

# Uczenie ze wzmocnieniem



Piotr Duch

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

Zima 2022

# Plan wykładu

- 1 Wprowadzenie
- 2 Algorytmy w systemach wieloagentowych
- 3 Podstawowe pojęcia
- 4 Uczenie pasywne
- 5 Uczenie aktywne
- 6 Aproksymacja funkcji wartości stanu



## Informacje ogólne:

- Materiały wykładowe oraz laboratoryjne dostępne są na stronie (pduch.iis.p.lodz.pl).
- Literatura podstawowa:
  - Richard S. Sutton, and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
  - Morales, Miguel. *Grokking deep reinforcement learning*. Simon and Schuster, 2020.
- 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](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/>



## Informacje ogólne - zaliczenie:

### ■ Projekt:

- Zadania do wykonania w Pythonie (3 notebooki pythonowe, jeden projekt),
- Projekt własnej gry w Pythonie wraz z implementacją wybranych algorytmów.
- Rozbudowa projektu Pacman (implementacja algorytmu aproksymacji funkcji wartości stanu),
- Ocena końcowa:
  - Część I - 66% - 73% ocena 3, 73% - 80% ocena 3.5, 80% - 87% ocena 4, 87% - 94% ocena 4.5, 94% i więcej - 5.
  - Część II - ocena projektu.
  - Część III - ocena na podstawie turnieju botów.
  - Ocena końcowa jest oceną ważoną z każdej części (30%, 40%, 40%).  
Konieczne jest uzyskanie pozytywnej oceny z każdej części.

### ■ Wykład - ??.

### ■ Kontakt:

- poprzez platformę MS Teams na chacie indywidualnym,
- mailowo: [pduch@iis.p.lodz.pl](mailto:pduch@iis.p.lodz.pl).



## Informacje szczegółowe - plan działania:

- Minimax, Alpha-Beta, Expectimax - *Project 2: Multi-Agent Search, Berkeley*.
- MCTS - *Monte Carlo Tree Search* - implementacja we własnym projekcie.
- Uczenie pasywne (*Policy Evaluation, Policy Improvement, Policy Iteration, Value Iteration*) - Notebook Pythonowy 1 + implementacja wybranego algorytmu we własnym projekcie (Pliki dodatkowe do notebooków).
- Uczenie aktywne (*Q-Learning, Sarsa, Expected Sarsa, Sarsa ( $\lambda$ ), Double Q-Learning*) - Notebook Pythonowy 2 i Notebook Pythonowy 3 + implementacja wybranego algorytmu we własnym projekcie.
- Aproksymacja funkcji wartości - implementacja we własnym projekcie + implementacja w pacmanie (Multiagent Pacman).



## Informacje szczegółowe - terminy:

- Minimax, Alpha-Beta, Expectimax - 3.11.2022.
- MCTS - *Monte Carlo Tree Search* - implementacja we własnym projekcie - 22.12.2022.
- Uczenie pasywne (*Policy Evaluation, Policy Improvement, Policy Iteration, Value Iteration*) - 10.11.2022 + implementacja wybranego algorytmu we własnym projekcie - 8.12.2022.
- Uczenie aktywne (*Q-Learning, Sarsa, Expected Sarsa, Sarsa ( $\lambda$ ), Double Q-Learning*) - 24.11.2022 i implementacja wybranego algorytmu we własnym projekcie - 15.12.2022.
- Aproksymacja funkcji wartości - implementacja we własnym projekcie - 12.01.2023 + implementacja w pacmanie (link zostanie dodany później) - 26.01.2023.



# Uczenie ze wzmocnieniem

## Wprowadzenie



# Uczenie ze wzmocnieniem - wprowadzenie

## Wprowadzenie





# Uczenie ze wzmocnieniem - wprowadzenie

## Wprowadzenie

Uczenie maszynowe:



# Uczenie ze wzmocnieniem - wprowadzenie

## Wprowadzenie

Uczenie maszynowe:

- Uczenie z nadzorem:
  - Klasyfikacja.
  - Regresja (predykcja).



# Uczenie ze wzmocnieniem - wprowadzenie

## Wprowadzenie

### Uczenie maszynowe:

- Uczenie z nadzorem:
  - Klasyfikacja.
  - Regresja (predykcja).
- Uczenie bez nadzoru:
  - Grupowanie (m.in. klasteryzacja, analiza skupień).
  - Redukcja wymiarów.
  - Uzupełnianie wartości.



# Uczenie ze wzmocnieniem - wprowadzenie

## Wprowadzenie

### Uczenie maszynowe:

- Uczenie z nadzorem:
  - Klasyfikacja.
  - Regresja (predykcja).
- Uczenie bez nadzoru:
  - Grupowanie (m.in. klasteryzacja, analiza skupień).
  - Redukcja wymiarów.
  - Uzupełnianie wartości.
- Uczenie ze wzmocnieniem.



# Uczenie ze wzmocnieniem - wprowadzenie

## Wprowadzenie

Co odróżnia uczenie ze wzmocnieniem od innych działów uczenia maszynowego:



# Uczenie ze wzmocnieniem - wprowadzenie

## Wprowadzenie

Co odróżnia uczenie ze wzmocnieniem od innych działów uczenia maszynowego:

- Nie potrzebna jest baza danych - agent uczy się na podstawie informacji o nagrodach otrzymywanych ze środowiska.



# Uczenie ze wzmocnieniem - wprowadzenie

## Wprowadzenie

Co odróżnia uczenie ze wzmocnieniem od innych działów uczenia maszynowego:

- Nie potrzebna jest baza danych - agent uczy się na podstawie informacji o nagrodach otrzymywanych ze środowiska.
- Nagroda może być odłożona w czasie.



# Uczenie ze wzmocnieniem - wprowadzenie

## Wprowadzenie

Co odróżnia uczenie ze wzmocnieniem od innych działów uczenia maszynowego:

- Nie potrzebna jest baza danych - agent uczy się na podstawie informacji o nagrodach otrzymywanych ze środowiska.
- Nagroda może być odłożona w czasie.
- Czas ma znaczenie.





# Uczenie ze wzmocnieniem - wprowadzenie

## Wprowadzenie

Co odróżnia uczenie ze wzmocnieniem od innych działów uczenia maszynowego:

- Nie potrzebna jest baza danych - agent uczy się na podstawie informacji o nagrodach otrzymywanych ze środowiska.
- Nagroda może być odłożona w czasie.
- Czas ma znaczenie.
- Działanie agenta ma wpływ na dane, jakie otrzymuje ze środowiska.



# Uczenie ze wzmocnieniem - wprowadzenie

## Wprowadzenie

Co to jest:

- **Agent** wchodzi w interakcję ze **środowiskiem**, w którym chce osiągnąć określony **cel**.
- **Akcje** podjęte przez agenta są oceniane przez środowisko.
- W wyniku wykonania wybranej akcji, agent otrzymuje **nagrodę** (może być pozytywna lub negatywna).



# Uczenie ze wzmocnieniem - wprowadzenie

## Metody

- Uczenie pasywne:
  - Ocena strategii (ang. *Policy Evaluation*)
  - Polepszanie strategii (ang. *Policy Improvement*)
  - Iteracyjne doskonalenie strategii (ang. *Policy Iteration*)
  - Iteracyjne obliczanie funkcji wartości (ang. *Value Iteration*)



# Uczenie ze wzmocnieniem - wprowadzenie

## Metody cd.

- Uczenie aktywne:
  - Metody różnic czasowych (ang. *Temporal Difference Learning*)
    - Monte Carlo
    - Q-Learning
    - SARSA
  - Metody aproksymacyjne
    - Aproksymacja funkcji wartości (ang. *Approximate Q-Learning*)
  - Metody wykorzystujące głębokie sieci neuronowe
    - Deep Q-Learning
    - Double Q-Learning
    - Actor-Critic
    - REINFORCE
    - Policy Gradient



# Wprowadzenie

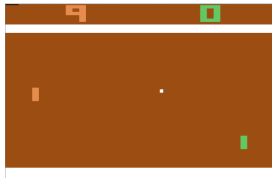
## Przykłady



(a) Space Invaders



(b) Beam Rider



(c) Pong



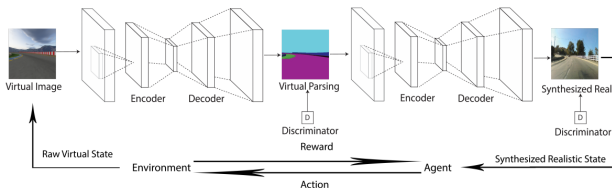
(d) Breakout

Rysunek 1: Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." (2013).



# Wprowadzenie

## Przykłady cd.



Pan, Xinlei, et al. "Virtual to real reinforcement learning for autonomous driving." arXiv preprint arXiv:1704.03952 (2017).



# Wprowadzenie

## Przykłady cd.

---

A: Where are you going? (1)

B: I'm going to the police station. (2)

A: I'll come with you. (3)

B: No, no, no, no, you're not going anywhere. (4)

A: Why? (5)

B: I need you to stay here. (6)

A: I don't know what you are talking about. (7)

A: How old are you? (1)

B: I'm 16. Why are you asking? (2)

A: I thought you were 12. (3)

B: What made you think so? (4)

A: I don't know what you are talking about. (5)

B: You don't know what you are saying. (6)

Li, Jiwei, et al. "Deep reinforcement learning for dialogue generation." (2016).



# Wprowadzenie

Przykłady cd.

Obrót akcjami:

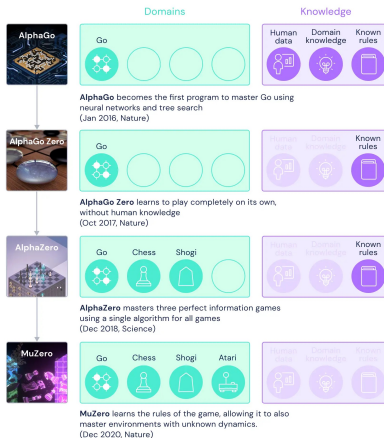
- Dempster, Michael AH, and Vasco Leemans. "An automated FX trading system using adaptive reinforcement learning." *Expert Systems with Applications* 30.3 (2006): 543-552.
- Xiong, Zhuoran, et al. "Practical deep reinforcement learning approach for stock trading." *arXiv preprint arXiv:1811.07522* (2018).
- Carapuço, João, Rui Neves, and Nuno Horta. "Reinforcement learning applied to Forex trading." *Applied Soft Computing* 73 (2018): 783-794.





