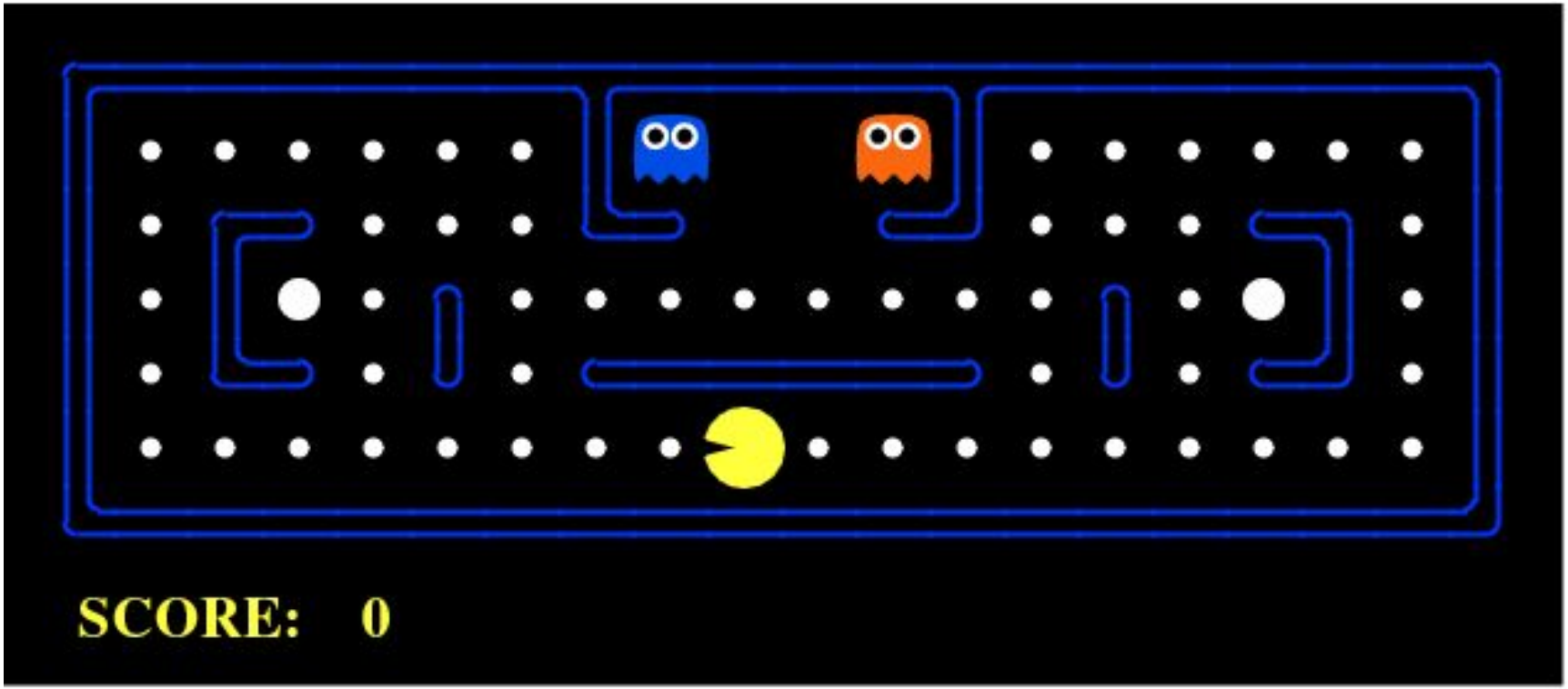
Reinforcement Learning для самых маленьких

