Uczenie ze wzmocnieniem w animacji komputerowej

Henryk Maciejewski

Projekt współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Rozwoju Regionalnego Program Operacyjny Polska Cyfrowa na lata 2014-2020,
Oś Priorytetowa nr 3 "Cyfrowe kompetencje społeczeństwa" Działanie nr 3.2 "Innowacyjne rozwiązania na rzecz aktywizacji cyfrowej"
Tytuł projektu: „Akademia Innowacyjnych Zastosowań Technologii Cyfrowych (AI Tech)”
Porozumienie nr POPC.03.02.00-00-0001/20-00

Plan

- Podstawowe pojęcia RL na przykładach (gry, animacje)

- MDP – Markov Decision Process
- Metody rozwiązywania MDP
- Wybrane algorytmy - idea

Plan

- Podstawowe pojęcia RL na przykładach (gry, animacje)

- MDP – Markov Decision Process
- Metody rozwiązywania MDP
  - Value-based vs policy optimization
  - Model-based vs model-free
  - Online vs offline
  - On-policy vs off-policy
- Wybrane algorytmy - idea

Przykład: Godot RL Agents

- Godot RL Agents -- Ball chase
- https://github.com/edbeeching/godot_rl_agents

- **Obserwacja** - stan środowiska (*Observation*):
  A vector, pointing in the direction of the next pink fruit. A circle of 2D raycasts around the agent.

- **Nagroda** (*Reward*)
  - *Dense* reward based on decrease in euclidean distance to next pink fruit
  - *Sparse* reward of 10.0 when the fruit is collected
  - Penalty of -10.0 when the agent hits a wall

Przykład: Godot RL Agents

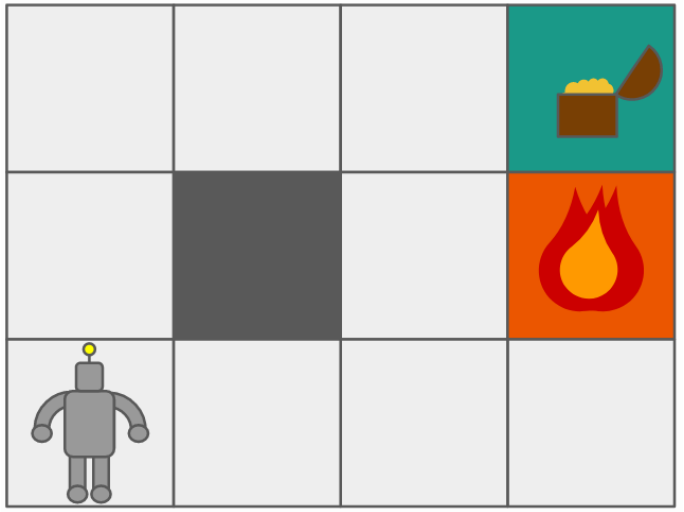
- Godot RL Agents -- Ball chase
- https://github.com/edbeeching/godot_rl_agents

- **Epizod** (*reset condition*)
  - Hitting a wall
  - Reaching the episode limit of 5000 timesteps

- **Akcje** (*Action space*)
  Continuous action space: 2D vector indicating the move direction in x and y

Przykład: GridWorld

- GridWorld – prosty przykład
- https://mohitd.github.io/reinforcement-learning.html



https://mohitd.github.io/reinforcement-learning.html

- **Obserwacja – s, stan gry**
  - Pozycja robota
- **Nagroda**
  +1 – skarb
  -1 – fire pit

- **Akcje - a:** ⇩ ⇦ ⇧ ⇨
- Epizod – ciąg stanów do skarbu albo fire pit-u

Algortym RL uczy się polityki:

