(s, a) \nabla \pi_{\theta_t}(s, a)$$

$$\theta_{t+1} = \theta_t + \alpha \frac{\hat{Q}(s, a) \nabla \pi_{\theta_t}(s, a)}{\pi_\theta(s, a)}$$

$$\theta_{t+1} = \theta_t + \alpha \hat{Q}(s, a) \nabla_\theta log \pi_{\theta_t}(s, a)$$

Głębokie uczenie ze wzmocnieniem

Monte-Carlo Policy Gradient (REINFORCE)

Aktualizacja parametrów metodą gradientu prostego (ang. *stochastic gradient ascent*).

Wykorzystanie nagrody skumulowanej jako wartości funkcji $Q(\hat{s}, a)$.

$$\Delta\theta_t = \alpha G_t \nabla_\theta log \pi_{\theta_t}(s, a)$$

Głębokie uczenie ze wzmocnieniem

Monte-Carlo Policy Gradient (REINFORCE)

REINFORCE: Monte-Carlo Policy-Gradient Control (episodic) for π_*

Input: a differentiable policy parameterization $\pi(a|s,\theta)$.
Algorithm parameter: step size $\alpha > 0$.
Initialize policy parameter $\theta \in \mathbb{R}$.
For each episode:

Generate an episode $s_0, a_0, r_1, ..., s_{t-1}, a_{t-1}, r_t$, following $\pi(.|.,\theta)$.
Loop for each step of the episode $t = 0, 1, ..., T-1$:

$$G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} r_k$$

$$\theta \leftarrow \theta + \alpha G_t \nabla_\theta log \pi_\theta(a_t, s_t|\theta)$$

Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2018.

Głębokie uczenie ze wzmocnieniem

REINFORCE with Baseline

$$\theta_{t+1} = \theta_t + \alpha(G_t - b(s_t))\nabla_\theta log\pi_{\theta_t}(s_t, a_t) \tag{11}$$

Metody wyznaczania wartości odniesienia:

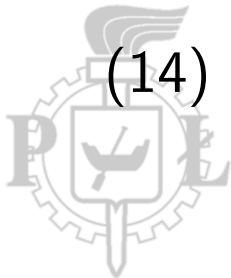
- *whitening*:

$$G_t^* = \frac{G_t - \mu_G}{\sigma_G} \tag{12}$$

- *learned baseline*:

$$\theta_{t+1} = \theta_t + \alpha(G_t - \hat{v}(s_t, w))\nabla_\theta log\pi_{\theta_t}(s_t, a_t) \tag{13}$$

$$w_{t+1} = w_t + \beta(G_t - \hat{v}(s_t, w))\nabla_v(s_t, w) \tag{14}$$



Głębokie uczenie ze wzmocnieniem

REINFORCE with Baseline

REINFORCE with Baseline (episodic), for estimating $\pi_\theta \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a|s,\theta)$.
Input: a differentiable state-value function parameterization $\upsilon(s,w)$.
Algorithm parameter: step sizes $\alpha > 0$ and $\beta > 0$.
Initialize policy parameter $\theta \in \mathbb{R}$ and state-value weights $w \in \mathbb{R}$.
For each episode:

Generate an episode $s_0, a_0, r_1, ..., s_{t-1}, a_{t-1}, r_t$, following $\pi(.|.,\theta)$.
Loop for each step of the episode $t = 0, 1, ..., T-1$:

$$G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} r_k$$
$$\delta \leftarrow G - \hat{v}(s_t, w)$$
$$w \leftarrow w + \beta\delta\nabla_w \upsilon(s_t, w)$$
$$\theta \leftarrow \theta + \alpha\delta\nabla_\theta log\pi_\theta(a_t, s_t|\theta)$$

Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2018.

Głębokie uczenie ze wzmocnieniem

Actor-Critic

- uczy się strategii - $\pi_\theta(s, a)$,
- uczy się funkcji oceny stanu - $V_\theta(s)$,
- wykorzystać $V_\theta(s)$, żeby szybciej nauczyć się strategii $\pi_\theta(s, a)$.