Валерия Бубнова
Андрей Харатян

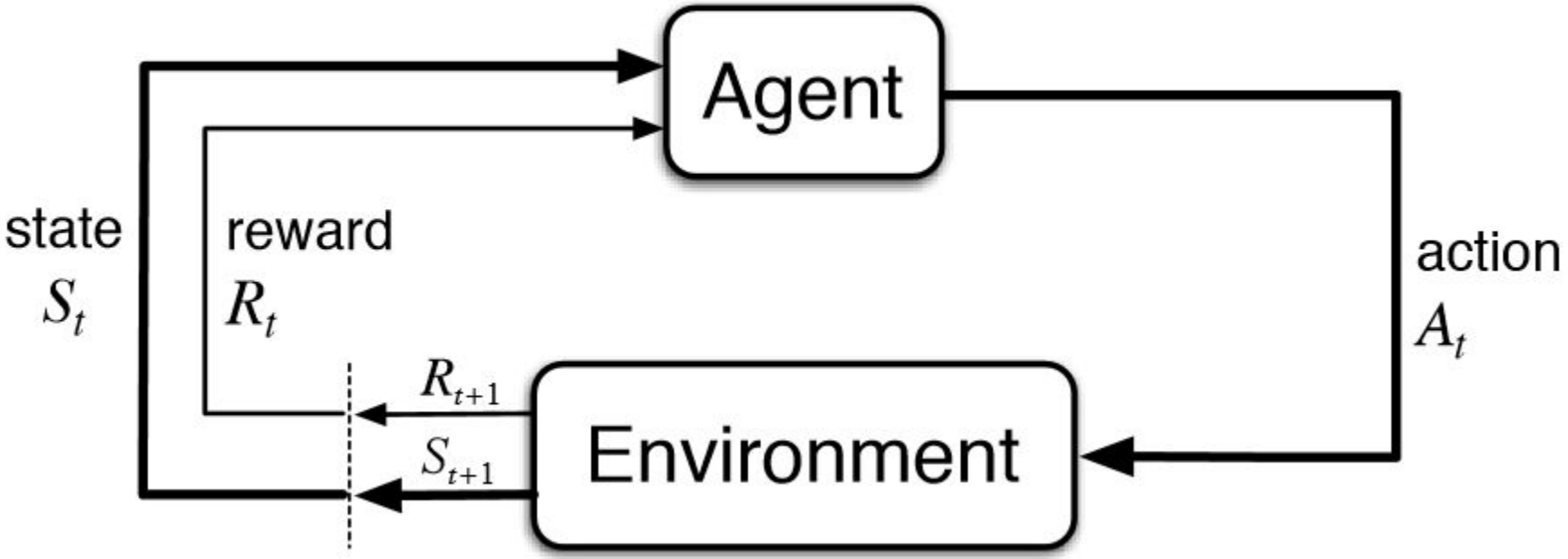
Мы расскажем:

1. Что такое RL и зачем это всё нужно
2. Что такое Markov Decision Process
3. Про задачу exploration/exploitation и многоруких бандитов
4. Про алгоритмы Q-learning и SARSA
5. Про любопытные примеры Deep Reinforcement Learning

Как это выглядит?



Кто мы? Чего мы хотим?



Что у нас есть?

Множество состояний среды: S

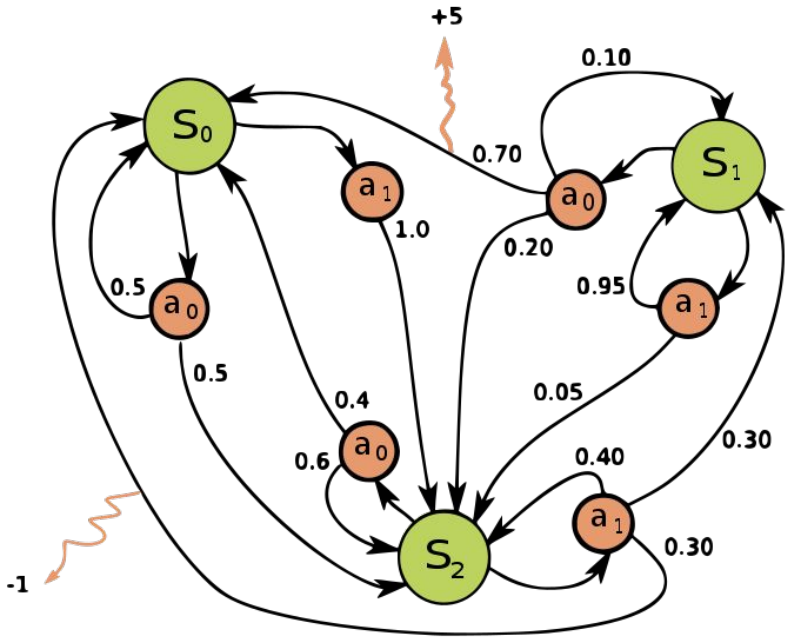
Множество возможных действий: A

Множество наград: R

Markov Decision Process (MDP)

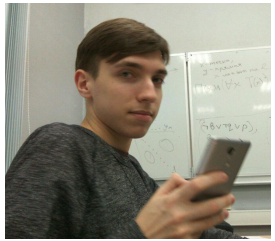
Состояние S_t называется Марковым, если и только если

$$P[S_{t+1}|S_t] = P[S_{t+1}|S_1, ..., S_t]$$



Состояние это достаточная статистика от будущего...