$$\pi(a|s) \rightarrow R$$

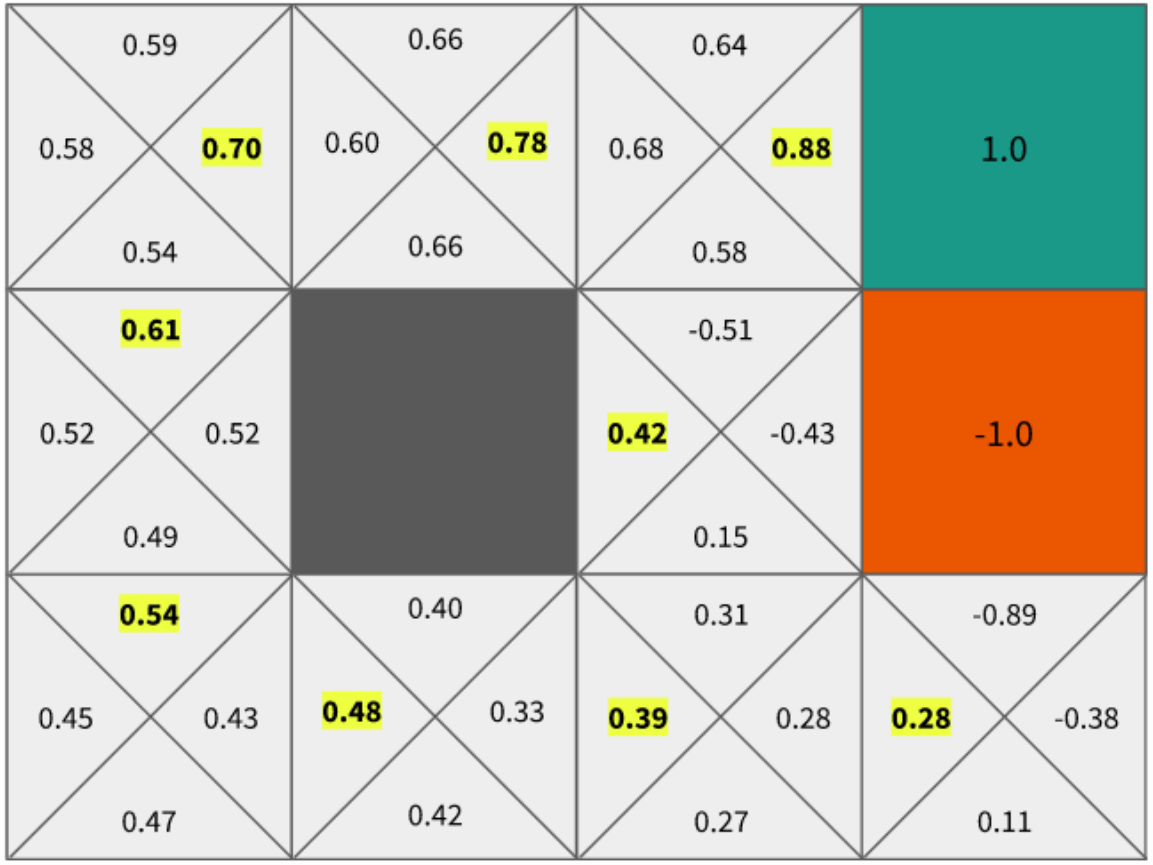
Przykład: GridWorld

- Przykład – algorytm Q-learning

  Q(s,a) – spodziewana nagroda po wykonaniu akcji a w stanie s

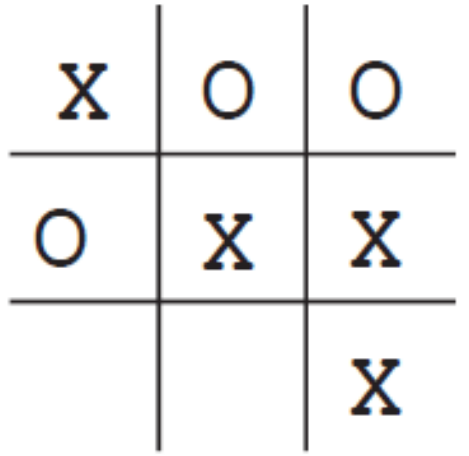
- Wówczas polityka (*greedy*):