Głębokie uczenie ze wzmocnieniem

Actor-Critic

$$\theta_{t+1} = \theta_t + \alpha(G_{t:t+1} - \hat{v}(s_t, w))\nabla_\theta log\pi_\theta(a_t, s_t|\theta) \quad (15)$$

$$= \theta_t + \alpha(r_{t+1} + \gamma\hat{v}(s_{t+1}, w) - \hat{v}(s_t, w))\nabla_\theta log\pi_\theta(a_t, s_t|\theta) \quad (16)$$

$$= \theta_t + \alpha\delta_t\nabla_\theta log\pi_\theta(a_t, s_t|\theta) \quad (17)$$

Głębokie uczenie ze wzmocnieniem

Actor-Critic

One-step Actor-Critic, for estimating $\pi_\theta \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a|s,\theta)$.
Input: a differentiable state-value function parameterization $\upsilon(s,w)$.
Algorithm parameter: step sizes $\alpha > 0$ and $\beta > 0$.
Initialize policy parameter $\theta \in \mathbb{R}$ and state-value weights $w \in \mathbb{R}$.
For each episode:
    Initialize s (first step of episode).
    Loop while s is not terminal (for each time step):
        Choose action a according to the policy $\pi_\theta(.,s,\theta)$,
        Take action a, observe s', r,
        $\delta \leftarrow r + \gamma\hat{\upsilon}(s',w) - \hat{\upsilon}(s,w)$.
        $w \leftarrow w + \beta\delta\nabla_w\upsilon(s',w)$.
        $\theta \leftarrow \theta + \alpha\delta\nabla_\theta log\pi_\theta(a,s|\theta)$.
        $s \leftarrow s'$.

Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2018.

Głębokie uczenie ze wzmocnieniem

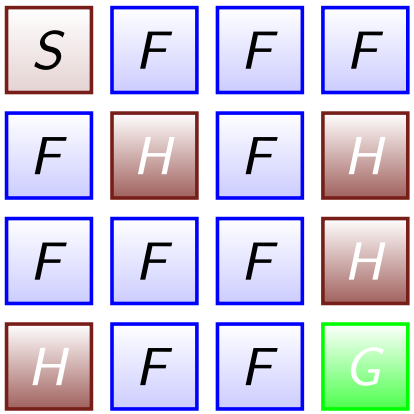
Materiały uzupełniające

- Książka *Reinforcement Learning: An Introduction*, Richard S. Sutton and Andrew G. Barto, wydanie drugie, 2018.
  - Rozdziały 13.1 - 13.4 - *Policy Gradient Methods*.
  - Rozdziały 13.5 - *Actor–Critic Methods*.
- Video *Hado Van Hasselt, Advanced Deep Learning & Reinforcement Learning Lectures. Reinforcement Learning 6: Policy Gradients and Actor Critics*.
- Video *RL Course by David Silver - Lecture 7: Policy Gradient Methods*.

Głębokie uczenie ze wzmocnieniem

Model środowiska

Frozen Lake:

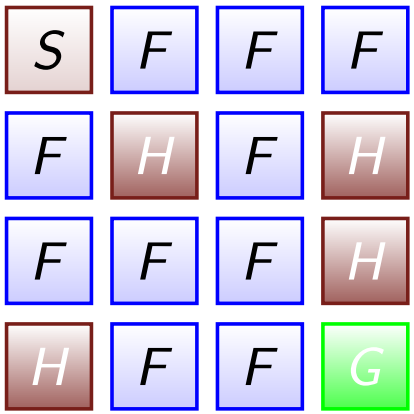


Stan jest reprezentowany za pomocą pojedynczej wartości z przedziału $< 0, 15 >$ oznaczającej pozycję agenta na planszy.
W przypadku sieci wejściem będzie wektor 16 elementowy, gdzie wartością 1 będzie zaznaczony aktualny stan - pozostałe wartości będą równe 0.

Głębokie uczenie ze wzmocnieniem

Model środowiska

Frozen Lake Extended:



Stan jest reprezentowany za pomocą trzech tablic - w pierwszej zaznaczone cyfrą 1 są dziury, w drugiej pozycja gracza, a w trzeciej cel.
W przypadku sieci wejściem będzie wektor złożony z trzech tablic.

Głębokie uczenie ze wzmocnieniem

Model środowiska

OpenAI Gym - CartPole:

Akcje:

- ruch wózka w lewo,
- ruch wózka w prawo.

Stan:

- pozycja wózka,
- prędkość wózka,
- kąt słupa,
- prędkość końcówki słupa.

Nagrody:

- 1 - za każdy krok.

Uczenie ze wzmocnieniem

Kursy online

- Reinforcement Learning:
  - *Practical Reinforcement Learning* - Yandex School of Data Analysis,
  - *Deep Reinforcement Learning* - CS 285 at UC Berkeley.
- Deep Learning:
  - *MIT 6.S191 Introduction to Deep Learning* - Massachusetts Institute of Technology,
  - *Natural Language Processing with Deep Learning* - CS224n at Stanford.

Uczenie ze wzmocnieniem

Wykłady

- *Hado Van Hasselt, Advanced Deep Learning & Reinforcement Learning*.
- *RL Course by David Silver*.
- *Deep Learning State of the Art* - MIT Deep Learning Series.