# Wprowadzenie

## Przykłady cd.

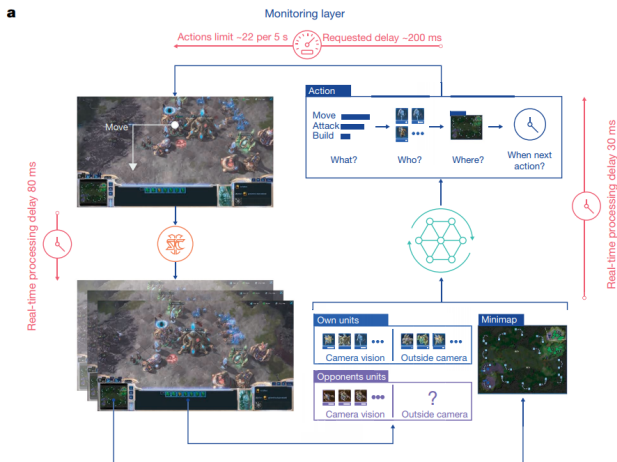


Rysunek 3: Od AlphaGo do MuZero



# Wprowadzenie

## Przykłady cd.



Rysunek 4: AlphaStar



# Algorytmy w systemach wieloagentowych (ang. *Multi-agent search algorithms*)



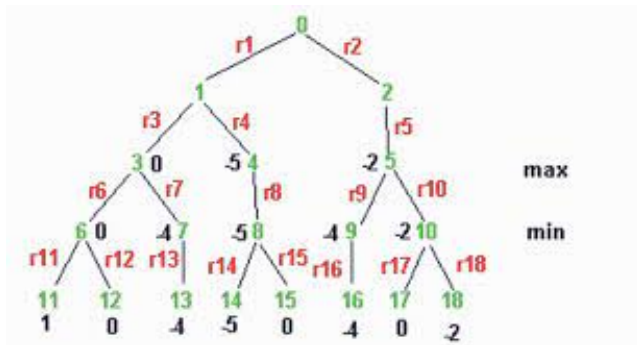
# Algorytmy

- Minimax
- Alpha-Beta
- Monte Carlo Tree Search (MCTS)



# Algorytmy

## Minimax

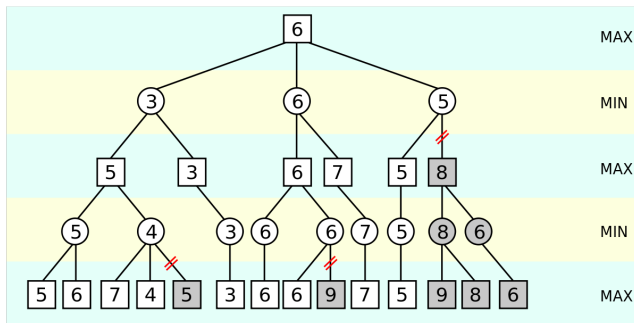


Rysunek 5: Minimax



# Algorytmy

## Alpha-Beta

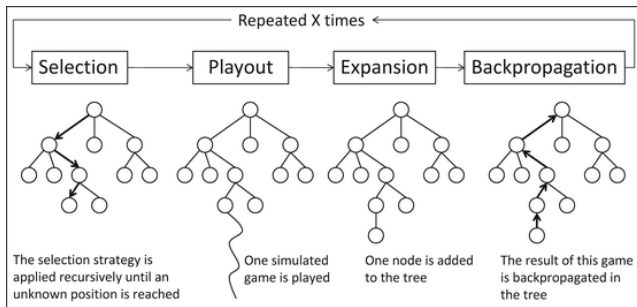


Rysunek 6: Alpha-Beta - Wikipedia



# Algorytmy

## MCTS



Rysunek 7: Monte-Carlo Tree Search in Board Games



# Algorytmy

## MCTS

- Selekcja - wybieramy ścieżkę od początkowego wężła do najbardziej obiecującego liści.





# Algorytmy

## MCTS

- Selekcja - wybieramy ścieżkę od początkowego węzła do najbardziej obiecującego liścia.

$$UCB(node_i) = \frac{w_i}{n_i} + c \sqrt{\frac{\log N}{n_i}} \quad (1)$$

- Ekspansja - rozwinięcie, wybieramy losowy węzeł z ostatniego liścia.
- Symulacja (Roll-out) - rozgrywamy wiele gier losowo zapamiętując wyniki.
- Propagacja wsteczna - aktualizujemy wartości wcześniejszych węzłów.



# Algorytmy

## Projekt

- Minimax, Alpha-Beta, Expectimax - *Project 2: Multi-Agent Search, Berkeley.*
- MCTS - implementacja we własnym projekcie.



# Uczenie ze wzmocnieniem

## Podstawowe pojęcia



# Podstawowe pojęcia

Interakcja agent - środowisko



# Podstawowe pojęcia

Środowisko (ang. *Environment*, np. plansza do gry Pacman):

- Opisuje świat, z którym agent wchodzi w interakcję
- Wejście:
  - Akcja
- Wyjście:
  - Stan
  - Nagroda



# Podstawowe pojęcia

Agent (ang. *Agent*):

- Poprzez interakcję ze środowiskiem uczy się, jak osiągnąć założony cel
- Wejście:
  - Stan
  - Nagroda
- Wyjście:
  - Akcja



# Podstawowe pojęcia

Nagroda (ang. *Reward*):

- Wartość zwracana przez środowisko w momencie wykonania akcji wybranej przez agenta.
- Reprezentuje cel, lub cele, jakie agent ma osiągnąć.
- Oznaczenie:  $r_t$  - nagroda otrzymana w chwili czasu  $t$ .



# Podstawowe pojęcia

Oczekiwana nagroda (ang. *Return*):

- Oczekiwana nagroda po zakończeniu bieżącego epizodu.
- Celem uczenia za wzmocnieniem jest maksymalizacja nagrody oczekiwanej.
- Oznaczenie:  $G_t$  - oczekiwana nagroda w chwili czasu  $t$ .

$$G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T \quad (2)$$





# Podstawowe pojęcia

Akcja (ang. *Action*):

- Dyskretna (1 z  $N$  dostępnych w danym środowisku).
- Ciągła (wartość lub wektor wartości).



# Podstawowe pojęcia

Strategia (ang. *Policy*):

- Zasady, według których wybierana jest akcja w danym stanie.
- Może być deterministyczna lub stochastyczna.
- Oznaczenie:  $\pi$ .



# Podstawowe pojęcia

Funkcja wartości (oceny, ang. *Value function*):

- Określa, jak dobrze być w danym stanie.
- Oczekiwana suma nagród, jaką możemy otrzymać, rozpoczynając w stanie  $s$  i działając zgodnie ze strategią  $\pi$ .
- Rodzaje:
  - Funkcja wartości stanu  $V^\pi(s)$  - jak dobrze być w stanie  $s$ , działając zgodnie ze strategią  $\pi$ .
  - Funkcja wartości stanu-akcji  $Q^\pi(s, a)$  - jak dobrze będąc w stanie  $s$  jest wykonać akcję  $a$ , działając zgodnie ze strategią  $\pi$ .



# Podstawowe pojęcia

Stan (ang. *State*):

- Zbiór wartości opisujących aktualną sytuację.
- Jest podstawą wyboru akcji przez agenta zgodnie z jego strategią.



# Podstawowe pojęcia

## Proces decyzyjny Markowa

### Własność Markowa

Własność procesów stochastycznych polegająca na tym, że warunkowe rozkłady prawdopodobieństwa przyszłych stanów procesu są zdeterminowane wyłącznie przez jego bieżący stan, bez względu na przeszłość.

Wikipedia



# Podstawowe pojęcia

## Proces decyzyjny Markowa

### Proces decyzyjny Markowa (ang. *Markov Decision Process* (MDP))

Ciąg zdarzeń, w którym prawdopodobieństwo każdego zdarzenia zależy jedynie od wyniku poprzedniego. W ujęciu matematycznym, procesy Markowa to takie procesy stochastyczne, które spełniają własność Markowa.

Wikipedia



# Podstawowe pojęcia

Proces decyzyjny Markowa

Elementy MDP:



# Podstawowe pojęcia

## Proces decyzyjny Markowa

### Elementy MDP:

- Stany:
  - $S$  - zbiór wszystkich możliwych stanów.
  - $s$  - pojedynczy stan ( $s \in S$ ).
  - $s_0$  - stan początkowy ( $s_0 \in S$ ).





# Podstawowe pojęcia

## Proces decyzyjny Markowa

### Elementy MDP:

#### ■ Stany:

- $S$  - zbiór wszystkich możliwych stanów.
- $s$  - pojedynczy stan ( $s \in S$ ).
- $s_0$  - stan początkowy ( $s_0 \in S$ ).

#### ■ Akcje:

- $A$  - zbiór wszystkich możliwych akcji.
- $a$  - pojedyncza akcja.
- $A(s)$  - zbiór akcji możliwych do wykonania w stanie  $s$ .



# Podstawowe pojęcia

## Proces decyzyjny Markowa

### Elementy MDP:

#### ■ Stany:

- $S$  - zbiór wszystkich możliwych stanów.
- $s$  - pojedynczy stan ( $s \in S$ ).
- $s_0$  - stan początkowy ( $s_0 \in S$ ).

#### ■ Akcje:

- $A$  - zbiór wszystkich możliwych akcji.
- $a$  - pojedyncza akcja.
- $A(s)$  - zbiór akcji możliwych do wykonania w stanie  $s$ .

#### ■ Model środowiska:

- $P(s'|s, a)$  - prawdopodobieństwo przejścia ze stanu  $s$  do stanu  $s'$ , wykonując akcję  $a$ .
- $P(s', r|s, a)$  - prawdopodobieństwo przejścia ze stanu  $s$  do stanu  $s'$  i otrzymania nagrody  $r$ , wykonując akcję  $a$ .