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



https://mohitd.github.io/reinforcement-learning.html

Przykład: Kółko i krzyżyk

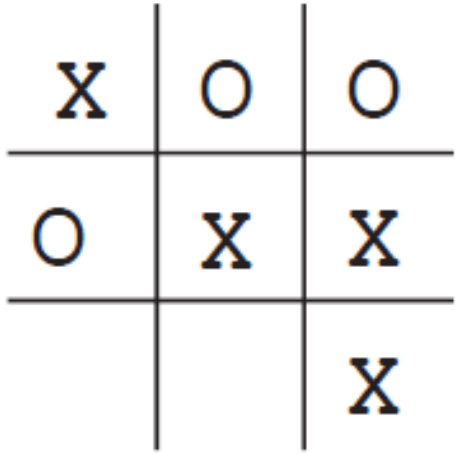


- **Stan** s: dowolny (legalny) układ X i O na planszy

- **Nagroda**:
  +1 – wygrywamy
  0 – remis albo przegrywamy

- **Akcje** – zakładamy, że agent gra X

Podejścia do uczenia agenta



Metody RL

- Np. value-based: uczymy Q(s,a) albo V(s)
  w oparciu o *interakcje ze środowiskiem*: $\mathbf{s_0,a_0,r_1,s_1,a_1,r_2,s_2,\ldots}$
- Greedy policy: $\pi(s) = argmax_a\, Q(s,a)$

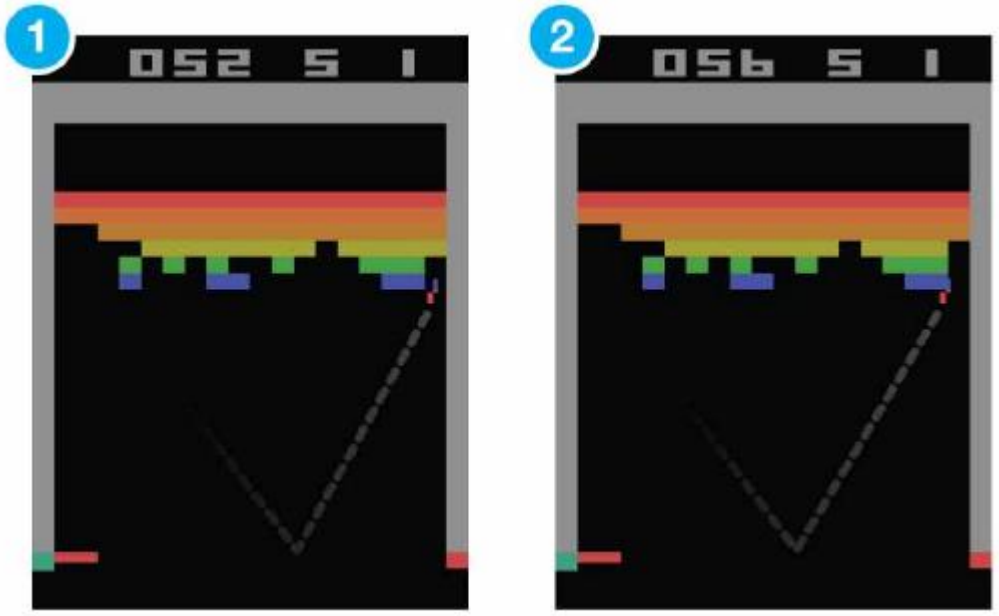
Metody ewolucyjne (nie RL!)

- Zakładamy politykę, oceniamy jej wartość (winning probability) – na podstawie wielu gier
- Algorytm genetyczny – tworzymy lepsze polityki z populacji polityk
- Brak interakcji ze środowiskiem

Przykład: gry Atari, algorytm DQN

- Gry Atari
- https://www.youtube.com/watch?v=V1eYniJ0Rnk

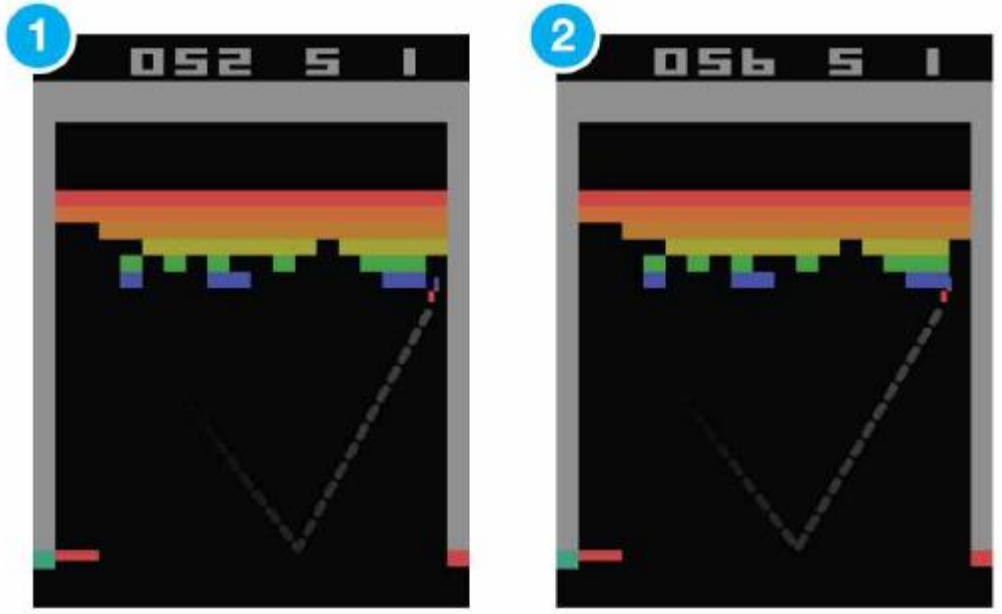
- Mnih, V., et al. "Human-level control through deep reinforcement learning." Nature 2015.
- Algorytm DQN (Deep Q-Network)