Markov Decision Process (MDP)



S0
Deadline
soon

Look
for
memes

-50

S2
No more
study

Do
nothing

-1

S2
No more
study

Study
not
hard

+5

S1
Holidays

Look
for
memes

+1

S0
Deadline
soon

Study
hard

+10

S1
Holidays

Look
for
memes

+1

S0
Deadline
soon

Markov Decision Process (MDP)

Стратегия π – это условное распределение действий при некотором состоянии среды.

$$\pi(a|s) = P[A_t = a|S_t = s]$$

Отклик G_t – это затухающая суммарная награда за все действия после момента t.

$$G_t = R_{t+1} + \gamma R_{t+2} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Markov Decision Process (MDP)

P – это матрица вероятностей переходов между состояниями:

$$P_{s,s'} = P[S_{t+1} = s | S_t = s]$$

Функция ценности состояния (the state-value function) $v_\pi(s)$ это математическое ожидание дальнейшего отклика при данном состоянии s, при условии следования стратегии π

$$v_\pi(s) = E_\pi[G_t | S_t = s]$$

The action-value function $Q_\pi(s, a)$ – это мат. ожидание отклика, начиная с состояния s при следовании стратегии π:

$$Q_\pi(s, a) = E_\pi[G_t | S_t = s, A_t = a]$$

Brief-summary

1. Model-based

$$(s, a, r) \rightarrow [model - learner] \rightarrow P/R \rightarrow [MDP - solver] \rightarrow$$
$$\rightarrow Q \rightarrow [argmax] \rightarrow \pi$$

2. Model-free, value-based

$$(s, a, r) \rightarrow [value\ updates] \rightarrow Q \rightarrow [argmax] \rightarrow \pi$$

3. Model-free, policy-based

$$(s, a, r) \rightarrow [policy\ updates] \rightarrow \pi$$

Exploration / exploitation

Exploration/exploitation

- Greedy
- ϵ-greedy
- Softmax selection

$$P_s(a) = \frac{e^{Q(s,a)/T}}{\sum_{b \in A} e^{Q(s,b)/T}},\ T = const$$

- UCB

$$Q'(s,a) = Q(s,a) + \alpha\sqrt{\frac{2logN_s}{n_{s,a}}}$$

N_s раз были в состоянии s

$n_{s,a}$ выбирали действие a

Temporal Difference

Обновляем state-value function:

$$v(s_{t-1}) \leftarrow v(s_{t-1}) + \alpha_t(R_t + \gamma v(s_t) - v(s_{t-1}))$$

$$v(s_{t-1}) \leftarrow E_{s_t}[R_t + \gamma v(s_t)]$$

Temporal Difference

TD(1):

$e(s) = 0 \ \forall s$

while learning:

Шаг: $s_t \rightarrow s_{t+1}(R_t)$

$e(s_t) += 1$

$\forall s_i$:

$$v(s_i) \leftarrow v(s_i) + \alpha(R_t + \gamma v(s_t) - v(s_{t-1}))e(s_i)$$

$$e(s_i) = \gamma e(s_i)$$

Q-Learning

Как это работает:

Q-function:

$$Q(S_t, A_t) = \max \sum_{i=t}^{\inf} \gamma^{i-t} R_i$$

$$\pi(S) = argmax_A Q(S, A)$$

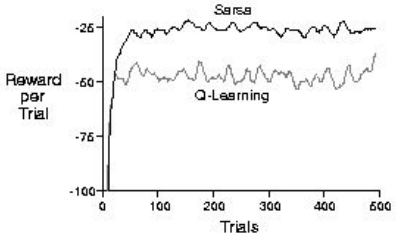
$$Q(S_t, A_t) = R_t + \gamma \max_A Q(S', A)$$

```
Initialize Q(s, a) arbitrarily
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) <-- Q(s, a) + α [r + γ max_α,Q(s', a') - Q(s, a)]
      s <-- s';
   until s is terminal
```