# Podstawowe pojęcia

## Proces decyzyjny Markowa

### Elementy MDP:

- Stany:
  - $S$  - zbiór wszystkich możliwych stanów.
  - $s$  - pojedynczy stan ( $s \in S$ ).
  - $s_0$  - stan początkowy ( $s_0 \in S$ ).
- Akcje:
  - $A$  - zbiór wszystkich możliwych akcji.
  - $a$  - pojedyncza akcja.
  - $A(s)$  - zbiór akcji możliwych do wykonania w stanie  $s$ .
- Model środowiska:
  - $P(s'|s, a)$  - prawdopodobieństwo przejścia ze stanu  $s$  do stanu  $s'$ , wykonując akcję  $a$ .
  - $P(s', r|s, a)$  - prawdopodobieństwo przejścia ze stanu  $s$  do stanu  $s'$  i otrzymania nagrody  $r$ , wykonując akcję  $a$ .
- Funkcja nagrody  $R(s)$ .



# Podstawowe pojęcia

## Proces decyzyjny Markowa



# Podstawowe pojęcia

## Funkcje wartości

- Funkcja wartości stanu dla strategii  $\pi$ .

$$v_{\pi}(s) \doteq \mathbb{E}[G_t | S_t = s] \quad (3)$$



# Podstawowe pojęcia

## Funkcje wartości

- Funkcja wartości stanu dla strategii  $\pi$ .

$$v_{\pi}(s) \doteq \mathbb{E}[G_t | S_t = s] \quad (3)$$

- Funkcja wartości stanu-akcji dla strategii  $\pi$ .

$$q_{\pi}(s, a) \doteq \mathbb{E}[G_t | S_t = s, A_t = a] \quad (4)$$



# Podstawowe pojęcia

## Funkcje wartości

- Funkcja wartości stanu dla strategii  $\pi$ .

$$v_{\pi}(s) \doteq \mathbb{E}[G_t | S_t = s] \quad (3)$$

- Funkcja wartości stanu-akcji dla strategii  $\pi$ .

$$q_{\pi}(s, a) \doteq \mathbb{E}[G_t | S_t = s, A_t = a] \quad (4)$$

- Funkcje wartości mogą być obliczane na podstawie doświadczenia.



# Podstawowe pojęcia

## Równanie Bellmana

$$\begin{aligned}v_{\pi}(s) &\doteq \mathbb{E}[G_t | S_t = s] \\&= \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t = s] \\&= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r | s, a) [r + \gamma \mathbb{E}[G_{t+1} | S_{t+1} = s]] \\&= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r | s, a) [r + \gamma v_{\pi}(s')], \text{ dla wszystkich } s \in S.\end{aligned}$$

(5)





# Podstawowe pojęcia

## Równanie Bellmana

$$v_{\pi}(s) \doteq \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) [r + \gamma v_{\pi}(s')]$$




# Podstawowe pojęcia

## Równanie Bellmana

Prawdopodobieństwo wyboru

akcji  $a$  w stanie  $s$  zgodnie

z założoną strategią  $\pi$

$$v_{\pi}(s) \doteq \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) [r + \gamma v_{\pi}(s')]$$




# Podstawowe pojęcia

## Równanie Bellmana

Prawdopodobieństwo wyboru

akcji  $a$  w stanie  $s$  zgodnie

z założoną strategią  $\pi$

$$v_{\pi}(s) \doteq \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) [r + \gamma v_{\pi}(s')]$$

Prawdopodobieństwo przejścia

do stanu  $s'$ , i otrzymania

nagrody  $r$ , ze stanu  $s$

wybierając akcję  $a$  zgodnie

z założoną strategią  $\pi$



# Podstawowe pojęcia

## Równanie Bellmana

Prawdopodobieństwo wyboru  
akcji  $a$  w stanie  $s$  zgodnie  
z założoną strategią  $\pi$

$$v_{\pi}(s) \doteq \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) [r + \gamma v_{\pi}(s')]$$

Prawdopodobieństwo przejścia  
do stanu  $s'$ , i otrzymania  
nagrody  $r$ , ze stanu  $s$   
wybierając akcję  $a$  zgodnie  
z założoną strategią  $\pi$

Nagroda otrzymana w  
wyniku wykonania akcji  $a$   
oraz przejścia do stanu  $s'$



# Podstawowe pojęcia

## Równanie Bellmana

Prawdopodobieństwo wyboru  
akcji  $a$  w stanie  $s$  zgodnie  
z założoną strategią  $\pi$

$$v_{\pi}(s) \doteq \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) [r + \gamma v_{\pi}(s')]$$

Prawdopodobieństwo przejścia  
do stanu  $s'$ , i otrzymania  
nagrody  $r$ , ze stanu  $s$   
wybierając akcję  $a$  zgodnie  
z założoną strategią  $\pi$

Nagroda otrzymana w  
wyniku wykonania akcji  $a$   
oraz przejścia do stanu  $s'$

Przewidywana skumulowana  
nagroda w stanie  $s'$



# Podstawowe pojęcia

## Materiały uzupełniające

- Książka *Reinforcement Learning: An Introduction*, Richard S. Sutton and Andrew G. Barto, wydanie drugie, 2018.
  - Rozdział 1 - *Introduction*.
  - Rozdział 3 - *Finite Markov Decision Processes*.
- Książka *Grokking Deep Reinforcement Learning*, Miguel Morales, 2020.
  - Rozdział 2 - *Mathematical foundations of reinforcement learning*.
- Video *RL Course by David Silver - Lecture 1: Introduction to Reinforcement Learning*.



# Uczenie pasywne

(ang. *model based learning*)



# Uczenie pasywne

Algorytmy:

- Ocena strategii (ang. *Policy Evaluation*).
- Polepszanie strategii (ang. *Policy Improvement*).
- Iteracyjne doskonalenie strategii (ang. *Policy Iteration*).
- Iteracyjne obliczanie funkcji wartości (ang. *Value Iteration*).





# Uczenie pasywne

## Ocena strategii



# Uczenie pasywne

Ocena strategii (ang. *Policy Evaluation*)

Cele:

- Oszacowanie wartości dla każdego stanu, dla założonej strategii.
- Wyznaczenie  $v_{\pi}(s)$  dla każdego  $s \in S$ , dla założonej strategii  $\pi$ .



# Uczenie pasywne

Ocena strategii (ang. *Policy Evaluation*)

Cele:

- Oszacowanie wartości dla każdego stanu, dla założonej strategii.
- Wyznaczenie  $v_\pi(s)$  dla każdego  $s \in S$ , dla założonej strategii  $\pi$ .

Rozwiązanie:

- Metoda iteracyjna:

$$V_{k+1}(s) = \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) [r + \gamma V_k(s')]$$



# Uczenie pasywne

Ocena strategii (ang. *Policy Evaluation*)

Cele:

- Oszacowanie wartości dla każdego stanu, dla założonej strategii.
- Wyznaczenie  $v_\pi(s)$  dla każdego  $s \in S$ , dla założonej strategii  $\pi$ .

Rozwiązanie:

- Metoda iteracyjna:

$$V_{k+1}(s) = \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) [r + \gamma V_k(s')]$$

Warianty:

- Dwie macierze.
- Obliczenia w miejscu.



# Uczenie pasywne

Ocena strategii (ang. *Policy Evaluation*)

## Ocena strategii

**Wejście:**

- $\pi$  - strategia, która ma zostać oszacowana.
- $\theta$  - dokładność szacowania strategii.

**Wyjście:**

- $V(s)$  - funkcja wartości stanów wyznaczona dla strategii  $\pi$ .

Inicjalizacja tablicy wartości stanów  $V(s)$  losowymi wartościami, za wyjątkiem stanu końcowego, któremu przypisana jest wartość 0.

Licz:

$$\Delta \leftarrow 0$$

Dla każdego  $s \in S$ :

$$v \leftarrow V(s)$$

$$V(s) \leftarrow \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) [r + \gamma v_\pi(s')]$$

$$\Delta \leftarrow \max(\Delta, |v - V(s)|)$$

Dopóki  $\Delta > \theta$

# Uczenie pasywne

Ocena strategii (ang. *Policy Evaluation*)

## Iterative Policy Evaluation

Input  $\pi$ , the policy to be evaluated

Algorithm parameter: a small threshold  $\theta > 0$  determining accuracy of estimation

Initialize  $V(s)$ , for all  $s \in S^+$ , arbitrarily except that  $V(\text{terminal}) = 0$

Loop:

$\Delta \leftarrow 0$

Loop for each  $s \in S$ :

$v \leftarrow V(s)$

$V(s) \leftarrow \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) [r + \gamma v_\pi(s')]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

until  $\Delta > \theta$

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



# Uczenie pasywne

Ocena strategii (ang. *Policy Evaluation*) - przykład liczbowy

**Model środowiska:**

**Strategia:**

W każdym stanie prawdopodobieństwo wyboru każdej z możliwych akcji jest takie samo.



# Uczenie pasywne

Ocena strategii (ang. *Policy Evaluation*) - przykład liczbowy cd.

Wejście:

- Stany:

- $S = \{s_0, s_1, s_2\}$ .

- Akcje:

- $A_{s_0} = \{a_0, a_1\}$ .

- $A_{s_1} = \{a_0, a_1\}$ .

- $A_{s_2} = \{a_0, a_1\}$ .

- Funkcja wartości:

- $V(s_0) = 0, V(s_1) = 0, V(s_2) = 0$ .

- Strategia:

- $\pi(a_0|s_0) = \pi(a_1|s_0) = 0.5$ .

- $\pi(a_0|s_1) = \pi(a_1|s_1) = 0.5$ .

- $\pi(a_0|s_2) = \pi(a_1|s_2) = 0.5$ .





# Uczenie pasywne

Ocena strategii (ang. *Policy Evaluation*) - przykład liczbowy cd.

Wejście:

- Prawdopodobieństwo przejścia:

- $p(s_2|s_0, a_0) = 0.5$ ,  $p(s_0|s_0, a_0) = 0.5$ ,  $p(s_2|s_0, a_1) = 1$ .
- $p(s_0|s_1, a_0) = 0.7$ ,  $p(s_1|s_1, a_0) = 0.1$ ,  $p(s_2|s_1, a_0) = 0.2$ ,  
 $p(s_1|s_1, a_1) = 0.95$ ,  $p(s_2|s_1, a_1) = 0.05$ .
- $p(s_0|s_2, a_0) = 0.4$ ,  $p(s_2|s_2, a_0) = 0.6$ ,  $p(s_0|s_2, a_1) = 0.3$ ,  
 $p(s_1|s_2, a_1) = 0.3$ ,  $p(s_2|s_2, a_1) = 0.4$ .