Mnih, V., et al. "Human-level control through deep reinforcement learning." Nature 518 no. 7540 (2015): 529-533

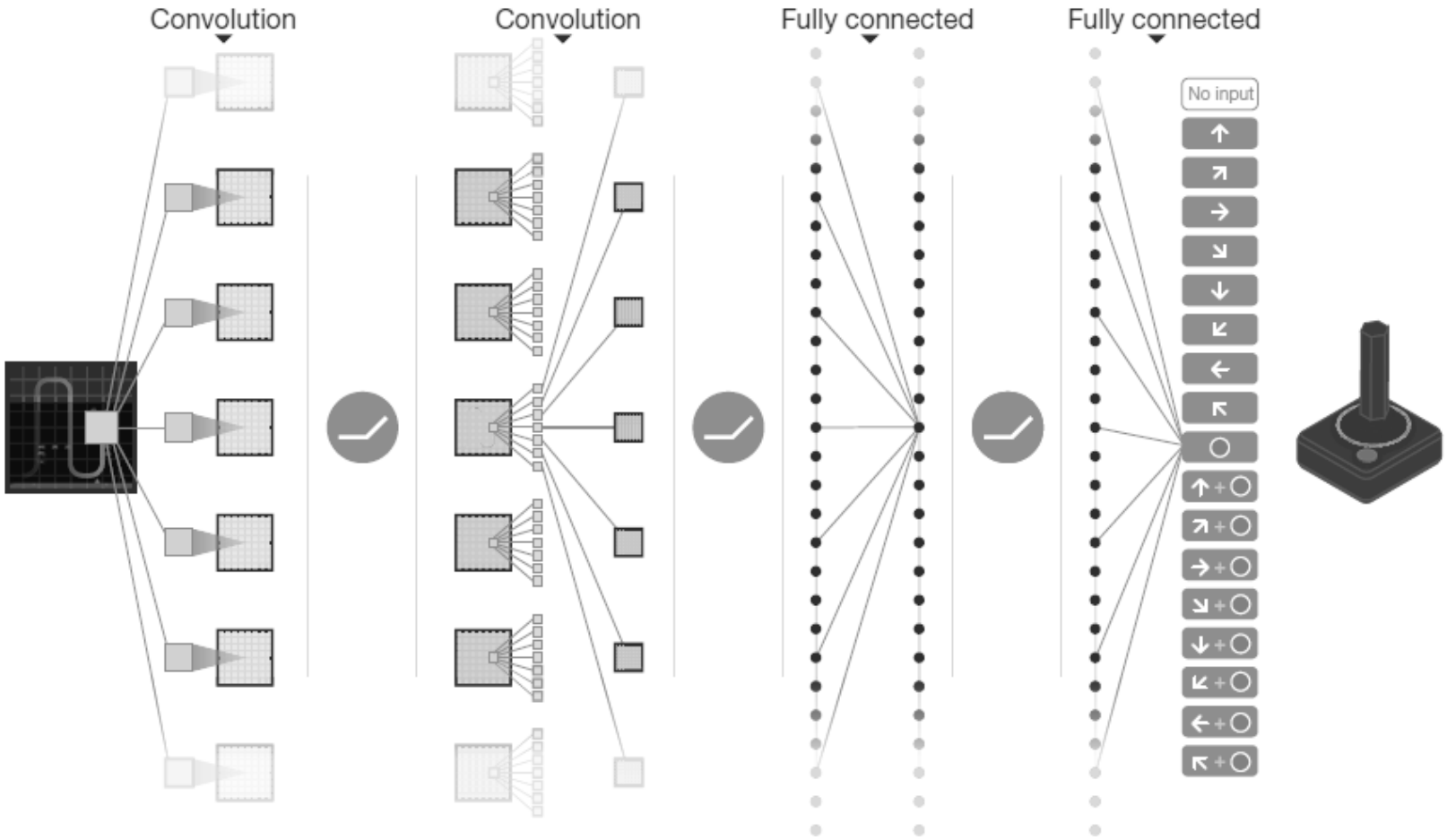
Przykład: gry Atari, algorytm DQN

- **Obserwacja** (s, stan środowiska): obraz 84x84, wejście do funkcji Q(**s**,a) (a dokładniej: seria obrazów – patrz dalej)
- **Akcje**: 4-18 możliwych akcji w różnych grach Atari
- **Nagroda**: r_i (-1,0,+1) – zmiana score gry
- **Epizod**: koniec epizodu sygnalizowany przed emulator gry



Mnih, V., et al. "Human-level control through deep reinforcement learning." Nature 518 no. 7540 (2015): 529-533

- Model funkcji Q – sieć głęboka (wagi θ)
  - we: s (obraz)
  - wy: wartość Q(s,a) dla każdej akcji a ϵ A
    (4 do 18 akcji w różnych grach Atari)



Mnih, V., et al. "Human-level control through deep reinforcement learning."
Nature 518 no. 7540 (2015): 529-533

- Cel uczenia: uzyskanie aproksymacji optymalnej funkcji Q(s,a) – action-value:

$$Q^*(s,a) = \max_{\pi} \mathbb{E}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots \,|\, s_t = s,\ a_t = a,\ \pi\right]$$

- Uczenie funkcji Q – minimalizujemy funkcję straty:

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

- Uzasadnienie: optymalna funkcja Q spełnia **równanie Bellmana**:

$$Q^{\pi}(s,a) = E_{\pi}[r + \gamma\, max_{a'} Q(s',a')]$$

- Korzystamy z opisu gry przy pomocy **MDP** (Markov Decision Process)
- MDP – proces bez pamięci (**własność Markowowska** procesu stoch.)
- Pojedynczy obraz jako stan nie spełnia własności Markowowskiej (dlaczego?)
- Zdefiniujmy stan s_t jako sekwencję obrazów:

$$s_t = x_1, a_1, x_2, \ldots, x_t$$

Algorytm (Deep Q-Learning)

Algorithm 1: deep Q-learning with experience replay.

Initialize replay memory D to capacity N

Initialize action-value function Q with random weights θ

Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$

For episode $= 1, M$ **do**

  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$

  For $t = 1$,T **do**

    With probability ε select a random action a_t

    otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$

    Execute action a_t in emulator and observe reward r_t and image x_{t+1}

    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$

    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D

    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D

    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^{-}) & \text{otherwise} \end{cases}$

    Perform a gradient descent step on $\left(y_j - Q(\phi_j, a_j; \theta)\right)^2$ with respect to the network parameters θ

    Every C steps reset $\hat{Q} = Q$

  End For

End For

Mnih, V., et al. "Human-level control through deep reinforcement learning." Nature 518 no. 7540 (2015): 529-533