SARSA

```
Initialize Q(s, a) arbitrarily
Repeat (for each episode):
   Initialize s
   Choose a from s using policy derived from Q
         (e.g., ε-greedy)
   Repeat (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) <-- Q(s, a) + α[r + γQ(s', a')- Q(s, a)]
      s <-- s'; a <-- a';
   until s is terminal
```

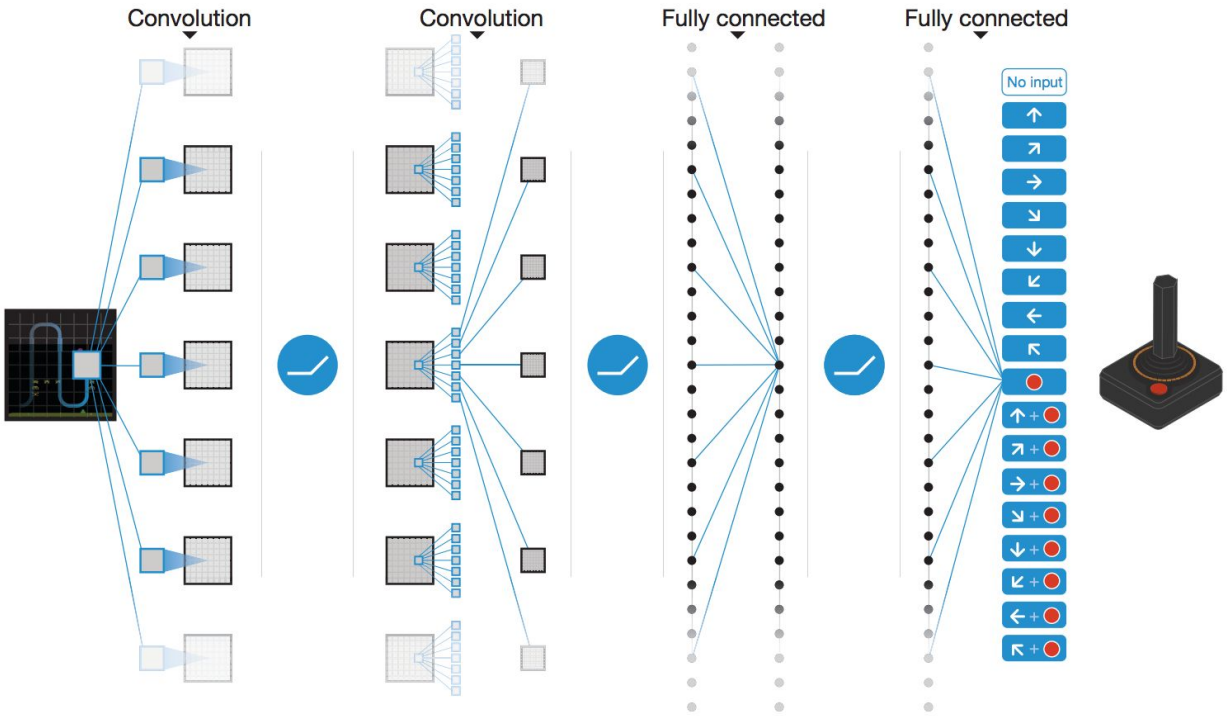
Cliff World



Q-Learning

SARSA

DQN

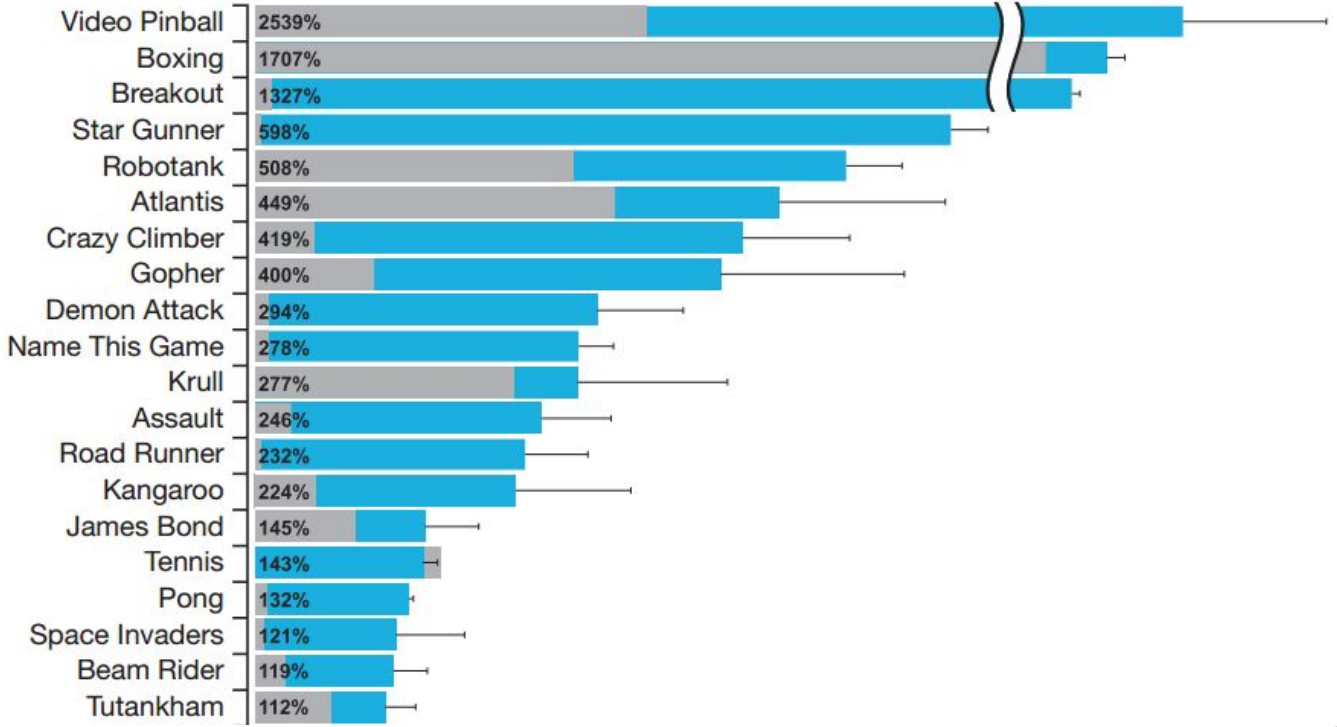

Convolution
Convolution
Fully connected
Fully connected
No input

DQN

Given a transition $< s, a, r, s' >$, the Q-table update rule in the previous algorithm must be replaced with the following:

1. Do a feedforward pass for the current state s to get predicted Q-values for all actions.
2. Do a feedforward pass for the next state s' and calculate maximum overall network outputs $max_{a'} Q(s', a')$.
3. Set Q-value target for action to $r + \gamma max_{a'} Q(s', a')$ (use the max calculated in step 2). For all other actions, set the Q-value target to the same as originally returned from step 1, making the error 0 for those outputs.
4. Update the weights using backpropagation.

DQN


Video Pinball
Boxing
Breakout
Star Gunner
Robotank
Atlantis
Crazy Climber
Gopher
Demon Attack
Name This Game
Krull
Assault
Road Runner
Kangaroo
James Bond
Tennis
Pong
Space Invaders
Beam Rider
Tutankham
2539%
1707%
1327%
598%
508%
449%
419%
400%
294%
278%
277%
246%
232%
224%
145%
143%
132%
121%
119%
112%


and gaps.

Использованная литература

- University College London Lectures: http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/MDP.pdf
- University of Michigan Lectures: http://hunch.net/~jl/projects/RL/RLTheoryTutorial.pdf
- http://www.cse.unsw.edu.au/~cs9417ml/RL1/tdlearning.html
- Missouri University of Science and Technology: http://web.mst.edu/~gosavia/tutorial.pdf
- MIT 6.S191 Lecture 6: Deep Reinforcement Learning: https://www.youtube.com/watch?v=xWe58WGWmlk&t=2308s