- Nagroda:

- $r(s_1, a_0, s_0) = 5$ .
- $r(s_2, a_1, s_0) = -1$ .
- W pozostałych przypadkach:  $r = 0$ .

- Discount factor:

- $\gamma = 0.9$ .



# Uczenie pasywne

Ocena strategii (ang. *Policy Evaluation*) - przykład liczbowy cd.

Dla każdego  $s \in S$ :

$$V_{k+1}(s) = \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) [r + \gamma V_k(s')]$$

Krok 1 (dla stanu  $s_0$ ):

Potrzebne dane:

- $\pi(a_0|s_0) = \pi(a_1|s_0) = 0.5$ .
- $p(s_2|s_0, a_0) = 0.5$ ,  $p(s_0|s_0, a_0) = 0.5$ ,  $p(s_2|s_0, a_1) = 1$ .
- $r = 0$ .
- $V_0(s_0) = V_0(s_1) = V_0(s_2) = 0$



# Uczenie pasywne

Ocena strategii (ang. *Policy Evaluation*) - przykład liczbowy cd.

Krok 1 (dla stanu  $s_0$ ):

$$V_1(s_0) = \pi(a_0|s_0) * [p(s_2|s_0, a_0) * (r + \gamma * V_0(s_2)) + \\ p(s_0|s_0, a_0) * (r + \gamma * V_0(s_0))] + \\ \pi(a_1|s_0) * [p(s_2|s_0, a_1) * (r + \gamma * V_0(s_2))]$$

Po podstawieniu wartości otrzymujemy:

$$V_1(s_0) = 0.5 * [0.5 * (0 + 0.9 * 0) + \\ 0.5 * (0 + 0.9 * 0)] + \\ 0.5 * [1 * (0 + 0.9 * 0)]$$

Ostatecznie otrzymujemy:

$$V_1(s_0) = 0$$



# Uczenie pasywne

Ocena strategii (ang. *Policy Evaluation*) - przykład liczbowy cd.

Dla każdego  $s \in S$ :

$$V_{k+1}(s) = \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) [r + \gamma V_k(s')]$$

Krok 1 (dla stanu  $s_1$ ):

Potrzebne dane:

- $\pi(a_0|s_1) = \pi(a_1|s_1) = 0.5$ .
- $p(s_0|s_1, a_0) = 0.7$ ,  $p(s_1|s_1, a_0) = 0.1$ ,  $p(s_2|s_1, a_0) = 0.2$ ,  
 $p(s_1|s_1, a_1) = 0.95$ ,  $p(s_2|s_1, a_1) = 0.05$ .
- $r(s_1, a_0, s_0) = 5$ ,  $r = 0$ .
- $V_0(s_0) = V_0(s_1) = V_0(s_2) = 0$



# Uczenie pasywne

Ocena strategii (ang. *Policy Evaluation*) - przykład liczbowy cd.

Krok 1 (dla stanu  $s_1$ ):

$$\begin{aligned}
 V_1(s_1) = & \pi(a_0|s_1) * [p(s_0|s_1, a_0) * (r(s_1, a_0, s_0) + \gamma * V_0(s_0)) + \\
 & p(s_1|s_1, a_0) * (r + \gamma * V_0(s_1)) + \\
 & p(s_2|s_1, a_0) * (r + \gamma * V_0(s_2))] + \\
 & \pi(a_1|s_1) * [p(s_1|s_1, a_1) * (r + \gamma * V_0(s_1)) + \\
 & p(s_2|s_1, a_1) * (r + \gamma * V_0(s_2))]
 \end{aligned}$$

Po podstawieniu wartości otrzymujemy:

$$\begin{aligned}
 V_1(s_1) = & 0.5 * [0.7 * (5 + 0.9 * 0) + \\
 & 0.1 * (0 + 0.9 * 0) + \\
 & 0.2 * (0 + 0.9 * 0)] + \\
 & 0.5 * [0.95 * (0 + 0.9 * 0) + \\
 & 0.05 * (0 + 0.9 * 0)]
 \end{aligned}$$

Ostatecznie otrzymujemy:

$$V_1(s_1) = 1.75$$



# Uczenie pasywne

Ocena strategii (ang. *Policy Evaluation*) - przykład liczbowy cd.

Dla każdego  $s \in S$ :

$$V_{k+1}(s) = \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) [r + \gamma V_k(s')]$$

Krok 1 (dla stanu  $s_2$ ):

Potrzebne dane:

- $\pi(a_0|s_2) = \pi(a_1|s_2) = 0.5$ .
- $p(s_0|s_2, a_0) = 0.4$ ,  $p(s_2|s_2, a_0) = 0.6$ ,  $p(s_0|s_2, a_1) = 0.3$ ,  
 $p(s_1|s_2, a_1) = 0.3$ ,  $p(s_2|s_2, a_1) = 0.4$ .
- $r(s_2, a_1, s_0) = -1$ ,  $r = 0$ .
- $V_0(s_0) = V_0(s_1) = V_0(s_2) = 0$



# Uczenie pasywne

Ocena strategii (ang. *Policy Evaluation*) - przykład liczbowy cd.

Krok 1 (dla stanu  $s_2$ ):

$$\begin{aligned}
 V_1(s_2) = & \pi(a_0|s_2) * [p(s_0|s_2, a_0) * (r + \gamma * V_0(s_0)) + \\
 & p(s_2|s_2, a_0) * (r + \gamma * V_0(s_2))] + \\
 & \pi(a_1|s_2) * [p(s_0|s_2, a_1) * (r(s_2, a_1, s_0) + \gamma * V_0(s_0)) + \\
 & p(s_1|s_2, a_1) * (r + \gamma * V_0(s_1)) + \\
 & p(s_2|s_2, a_1) * (r + \gamma * V_0(s_2))]
 \end{aligned}$$

Po podstawieniu wartości otrzymujemy:

$$\begin{aligned}
 V_1(s_2) = & 0.5 * [0.4 * (0 + 0.9 * 0) + \\
 & 0.6 * (0 + 0.9 * 0)] + \\
 & 0.5 * [0.3 * (-1 + 0.9 * 0) + \\
 & 0.4 * (0 + 0.9 * 0) + \\
 & 0.3 * (0 + 0.9 * 0)]
 \end{aligned}$$

Ostatecznie otrzymujemy:

$$V_1(s_2) = -0.15$$



# Uczenie pasywne

Ocena strategii (ang. *Policy Evaluation*) - przykład liczbowy cd.

Wartości funkcji dla poszczególnych stanów wyznaczone po  $n$  krokach algorytmu szacowania strategii.

Stan	Start	Krok 1	Krok 5	Krok 25	Krok 73
$V(s_0)$	0	0	0.24	1.32	1.47
$V(s_1)$	0	1.75	3.31	4.40	4.55
$V(s_2)$	0	-0.15	0.46	1.54	1.69





# Uczenie pasywne

Ocena strategii (ang. *Policy Evaluation*) - przykład liczbowy cd.

Wartości funkcji dla poszczególnych stanów wyznaczone po  $n$  krokach algorytmu szacowania strategii - implementacja *in-place*.

Stan	Start	Krok 1	Krok 5	Krok 25	Krok 53
$V(s_0)$	0	0	0.49	1.42	1.47
$V(s_1)$	0	1.75	3.56	4.51	4.55
$V(s_2)$	0	0.09	0.76	1.64	1.69



# Uczenie pasywne

Ocena strategii (ang. *Policy Evaluation*) - ćwiczenie

Implementacja algorytmu oceny strategii w pliku *PolicyEvaluation.py*:

- implementacja algorytmu z wykorzystaniem dwóch osobnych macierzy do obliczeń (funkcja *policy\_eval\_two\_arrays*),
- implementacja algorytmu z wykorzystaniem obliczeń w miejscu (funkcja *policy\_eval\_in\_place*).



# Uczenie pasywne

## Poprawa strategii



# Uczenie pasywne

Poprawa strategii (ang. *Policy Improvement*)

Problem:

- Jak można poprawić aktualną strategię?

Twierdzenie o poprawie strategii (ang. *Policy Improvement Theorem*)

$$q_{\pi}(s, a) \doteq \sum_{s', r} p(s', r | s, a) [r + \gamma v_{\pi}(s')] \quad (6)$$

$$\forall s \in S, q_{\pi}(s, \pi'(s)) \geq v_{\pi}(s) \rightarrow v'_{\pi}(s) \geq v_{\pi}(s) \quad (7)$$



# Uczenie pasywne

Poprawa strategii (ang. *Policy Improvement*)

Zgodnie z twierdzeniem o poprawie strategii, zastosowanie zachłannej strategii zawsze będzie lepsze, bądź równe obecnej strategii:

$$\pi'(s) = \operatorname{argmax}_a \sum_{s'} \sum_r p(s', r | s, a) [r + \gamma v_{\pi}(s')] \quad (8)$$



# Uczenie pasywne

## Iteracyjne doskonalenie strategii



# Uczenie pasywne

Iteracyjne doskonalenie strategii (ang. *Policy Iteration*)

Cel:

- Określenie optymalnej strategii.



# Uczenie pasywne

Iteracyjne doskonalenie strategii (ang. *Policy Iteration*)

Cel:

- Określenie optymalnej strategii.

Rozwiązanie:

- Oszacowanie aktualnej strategii (algorytm *Policy Evaluation*).
- Poprawienie aktualnej strategii (algorytm *Policy Improvement*).





# Uczenie pasywne

Iteracyjne doskonalenie strategii (ang. *Policy Iteration*)

Cel:

- Określenie optymalnej strategii.

Rozwiązanie:

- Oszacowanie aktualnej strategii (algorytm *Policy Evaluation*).
- Poprawienie aktualnej strategii (algorytm *Policy Improvement*).

Optymalna strategia:

- Jeżeli zastosowanie algorytmu *Policy Improvement* na aktualnej strategii jej nie zmieni, to oznacza, że aktualna strategia jest optymalna.



# Uczenie pasywne

Iteracyjne doskonalenie strategii (ang. *Policy Iteration*)

## Iteracyjne doskonalenie strategii (ang. *Policy Iteration*)

Wyjście:

- $V \approx v_*$  - optymalna funkcja wartości stanów.
- $\pi \approx \pi_*$  - optymalna strategia.