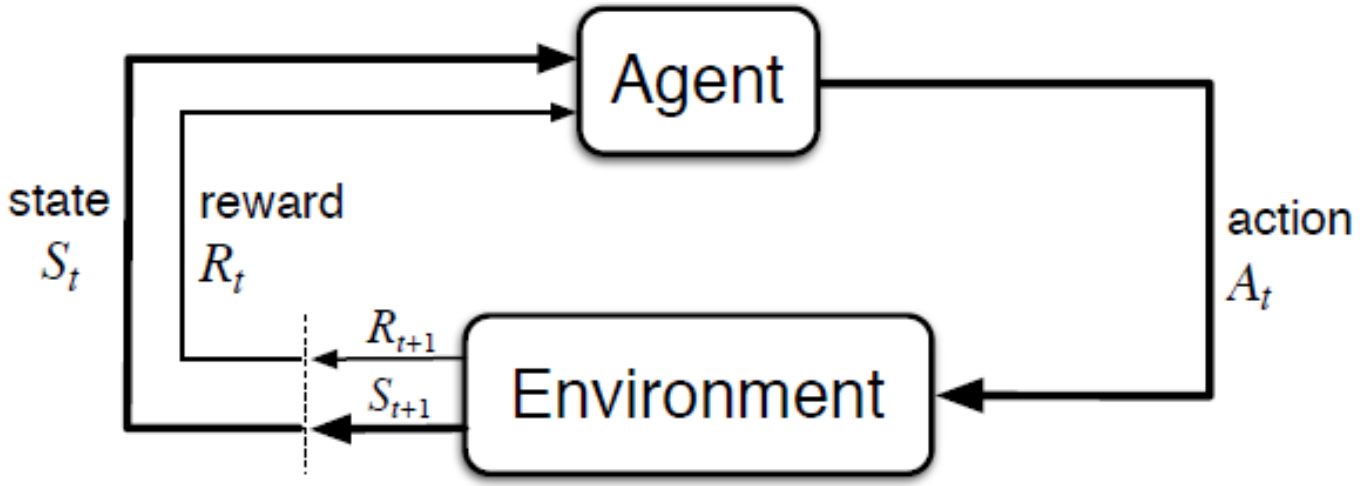
Plan

- Podstawowe pojęcia RL na przykładach (gry, animacje)

- MDP – Markov Decision Process
- Metody rozwiązywania MDP
  - Value-based vs policy optimization
  - Model-based vs model-free
  - Online vs offline
  - On-policy vs off-policy
- Wybrane algorytmy - idea

Markov Decision Process MDP

- Model procesu uczenia ze wzmocnieniem (RL)



Sutton, R., Barto A. Reinforcement learning: An introduction. MIT press, 2018

- Uczenie agenta w oparciu o sekwencje $S_0,A_0,R_1,S_1,A_1,R_2,S_2,\ldots$

MDP – podstawowe pojęcia

- S – przestrzeń stanów
- A – przestrzeń akcji

- $\pi : S \rightarrow A$ – polityka (deterministyczna, stochastyczna)
- $R : S \times A \times S \rightarrow \mathbb{R}$ – nagroda (Reward)

Opcjonalnie:

- $T : S \times A \times S \rightarrow \mathbb{R}$ – dynamika (model) środowiska

Metody RL model-based vs model-free

- Model-based – dynamika środowiska jest znana, możliwe planowanie
- Model-free – model środowiska nieznany, uczenie typu trial-and-error

Możliwe uczenie się modelu – np. algorytm MCTS (Monte Carlo Tree Search)

Funkcja state-value V(s), state-action-value Q(s,a)

- Oczekiwana nagroda
  - Niedyskontowana $G_t = R_{t+1} + R_{t+2} + \cdots + R_T$
  - Dyskontowana $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+2} + \cdots$

- Metody **value-based:**
  - wyznaczamy funkcje **V(s) – state-value** albo **Q(s,a) – state-action value**
  - wówczas polityka: $\pi(s) = \text{argmax}_a\, Q(s,a)$

- $V_\pi(s) = E_\pi[G_t|S_t = s]$ - funkcja state-value dla polityki π
- $Q_\pi(s,a) = E_\pi[G_t|S_t = s, A_t = a]$ - state-action-value dla polityki π

Równanie Bellmana dla V(s) i Q(s,a)

- W MDP zachodzą **równania Bellmana**:

$$V_\pi(s) = E_\pi[R_{t+1} + \gamma\, V_\pi(S_{t+1}) | S_t = s] \qquad \text{dla każdego } s \epsilon S$$

$$Q_\pi(s,a) = E_\pi[R_{t+1} + \max_{a'} Q_\pi\,(S_{t+1}, a') | S_t = s, A_t = a]$$

- **Wyznaczenie V lub Q spełniające te równania prowadzi do optymalnej polityki π.**

Jak rozwiązać równanie Bellmana

- Rozwiązania dokładne
  - Rozwiązujemy n=|S| równań
  - Tylko model-based, b. zasobochłonne…
  - W praktyce – szukamy rozwiązań przybliżonych (zwykle)

- Rozwiązania przybliżone (dla uczenia model-free)
  - **Monte-Carlo**
  - **Temporal-difference**