Inicjalizacja tablicy strategii losowymi akcjami  $\pi(s) \in A(s)$  dla  $s \in S$ .

Licz:

*strategia\_stabilna*  $\leftarrow$  *true*

$V \leftarrow \text{ocena\_strategii}(\pi)$

Dla każdego  $s \in S$ :

*poprzednia\_akcja*  $\leftarrow \pi(s)$

$\pi(s) \leftarrow \operatorname{argmax}_a \sum_{s'} \sum_r p(s', r | s, a) [r + \gamma V(s')]$

Jeżeli *poprzednia\_akcja*  $\neq \pi(s)$  ustaw *strategia\_stabilna*  $\leftarrow$  *false*

Dopóki *strategia\_stabilna*  $\neq$  *true*



# Uczenie pasywne

Iteracyjne doskonalenie strategii (ang. *Policy Iteration*)

## Policy Iteration

1. Initialization

$V(s) \in \mathbb{R}$  and  $\pi(s) \in \mathbf{A}(s)$  arbitrarily for all  $s \in S$ .

2. Policy Evaluation

3. Policy Improvement

Loop:

$policy\_stable \leftarrow true$

For each  $s \in S$ :

$old\_action \leftarrow \pi(s)$

$\pi(s) \leftarrow \operatorname{argmax}_a \sum_{s'} p(s', r|s, a)[r + \gamma V(s')]$

If  $old\_action \neq \pi(s)$ , then  $policy\_stable \leftarrow false$

If  $policy\_stable == false$ , then stop and return  $V \approx v_*$  and  $\pi \approx \pi_*$ ; else go to 2

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



# Uczenie pasywne

Iteracyjne doskonalenie strategii (ang. *Policy Iteration*) - przykład liczbowy

**Model środowiska:**

**Strategia:**

W każdym stanie wybieramy akcję  $a_0$ .



# Uczenie pasywne

Iteracyjne doskonalenie strategii (ang. *Policy Iteration*) - przykład liczbowy cd.

Wejście:

- Stany:

- $S = \{s_0, s_1, s_2\}$ .

- Akcje:

- $A_{s_0} = \{a_0, a_1\}$ .

- $A_{s_1} = \{a_0, a_1\}$ .

- $A_{s_2} = \{a_0, a_1\}$ .

- Strategia:

- $\pi(s_0) = a_0$ .

- $\pi(s_1) = a_0$ .

- $\pi(s_2) = a_0$ .



# Uczenie pasywne

Iteracyjne doskonalenie strategii (ang. *Policy Iteration*) - przykład liczbowy cd.

Wejście:

- Prawdopodobieństwo przejścia:

- $p(s_2|s_0, a_0) = 0.5$ ,  $p(s_0|s_0, a_0) = 0.5$ ,  $p(s_2|s_0, a_1) = 1$ .
- $p(s_0|s_1, a_0) = 0.7$ ,  $p(s_1|s_1, a_0) = 0.1$ ,  $p(s_2|s_1, a_0) = 0.2$ ,  
 $p(s_1|s_1, a_1) = 0.95$ ,  $p(s_2|s_1, a_1) = 0.05$ .
- $p(s_0|s_2, a_0) = 0.4$ ,  $p(s_2|s_2, a_0) = 0.6$ ,  $p(s_0|s_2, a_1) = 0.3$ ,  
 $p(s_1|s_2, a_1) = 0.3$ ,  $p(s_2|s_2, a_1) = 0.4$ .

- Nagroda:

- $r(s_1, a_0, s_0) = 5$ .
- $r(s_2, a_1, s_0) = -1$ .
- W pozostałych przypadkach:  $r = 0$ .

- Discount factor:

- $\gamma = 0.9$ .



# Uczenie pasywne

Iteracyjne doskonalenie strategii (ang. *Policy Iteration*) - przykład liczbowy cd.

Funkcja wartości obliczona dla strategii  $\pi_0$  (za pomocą algorytmu *Policy Evaluation*):

- $v_{\pi_0}(s_0) = 0.$
- $v_{\pi_0}(s_1) = 3.87.$
- $v_{\pi_0}(s_2) = 0.$



# Uczenie pasywne

Iteracyjne doskonalenie strategii (ang. *Policy Iteration*) - przykład liczbowy cd.

Poprawa strategii w stanie  $s_0$ :

$$\pi_1(s_0) = \operatorname{argmax}_a \sum_{s', r} p(s', r | s, a) [r + \gamma v_{\pi_0}(s')]$$

Potrzebne dane:

- $\pi(s_0) = a_0$ .
- $p(s_2 | s_0, a_0) = 0.5$ ,  $p(s_0 | s_0, a_0) = 0.5$ ,  $p(s_2 | s_0, a_1) = 1$ .
- $r = 0$ .
- $v_{\pi_0}(s_0) = 0.0$ ,  $v_{\pi_0}(s_1) = 3.87$ ,  $v_{\pi_0}(s_2) = 0.0$ .
- $\gamma = 0.9$ .





# Uczenie pasywne

Iteracyjne doskonalenie strategii (ang. *Policy Iteration*) - przykład liczbowy cd.

Poprawa strategii w stanie  $s_0$ :

$$\begin{aligned}\pi_1(s_0) = \operatorname{argmax}_a (& p(s_2|s_0, a_0)[r + \gamma v_{\pi_0}(s_2)] + \\ & p(s_0|s_0, a_0)[r + \gamma v_{\pi_0}(s_0)], \\ & p(s_2|s_0, a_1)[r + \gamma v_{\pi_0}(s_2)])\end{aligned}$$

Po podstawieniu wartości otrzymujemy:

$$\begin{aligned}\pi_1(s_0) = \operatorname{argmax}_a (& 0.5[0 + 0.9 * 0] + \\ & 0.5[0 + 0.9 * 0], \\ & 1[0 + 0.9 * 0]) \\ = \operatorname{argmax}_a (& 0, 0)\end{aligned}$$

Ostatecznie nową akcją wybraną dla stanu  $s_0$  jest akcja  $a_0$ , czyli:

$$\pi_1(s_0) = a_0$$



# Uczenie pasywne

Iteracyjne doskonalenie strategii (ang. *Policy Iteration*) - przykład liczbowy cd.

Poprawa strategii w stanie  $s_1$ :

$$\pi_1(s_1) = \operatorname{argmax}_a \sum_{s', r} p(s', r | s, a) [r + \gamma v_{\pi_0}(s')]$$

Potrzebne dane:

- $\pi(s_1) = a_0$ .
- $p(s_0 | s_1, a_0) = 0.7$ ,  $p(s_1 | s_1, a_0) = 0.1$ ,  $p(s_2 | s_1, a_0) = 0.2$ ,  
 $p(s_1 | s_1, a_1) = 0.95$ ,  $p(s_2 | s_1, a_1) = 0.05$ .
- $r(s_1, a_0, s_0) = 5$ ,  $r = 0$ .
- $v_{\pi_0}(s_0) = 0.0$ ,  $v_{\pi_0}(s_1) = 3.87$ ,  $v_{\pi_0}(s_2) = 0.0$ .
- $\gamma = 0.9$ .



# Uczenie pasywne

Iteracyjne doskonalenie strategii (ang. *Policy Iteration*) - przykład liczbowy cd.

Poprawa strategii w stanie  $s_1$ :

$$\begin{aligned}\pi_1(s_1) = \operatorname{argmax}_a (& p(s_0|s_1, a_0)[r + \gamma v_{\pi_0}(s_0)] + \\ & p(s_1|s_1, a_0)[r + \gamma v_{\pi_0}(s_1)] + \\ & p(s_2|s_1, a_0)[r + \gamma v_{\pi_0}(s_2)], \\ & p(s_1|s_1, a_1)[r + \gamma v_{\pi_0}(s_1)] + \\ & p(s_2|s_1, a_1)[r + \gamma v_{\pi_0}(s_2)])\end{aligned}$$

Po podstawieniu wartości otrzymujemy:

$$\begin{aligned}\pi_1(s_1) = \operatorname{argmax}_a (& 0.7[5 + 0.9 * 0] + \\ & 0.1[0 + 0.9 * 3.87] + \\ & 0.2[0 + 0.9 * 0], \\ & 0.95[0 + 0.9 * 3.87] + \\ & 0.05[0 + 0.9 * 0]) \\ = \operatorname{argmax}_a (& 3.85, 3.31)\end{aligned}$$

Ostatecznie nową akcją wybraną dla stanu  $s_1$  jest akcja  $a_0$ , czyli:

$$\pi_1(s_1) = a_0$$



# Uczenie pasywne

Iteracyjne doskonalenie strategii (ang. *Policy Iteration*) - przykład liczbowy cd.

Poprawa strategii w stanie  $s_2$ :

$$\pi_1(s_2) = \operatorname{argmax}_a \sum_{s', r} p(s', r | s, a) [r + \gamma v_{\pi_0}(s')]$$

Potrzebne dane:

- $\pi(s_2) = a_0$ .
- $p(s_0 | s_2, a_0) = 0.4$ ,  $p(s_2 | s_2, a_0) = 0.6$ ,  $p(s_0 | s_2, a_1) = 0.3$ ,  
 $p(s_1 | s_2, a_1) = 0.3$ ,  $p(s_2 | s_2, a_1) = 0.4$ .
- $r(s_2, a_1, s_0) = -1$ ,  $r = 0$ .
- $v_{\pi_0}(s_0) = 0.0$ ,  $v_{\pi_0}(s_1) = 3.87$ ,  $v_{\pi_0}(s_2) = 0.0$ .
- $\gamma = 0.9$ .



# Uczenie pasywne

Iteracyjne doskonalenie strategii (ang. *Policy Iteration*) - przykład liczbowy cd.