On-policy Monte-Carlo

- Value-based
- On-policy
- Offline – modyfikacja polityki po zakończonym epizodzie (online – w trakcie)

- Polityka ε-greedy:
  - z prawdopodobieństwem ε wykonujemy **eksplorację**
  - z prawdopodobieństwem 1- ε **exploitation** – akcje *greedy*

$s_0, a_0, r_1, s_1, a_1, r_2, s_2, \ldots, r_T, s_T$

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
$Q(s,a) \leftarrow$ arbitrary
$Returns(s,a) \leftarrow$ empty list
$\pi(a|s) \leftarrow$ an arbitrary ε-soft policy

Repeat forever:
(a) Generate an episode using π
(b) For each pair s, a appearing in the episode:
$G \leftarrow$ return following the first occurrence of s, a
Append G to $Returns(s,a)$
$Q(s,a) \leftarrow \text{average}(Returns(s,a))$
(c) For each s in the episode:
$a^* \leftarrow \arg\max_a Q(s,a)$
For all $a \in \mathcal{A}(s)$:

$$\pi(a|s) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(s)| & \text{if } a = a^* \\ \varepsilon/|\mathcal{A}(s)| & \text{if } a \neq a^* \end{cases}$$

Sutton, R., Barto A. Reinforcement learning: An introduction. MIT press, 2018

Algorytm Q-learning

- Value-based
- Off-policy
- Online – modyfikacja polityki w trakcie epizodu
- Algorytm typu **temporal-difference**

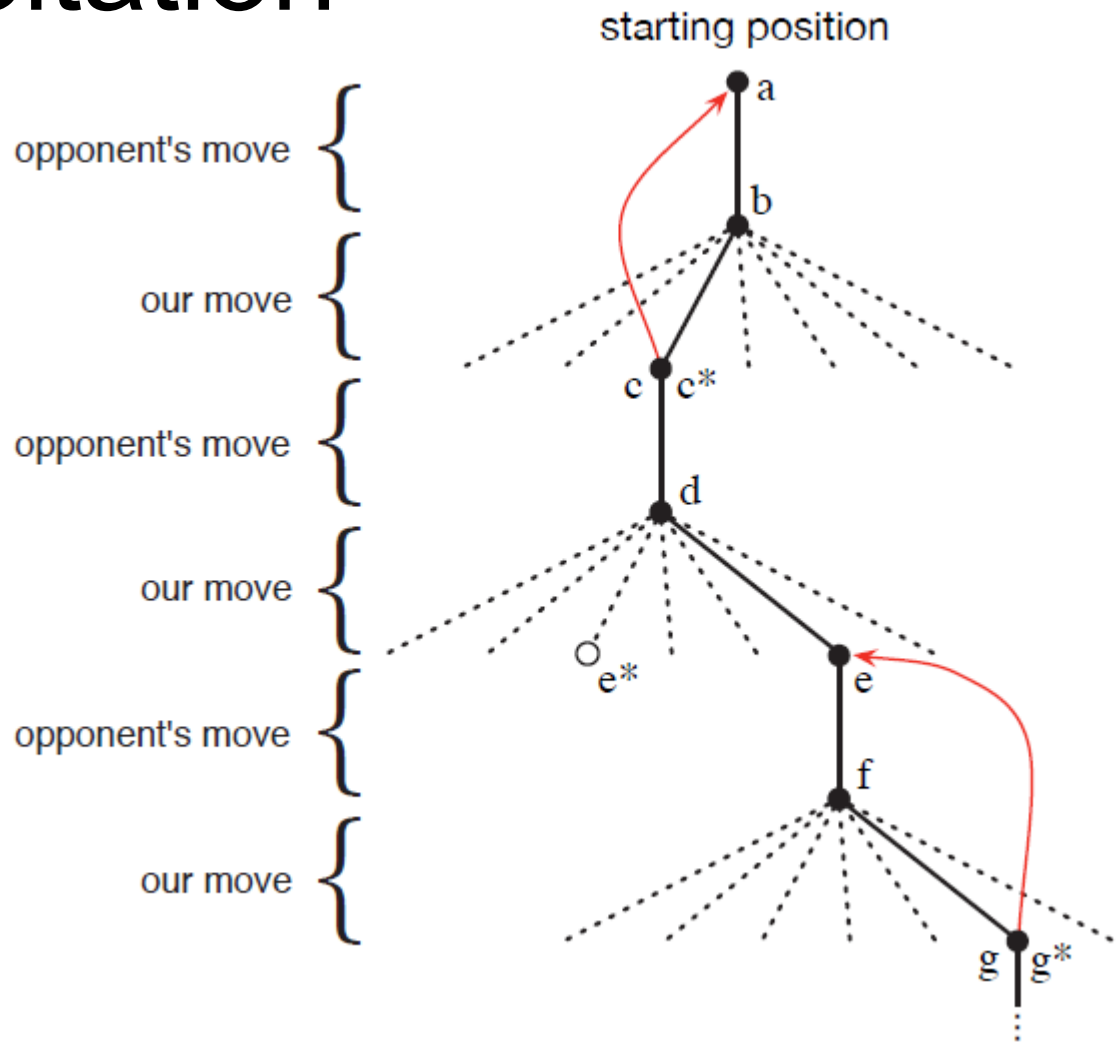
Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose A from S using policy derived from Q (e.g., ϵ-greedy)
    Take action A, observe R, S'
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
    $S \leftarrow S'$;
  until S is terminal