Poprawa strategii w stanie  $s_2$ :

$$\begin{aligned}\pi_1(s_2) = \operatorname{argmax}_a (& p(s_0|s_2, a_0)[r + \gamma v_{\pi_0}(s_0)] + \\ & p(s_2|s_2, a_0)[r + \gamma v_{\pi_0}(s_2)], \\ & p(s_0|s_2, a_1)[r + \gamma v_{\pi_0}(s_0)] + \\ & p(s_1|s_2, a_1)[r + \gamma v_{\pi_0}(s_1)] + \\ & p(s_2|s_2, a_1)[r + \gamma v_{\pi_0}(s_2)])\end{aligned}$$

Po podstawieniu wartości otrzymujemy:

$$\begin{aligned}\pi_1(s_2) = \operatorname{argmax}_a (& 0.4[0 + 0.9 * 0] + \\ & 0.6[0 + 0.9 * 0], \\ & 0.3[-1 + 0.9 * 0] + \\ & 0.3[0 + 0.9 * 3.87] + \\ & 0.4[0 + 0.9 * 0]) \\ = \operatorname{argmax}_a (& 0, 0.741)\end{aligned}$$

Ostatecznie nową akcją wybraną dla stanu  $s_2$  jest akcja  $a_1$ , czyli:

$$\pi_1(s_2) = a_1$$



# Uczenie pasywne

Iteracyjne doskonalenie strategii (ang. *Policy Iteration*) - przykład liczbowy cd.

Wyniki działania algorytmu iteracji strategii po 1 kroku:

- Strategia:

- $\pi_1(s_0) = a_0.$

- $\pi_1(s_1) = a_0.$

- $\pi_1(s_2) = a_1.$

- Wartość funkcji:

- $v_{\pi_1}(s_0) = 2.83.$

- $v_{\pi_1}(s_1) = 6.49.$

- $v_{\pi_1}(s_2) = 3.47.$



# Uczenie pasywne

Iteracyjne doskonalenie strategii (ang. *Policy Iteration*) - przykład liczbowy cd.

Wyniki działania algorytmu iteracji strategii po 3 kroku:

- Optymalna strategia:

- $\pi_3(s_0) = a_1.$

- $\pi_3(s_1) = a_0.$

- $\pi_3(s_2) = a_1.$

- Wartość funkcji:

- $v_{\pi_3}(s_0) = 3.79.$

- $v_{\pi_3}(s_1) = 7.3.$

- $v_{\pi_3}(s_2) = 4.21.$



# Uczenie pasywne

Iteracyjne doskonalenie strategii (ang. *Policy Iteration*) - ćwiczenie

Implementacja algorytmu iteracyjnego doskonalenia strategii w pliku *PolicyIteration.py*:

- implementacja algorytmu oceny strategii z wykorzystaniem obliczeń w miejscu (funkcja *policy\_eval\_in\_place*),
- implementacja algorytmu poprawy strategii (funkcja *policy\_improvement*),
- implementacja algorytmu iteracyjnego doskonalenia strategii (funkcja *policy\_iteration*).





# Uczenie pasywne

## Iteracja funkcji wartości



# Uczenie pasywne

Iteracja funkcji wartości (ang. *Value Iteration*)

Cel:

- Optymalizacja wyznaczania najlepszej strategii.



# Uczenie pasywne

Iteracja funkcji wartości (ang. *Value Iteration*)

Cel:

- Optymalizacja wyznaczania najlepszej strategii.

Rozwiązanie:

- Połączenie algorytmów oceny strategii i poprawy strategii w pojedynczej aktualizacji funkcji wartości stanu.



# Uczenie pasywne

Iteracja funkcji wartości (ang. *Value Iteration*)

Cel:

- Optymalizacja wyznaczania najlepszej strategii.

Rozwiązanie:

- Połączenie algorytmów oceny strategii i poprawy strategii w pojedynczej aktualizacji funkcji wartości stanu.

Optymalna strategia:

- Strategia określana jest tylko jeden raz, na końcu działania algorytmu, na podstawie otrzymanej funkcji wartości.



# Uczenie pasywne

Iteracja funkcji wartości (ang. *Value Iteration*)

## Iteracja wartości (ang. *Value Iteration*)

**Wejście:**

- $\theta$  - dokładność obliczania funkcji wartości.

**Wyjście:**

- $\pi(s)$  - optymalna strategia  $\pi$ .

Inicjalizacja tablicy wartości stanów  $V(s)$  losowymi wartościami, za wyjątkiem stanu końcowego, któremu przypisana jest wartość 0.

Licz:

$$\Delta \leftarrow 0$$

Dla każdego  $s \in S$ :

$$v \leftarrow V(s)$$

$$V(s) \leftarrow \max_a \sum_{s'} \sum_r p(s', r | s, a) [r + \gamma V(s')]$$

$$\Delta \leftarrow \max(\Delta, |v - V(s)|)$$

Dopóki  $\Delta > \theta$

# Uczenie pasywne

Iteracja funkcji wartości (ang. *Value Iteration*)

## Iteracja wartości (ang. *Value Iteration*)

Wyznaczenie optymalnej strategii,  $\pi \approx \pi_*$ , na podstawie obliczonej funkcji wartości, z wykorzystaniem algorytmu zachłannego:

$$\pi(s) = \operatorname{argmax}_a \sum_{s'} \sum_r p(s', r | s, a) [r + \gamma V(s')] \quad (9)$$



# Uczenie pasywne

Iteracja funkcji wartości (ang. *Value Iteration*)

## Value Iteration

Algorithm parameter: a small threshold  $\theta > 0$  determining accuracy of estimation

Initialize  $V(s)$ , for all  $s \in S^+$ , arbitrarily except that  $V(\text{terminal}) = 0$

Loop:

$\Delta \leftarrow 0$

Loop for each  $s \in S$ :

$v \leftarrow V(s)$

$V(s) \leftarrow \max_a \sum_{s'} \sum_r p(s', r|s, a)[r + \gamma V(s')]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

Until  $\Delta > \theta$

Output a deterministic policy,  $\pi \approx \pi_*$ , such that

$$\pi(s) = \operatorname{argmax}_a \sum_{s'} \sum_r p(s', r|s, a)[r + \gamma V(s')] \quad (10)$$

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



# Uczenie pasywne

Iteracja funkcji wartości (ang. *Value Iteration*) - przykład liczbowy

**Model środowiska:**





# Uczenie pasywne

Iteracja funkcji wartości (ang. *Value Iteration*) - przykład liczbowy cd.

Wejście:

- Stany:

- $S = \{s_0, s_1, s_2\}$ .

- Akcje:

- $A_{s_0} = \{a_0, a_1\}$ .

- $A_{s_1} = \{a_0, a_1\}$ .

- $A_{s_2} = \{a_0, a_1\}$ .

- Funkcja wartości:

- $V(s_0) = 0$ .

- $V(s_1) = 0$ .

- $V(s_2) = 0$ .



# Uczenie pasywne

Iteracja funkcji wartości (ang. *Value Iteration*) - przykład liczbowy cd.

Wejście:

- Prawdopodobieństwo przejścia:

- $p(s_2|s_0, a_0) = 0.5$ ,  $p(s_0|s_0, a_0) = 0.5$ ,  $p(s_2|s_0, a_1) = 1$ .
- $p(s_0|s_1, a_0) = 0.7$ ,  $p(s_1|s_1, a_0) = 0.1$ ,  $p(s_2|s_1, a_0) = 0.2$ ,  
 $p(s_1|s_1, a_1) = 0.95$ ,  $p(s_2|s_1, a_1) = 0.05$ .
- $p(s_0|s_2, a_0) = 0.4$ ,  $p(s_2|s_2, a_0) = 0.6$ ,  $p(s_0|s_2, a_1) = 0.3$ ,  
 $p(s_1|s_2, a_1) = 0.3$ ,  $p(s_2|s_2, a_1) = 0.4$ .

- Nagroda:

- $r(s_1, a_0, s_0) = 5$ .
- $r(s_2, a_1, s_0) = -1$ .
- W pozostałych przypadkach:  $r = 0$ .

- Discount factor:

- $\gamma = 0.9$ .



# Uczenie pasywne

Iteracja funkcji wartości (ang. *Value Iteration*) - przykład liczbowy cd.

Aktualizowanie funkcji wartości w stanie  $s_0$ :

$$V(s_0) = \max_a \sum_{s'} \sum_r p(s', r|s, a)[r + \gamma V(s')]$$

Potrzebne dane:

- $p(s_2|s_0, a_0) = 0.5$ ,  $p(s_0|s_0, a_0) = 0.5$ ,  $p(s_2|s_0, a_1) = 1$ .
- $r = 0$ .
- $V(s_0) = 0.0$ ,  $V(s_1) = 0.0$ ,  $V(s_2) = 0.0$ .
- $\gamma = 0.9$ .



# Uczenie pasywne

Iteracja funkcji wartości (ang. *Value Iteration*) - przykład liczbowy cd.

Aktualizowanie funkcji wartości w stanie  $s_0$ :

$$V(s_0) = \max_a (p(s_2|s_0, a_0)[r + \gamma V(s_2)] + \\ p(s_0|s_0, a_0)[r + \gamma V(s_0)], \\ p(s_2|s_0, a_1)[r + \gamma V(s_2)])$$

Po podstawieniu wartości otrzymujemy:

$$V(s_0) = \max_a (0.5[0 + 0.9 * 0] + \\ 0.5[0 + 0.9 * 0], \\ 1[0 + 0.9 * 0]) \\ = \max_a (0, 0)$$

Ostatecznie:

$$V(s_0) = 0$$



# Uczenie pasywne

Iteracja funkcji wartości (ang. *Value Iteration*) - przykład liczbowy cd.

Aktualizowanie funkcji wartości w stanie  $s_1$ :

$$V(s_1) = \max_a \sum_{s'} \sum_r p(s', r|s, a)[r + \gamma V(s')]$$

Potrzebne dane:

- $p(s_0|s_1, a_0) = 0.7$ ,  $p(s_1|s_1, a_0) = 0.1$ ,  $p(s_2|s_1, a_0) = 0.2$ ,  
 $p(s_1|s_1, a_1) = 0.95$ ,  $p(s_2|s_1, a_1) = 0.05$ .
- $r(s_1, a_0, s_0) = 5$ ,  $r = 0$ .
- $V(s_0) = 0.0$ ,  $V(s_1) = 0.0$ ,  $V(s_2) = 0.0$ .
- $\gamma = 0.9$ .



# Uczenie pasywne

Iteracja funkcji wartości (ang. *Value Iteration*) - przykład liczbowy cd.

Poprawa strategii w stanie  $s_1$ :

$$\begin{aligned} V(s_1) = \max_a & (p(s_0|s_1, a_0)[r + \gamma V(s_0)] + \\ & p(s_1|s_1, a_0)[r + \gamma V(s_1)] + \\ & p(s_2|s_1, a_0)[r + \gamma V(s_2)], \\ & p(s_1|s_1, a_1)[r + \gamma V(s_1)] + \\ & p(s_2|s_1, a_1)[r + \gamma V(s_2)]) \end{aligned}$$

Po podstawieniu wartości otrzymujemy:

$$\begin{aligned} V(s_1) = \max_a & (0.7[5 + 0.9 * 0] + \\ & 0.1[0 + 0.9 * 0] + \\ & 0.2[0 + 0.9 * 0], \\ & 0.95[0 + 0.9 * 0] + \\ & 0.05[0 + 0.9 * 0]) \\ = \max_a & (3.5, 0) \end{aligned}$$

Ostatecznie:

$$V(s_1) = 3.5$$



# Uczenie pasywne

Iteracja funkcji wartości (ang. *Value Iteration*) - przykład liczbowy cd.

Aktualizowanie funkcji wartości w stanie  $s_2$ :

$$V(s_2) = \max_a \sum_{s'} \sum_r p(s', r|s, a)[r + \gamma V(s')]$$

Potrzebne dane:

- $p(s_0|s_2, a_0) = 0.4$ ,  $p(s_2|s_2, a_0) = 0.6$ ,  $p(s_0|s_2, a_1) = 0.3$ ,  
 $p(s_1|s_2, a_1) = 0.3$ ,  $p(s_2|s_2, a_1) = 0.4$ .
- $r(s_2, a_1, s_0) = -1$ ,  $r = 0$ .
- $V(s_0) = 0.0$ ,  $V(s_1) = 3.5$ ,  $V(s_2) = 0.0$ .
- $\gamma = 0.9$ .



# Uczenie pasywne

Iteracja wartości - przykład liczbowy cd.

Poprawa strategii w stanie  $s_2$ :

$$\begin{aligned}
 V(s_2) = \max_a (& p(s_0|s_2, a_0)[r + \gamma V(s_0)] + \\
 & p(s_2|s_2, a_0)[r + \gamma V(s_2)], \\
 & p(s_0|s_2, a_1)[r + \gamma V(s_0)] + \\
 & p(s_1|s_2, a_1)[r + \gamma V(s_1)] + \\
 & p(s_2|s_2, a_1)[r + \gamma V(s_2)])
 \end{aligned}$$

Po podstawieniu wartości otrzymujemy:

$$\begin{aligned}
 V(s_2) = \max_a (& 0.4[0 + 0.9 * 0] + \\
 & 0.6[0 + 0.9 * 0], \\
 & 0.3[-1 + 0.9 * 0] + \\
 & 0.3[0 + 0.9 * 3.5] + \\
 & 0.4[0 + 0.9 * 0]) \\
 = \max_a (& 0, 0.645)
 \end{aligned}$$

Ostatecznie:

$$V(s_2) = 0.645$$





# Uczenie pasywne

Iteracja funkcji wartości (ang. *Value Iteration*) - przykład liczbowy cd.

Wartości funkcji dla poszczególnych stanów wyznaczone po  $n$  krokach algorytmu iteracji wartości.

Stan	Start	Krok 1	Krok 5	Krok 25	Krok 37
$V(s_0)$	0	0	1.96	3.75	3.79
$V(s_1)$	0	3.5	5.6	7.26	7.3
$V(s_2)$	0	0.65	2.52	4.17	4.21



# Uczenie pasywne

Iteracja funkcji wartości (ang. *Value Iteration*) - przykład liczbowy cd.

Po zakończeniu obliczania funkcji wartości, następnym krokiem jest wyznaczenie optymalnej strategii dla każdego ze stanów za pomocą wzoru:

$$\pi(s) = \operatorname{argmax}_a \sum_{s'} \sum_r p(s', r | s, a) [r + \gamma V(s')]$$

Wyznaczona optymalna strategia wygląda następująco:

- $\pi(s_0) = a_1.$
- $\pi(s_1) = a_0.$
- $\pi(s_2) = a_1.$



# Uczenie pasywne

Iteracja funkcji wartości (ang. *Value Iteration*) - ćwiczenie

Implementacja algorytmu iteracyjnego obliczania funkcji wartości *Value Iteration.py*:

- implementacja algorytmu iteracyjnego obliczania funkcji wartości (funkcja *value\_iteration*).



# Uczenie pasywne

## Materiały uzupełniające

- Książka *Reinforcement Learning: An Introduction*, Richard S. Sutton and Andrew G. Barto, wydanie drugie, 2018.
  - Rozdział 4 - *Dynamic Programming*.
- Książka *Grokking Deep Reinforcement Learning*, Miguel Morales, 2020.
  - Rozdział 3 - *Balancing immediate and long-term goals*.
- Video *RL Course by David Silver - Lecture 3: Planning by Dynamic Programming*.
- Video *Artificial Intelligence Course by Pieter Abbeel - Lecture 8: Markov Decision Processes (MDPs)*.



# 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  $\pi$ , 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  $\pi$ :

$s_0, a_0, r_1, s_1, a_1, r_2, \dots, s_{T-1}, a_{T-1}, r_T$

$G \leftarrow 0$ :

Dla każdego kroku w epizodzie,  $t = T - 1, T - 2, \dots, 0$ :

$G \leftarrow \gamma G + r_{t+1}$

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 prediction - for estimating $V \approx v_\pi$

Input: a policy  $\pi$  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  $\pi$ :  $s_0, a_0, r_1, s_1, a_1, r_2, \dots, s_{T-1}, a_{T-1}, r_T$

$G \leftarrow 0$ :

Loop for each step of episode,  $t = T - 1, T - 2, \dots, 0$ :

$G \leftarrow \gamma G + r_{t+1}$

Unless  $s_t$  appears in  $s_0, s_1, \dots, s_{t+1}$ :

Append  $G$  to  $Returns(s_t)$

$V(s_t) \leftarrow \text{average}(Returns(s_t))$

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



# Uczenie aktywne

Monte Carlo

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  $\epsilon$ -zachłanny (ang.  *$\epsilon$ -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 $\epsilon$ -soft policies)* - oszacowanie

$$\pi \approx \pi_*$$

Parametry algorytmu: mała wartość  $\epsilon > 0$

Inicjalizacja:

- $\pi$  losowa  $\epsilon$ -mięka 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  $\pi$ :

$s_0, a_0, r_1, s_1, a_1, r_2, \dots, s_{T-1}, a_{T-1}, r_T$

$G \leftarrow 0$ :

Dla każdego kroku w epizodzie,  $t = T - 1, T - 2, \dots, 0$ :

$G \leftarrow \gamma G + r_{t+1}$

Jeżeli para  $s_t, a_t$  niepojawiła się wcześniej w sekwencji  $s_0, a_0, s_1, a_1, \dots, s_{t+1}, a_{t+1}$ :

Dodaj  $G$  do listy  $Returns(s_t, a_t)$

$Q(s_t, a_t) \leftarrow \text{average}(Returns(s_t, a_t))$

$a^* \leftarrow \arg\max_a Q(s_t, a)$

Dla każdej akcji  $a \in A(s_t)$ :

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \epsilon + \frac{\epsilon}{|A(s)|} & \text{if } a = a^* \\ \frac{\epsilon}{|A(s)|} & \text{if } a \neq a^* \end{cases} \quad (11)$$

# Uczenie aktywne

## Monte Carlo

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

Algorithm parameter: small  $\epsilon > 0$

Initialize:

- $\pi$  an arbitrary  $\epsilon$ -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  $\pi$ :  $s_0, a_0, r_1, s_1, a_1, r_2, \dots, s_{T-1}, a_{T-1}, r_T$

$G \leftarrow 0$ :

Loop for each step of episode,  $t = T - 1, T - 2, \dots, 0$ :

$G \leftarrow \gamma G + r_{t+1}$

Unless the pair  $s_t, a_t$  appears in  $s_0, a_0, s_1, a_1, \dots, s_{t+1}, a_{t+1}$ :

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  $a \in A(s_t)$ :

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \epsilon + \frac{\epsilon}{|A(s)|} & \text{if } a = a^* \\ \frac{\epsilon}{|A(s)|} & \text{if } a \neq a^* \end{cases} \quad (12)$$

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 konieczności oczekiwania na zakończenie epizodu.



# Uczenie aktywne

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

Szacowanie funkcji wartości za pomocą metod Monte Carlo:

$$V(s_t) \leftarrow V(s_t) + \alpha[G_t - V(s_t)] \quad (13)$$

Szacowanie funkcji wartości za pomocą metod różnic tymczasowych:

$$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)] \quad (14)$$



# Uczenie aktywne

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

## Tabelaryczny algorytm różnic tymczasowych z krokiem 1 do oszacowania $v_\pi$

Wejście: strategia do oszacowania  $\pi$

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 akcję  $a$  zgodnie ze strategią  $\pi$  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 $v_\pi$

Input: the policy  $\pi$  to be evaluated

Algorithm parameter: step size  $\alpha \in (0, 1]$

Initialize  $V(s)$ , for all  $s \in S^+$ , arbitrarily except that  $V(\text{terminal}) = 0$

Loop for each episode:

    Initialize  $s$

    Loop for each step of episode:

$a \leftarrow$  action given by  $\pi$  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:

$$\delta_t \doteq r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \quad (15)$$



# 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  $\pi$ , udowodnione zostało, że metody TD(0) są zbieżne do  $v_\pi$ , w przypadku kiedy wartość parametru uczącego ( $\alpha$ ) 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  $\pi^*$  -  
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  $\pi^*$  -  
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  $Q(\text{terminal}, \cdot) = 0$

Pętla po wszystkich epizodach:

Inicjalizacja  $s$

Dla każdego kroku w epizodzie:

Wybierz akcję  $a$  w stanie  $s$  wykorzystując strategię opartą o tablicę  $Q$  (np.,  $\epsilon$ -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  $Q(\text{terminal}, \cdot) = 0$

Loop for each episode:

    Initialize  $s$

    Loop for each step of episode:

        Choose  $a$  from  $s$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -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  $\epsilon$ -zachłanny:

$$a = \begin{cases} \operatorname{argmax}_a Q(s, \cdot) & \text{z prawdopodobieństwem } 1 - \epsilon^* \\ \text{losowa akcja} & \text{z prawdopodobieństwem } \epsilon \end{cases} \quad (16)$$

\* 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:

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

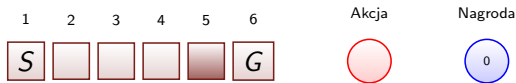
Stan	L	P
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0.1
6	0	0



# Uczenie aktywne

Q-Learning - przykład liczbowy cd.