Sutton, R., Barto A. Reinforcement learning: An introduction. MIT press, 2018

TD - Exploration vs exploitation

- **Temporal-difference TD** learning
  - Akcje c*, g* - greedy (exploitation) – wtedy modyfikujemy $V(S_t)$ stanu poprzedniego na podst. $V(S_{t+1})$
  - Akcja e – nieoptymalna, (exploration) – nie ma uczenia TD

$$V(S_t) \leftarrow V(S_t) + \alpha\left[V(S_{t+1}) - V(S_t)\right]$$



Sutton, R., Barto A. Reinforcement learning: An introduction. MIT press, 2018

Metody Policy-based (REINFORCE, PPO)

- Nie uczymy funkcji V(s), Q(s,a)
- Zamiast tego uczymy parametryzacji polityki $\pi(a|s;\theta)$
- Np. algorytm REINFORCE

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

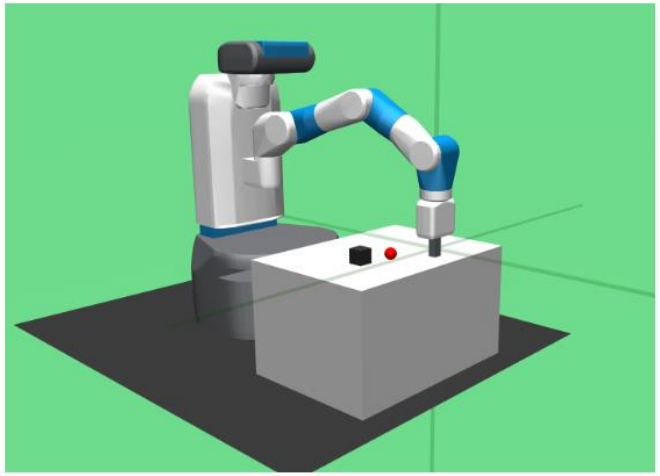
Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Algorithm parameter: step size $\alpha > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):
  Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot,\boldsymbol{\theta})$
  Loop for each step of the episode $t = 0, 1, \ldots, T-1$:
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ (G_t)
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G \nabla \ln \pi(A_t|S_t,\boldsymbol{\theta})$

Sutton, R., Barto A. Reinforcement learning: An introduction. MIT press, 2018

Inne algorytmy, problemy,…

- Algorytmy Actor-Critic (actor=model policy, critic=model value function)
  - Np. A2C, A3C, DDPG, …
- Problem sparse rewards – algorytm HER (rzadka nagroda, na końcu epizodu zakończonego sukcesem)
- Learning from demonstrations, imitation learning, …
- Curiosity learning



https://openai.com/blog/ingredients-for-robotics-research/

- Long episodes
- Offline RL - uczenie z ograniczoną eksploracją, na podst. statycznych zbiorów epizodów

Literatura

1. R.S. Sutton and A.G. Barto, Reinforcement Learning: An Introduction, second edition. MIT Press 2020.
2. Mnih, V., et al. Human-level control through deep reinforcement learning. Nature 2015.
3. Kiran, B. Ravi, et al. Deep reinforcement learning for autonomous driving: A survey. IEEE Transactions on Intelligent Transportation Systems (2021).