Epizod 2:



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

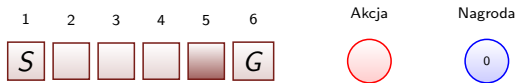
Stan	L	P
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0.1
6	0	0



# Uczenie aktywne

Q-Learning - przykład liczbowy cd.

Epizod 2:



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

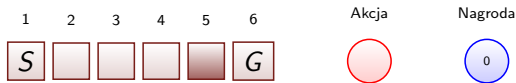
Stan	L	P
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0.1
6	0	0



# Uczenie aktywne

Q-Learning - przykład liczbowy cd.

Epizod 2:



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

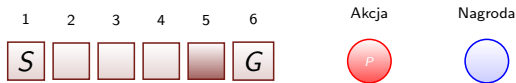
Stan	L	P
1	0	0
2	0	0
3	0	0
4	0	0.009
5	0	0.1
6	0	0



# 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.



Akcja



Nagroda



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  $Q(\text{terminal}, \cdot) = 0$

Pętla po wszystkich epizodach:

Inicjalizacja  $s$

Wybierz akcję  $a$  w stanie  $s$  wykorzystując strategię opartą o tablicę  $Q$  (np.,  $\epsilon$ -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.,  $\epsilon$ -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  $Q(\text{terminal}, \cdot) = 0$

Loop for each episode:

    Initialize  $s$

    Choose  $a$  from  $s$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy)

    Loop for each step of episode:

        Take action  $a$ , observe  $r, s'$

        Choose  $a'$  from  $s'$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -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( $\lambda$ )

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( $\lambda$ )

### Algorytm SARSA( $\lambda$ ) 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  $Q(\text{terminal}, \cdot) = 0$

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.,  $\epsilon$ -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.,

$\epsilon$ -zachłanną)

$$\delta \leftarrow r + \gamma Q(s', a') - Q(s, a)$$

$$E(s, a) = E(s, a) + \delta$$

        Dla każdego  $s \in S$ ,  $a \in A(s)$ :

$$Q(s, a) \leftarrow Q(s, a) + \alpha \delta E(s, a)$$

$$E(s, a) \leftarrow \gamma \lambda E(s, a)$$

$$s \leftarrow s', a \leftarrow a'$$

    Dopóki  $s$  nie jest stanem końcowym

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

# Uczenie aktywne

## SARSA( $\lambda$ )

### SARSA( $\lambda$ ) 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  $Q(\text{terminal}, \cdot) = 0$

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.,  $\epsilon$ -greedy)

Loop for each step of episode:

Take action  $a$ , observe  $r$ ,  $s'$

Choose  $a'$  from  $s'$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy)

$\delta \leftarrow r + \gamma Q(s', a') - Q(s, a)$

$E(s, a) = E(s, a) + \delta$

For all  $s \in S$ ,  $a \in A(s)$ :

$Q(s, a) \leftarrow Q(s, a) + \alpha \delta E(s, a)$

$E(s, a) \leftarrow \gamma \lambda E(s, a)$

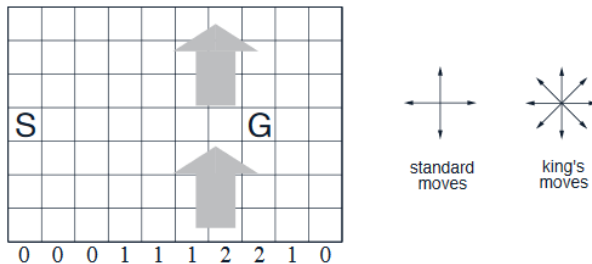
$s \leftarrow s'$ ,  $a \leftarrow a'$

Until  $s$  is not terminal

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

# Uczenie aktywne

## Model środowiska



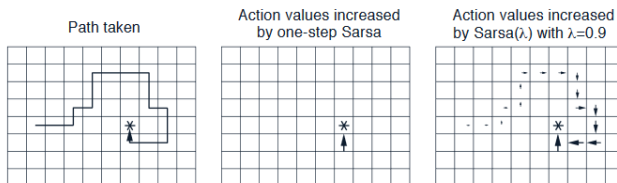
Rysunek 8: Windy Gridworld

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



# Uczenie aktywne

## SARSA( $\lambda$ )



Rysunek 9: Porównanie działania algorytmów SARSA i SARSA( $\lambda$ )

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



# Uczenie aktywne

## SARSA( $\lambda$ )

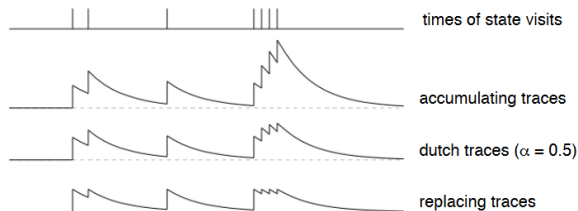
Strategie aktualizacji śladu:

- $E_t(s, a) = \gamma \lambda E_{t-1}(s, a) + 1$  - 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( $\lambda$ )



Rysunek 10: Porównanie strategii aktualizacji śladu

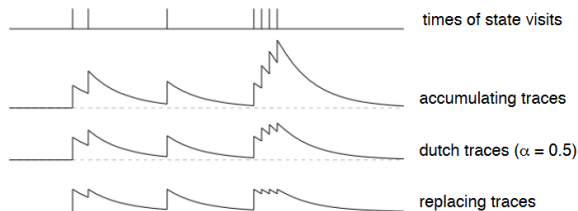
Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2015.





# Uczenie aktywne

## SARSA( $\lambda$ )



Rysunek 11: 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.
- $\mathbb{E}_{s'}(\max_{a'}(Q(s_{t+1}, a')) \geq \max_{a'}(\mathbb{E}_{s'}(Q(s_{t+1}, a')))$
- Problem ten jest nazywany *Maximization Bias*.



# Uczenie aktywne

## *Double Q-Learning*

Rozwiązanie problemu:

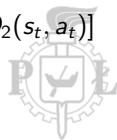
Uczenie oddzielnie dwóch funkcji  $Q$  -  $Q_1$  i  $Q_2$ .

Aktualizacja wartości funkcji  $Q_1$  na podstawie wartości funkcji  $Q_2$ :

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

Aktualizacja wartości funkcji  $Q_2$  na podstawie wartości funkcji  $Q_1$ :

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



# 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  $Q(\text{terminal}, \cdot) = 0$

Pętla po wszystkich epizodach:

    Inicjalizacja  $s$

    Dla każdego kroku w epizodzie:

        Wybierz akcję  $a$  w stanie  $s$  wykorzystując strategię  $\epsilon$ -zachłanną dla  $Q_1 + Q_2$

        Wykonaj akcję  $a$  i zaobserwuj  $r$  oraz  $s'$

        Z prawdopodobieństwem 0.5 aktualizuj:

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

        lub:

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

$s \leftarrow s'$

    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  $Q(\text{terminal}, \cdot) = 0$

Loop for each episode:

    Initialize  $s$

    Loop for each step of episode:

        Choose  $a$  from  $s$  using the policy  $\epsilon$ -greedy in  $Q_1 + Q_2$

        Take action  $a$ , observe  $r$ ,  $s'$

        With 0.5 probability:

$$Q_1(s, a) = Q_1(s, a) + \alpha[r + \gamma Q_2(s', \operatorname{argmax}_a(Q_1(s', a))) - Q_1(s, a)]$$

        else:

$$Q_2(s, a) = Q_2(s, a) + \alpha[r + \gamma Q_1(s', \operatorname{argmax}_a(Q_2(s', a))) - Q_2(s, a)]$$

$s \leftarrow s'$

    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:*

S	F	F	F
F	H	F	H
F	F	F	H
H	F	F	G

### 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:

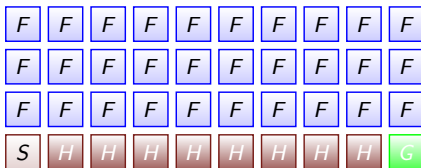
- lewo,
- prawo,
- góra,
- dół.



# 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:

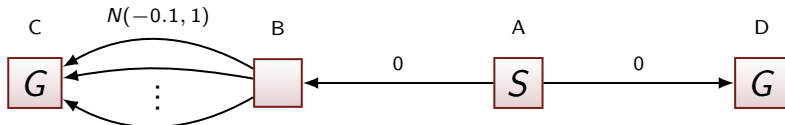
- 1 - po dotarciu do pola  $G$ ,
- $-100$  - po dotarciu do pola  $H$ ,
- $-1$  - w pozostałych przypadkach.



# 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( $\lambda$ )*.
- 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.



## 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:

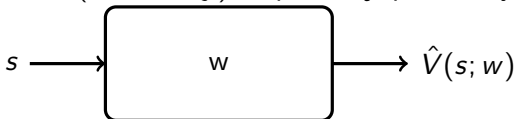
- opracowanie idealnej strategii na podstawie znajomości modelu środowiska (uczenie pasywne),
- opracowanie strategii na podstawie doświadczenia (uczenie aktywne),
- do tej pory wartości opisujące stan lub parę stan-akcja przechowywane były w tablicy, co może okazać się problematyczne:
  - w przypadku 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:

$$Q(s, a) = \sum_{i=1}^n f_i(s, a)w_i \quad (17)$$

Błąd tymczasowy:

$$\delta = (r + \gamma \max_{a'} Q(s', a')) - Q(s, a) \quad (18)$$



# Aproksymacja funkcji wartości stanu

Liniowa kombinacja cech

Aktualizacja wartości wag:

$$w_i = w_i + \alpha \delta f_i(s, a) \quad (19)$$

Minimalizacja błędu:

$$J(w) = \|(r + \gamma \max_{a'} Q(s', a')) - Q(s, a)\|^2 \quad (20)$$



# 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 ...



# Aproksymacja funkcji wartości stanu

## 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*.

