

# DEEP LEARNING

## Введение в обучение с подкреплением

---

Святослав Елизаров, Коваленко Борис, Артем Грачев

20 января 2018

Высшая школа экономики

- **Supervised Learning** – we have inputs, we have outputs. We just need to extract a law how to get outputs from inputs. *"I know how to classify this data, I just need you (the classifier, the machine) to do it automatically."*
- **Unsupervised learning** – we have inputs, but we have not outputs. We want to structure the data. To make good data representation, clustering. *"I have no idea how to classify this data, can you (the algorithm) create a classifier for me?"*
- **Reinforcement learning** – we have inputs, we still have not outputs. But we know what good for us what not (reward structure). *"I have no idea how to classify this data, can you classify this data and I'll give you a reward if it's correct or I'll punish you if it's not."*

# REINFORCEMENT LEARNING

---

# MARKOV DECISION PROCESS

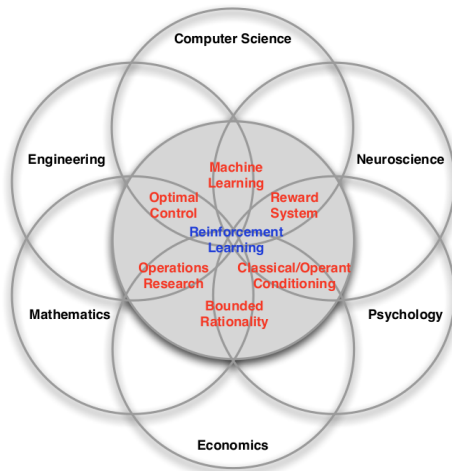


Рис. 1: Many faces of RL

# MARKOV DECISION PROCESS

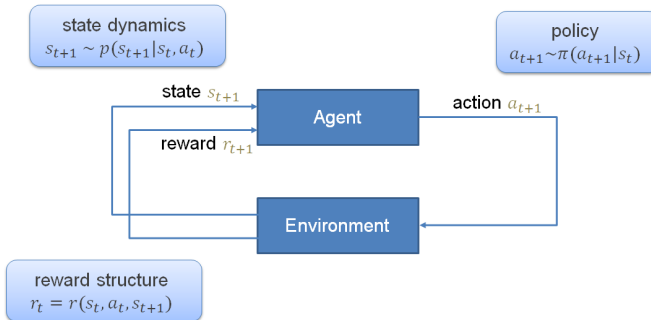


Рис. 2: Динамика системы

$$(S, A, R, P, \gamma)$$

S - Множество возможных состояний

A - Множество возможных действий

R - Распределение наград на парах  $(s_t, a_t)$

P - Распределение переходов между парами  $(s_t, a_t)$

$\gamma$  - коэффициент затухания

В  $t_0$  среда выбирает начальное состояние  $s_0 \sim p(s_0)$

Цикл от  $t_0$  и далее

- Агент выбирает действие  $a_t$
- Среда выбирает награду  $r_t \sim R(.|s_t, a_t)$
- Среда выбирает следующее состояние  $s_{t+1} \sim P(.|s_t, a_t)$
- Агент получает награду и следующее состояние среды

В  $t_0$  среда выбирает начальное состояние  $s_0 \sim p(s_0)$

Цикл от  $t_0$  и далее

- Агент выбирает действие  $a_t$
- Среда выбирает награду  $r_t \sim R(.|s_t, a_t)$
- Среда выбирает следующее состояние  $s_{t+1} \sim P(.|s_t, a_t)$
- Агент получает награду и следующее состояние среды

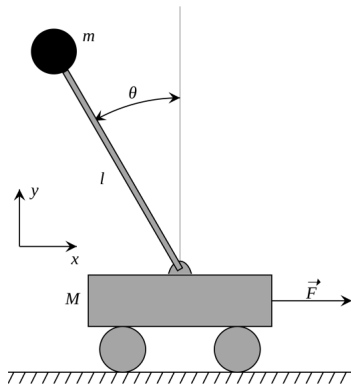
Policy  $\pi$  - функция  $S \rightarrow A$ ,

## Задача

Найти такую  $\pi$ , которая максимизирует  $\sum_{t>0} \gamma^t r_t$



## Управление обратным маятником



Состояние - ? Действие - ? Награда - ?

## Управление движением робота



Состояние - ? Действие - ? Награда - ?

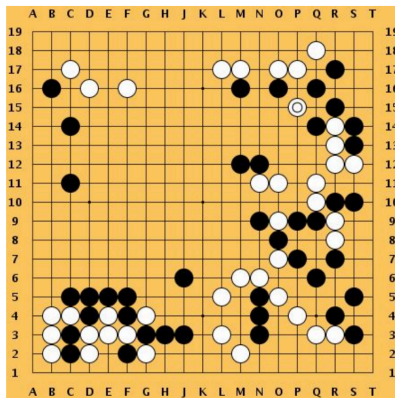
## Компьютерные игры



Состояние - ? Действие - ? Награда - ?

# ПРИМЕРЫ ЗАДАЧ

Go




Состояние - ? Действие - ? Награда - ?


- Траектория:  $s_0, a_0, s_1, r_0, a_1, s_2, r_1, \dots$ 
  - На стороне среды:  $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$
  - На стороне среды:  $r_{t+1} = r(s_t, a_t, s_{t+1})$
  - На стороне агента:  $a_t \sim \pi(a_t|s_t)$
- Ожидаемая награда:  $R(t) = \sum_{\tau=t}^{\infty} \gamma^{\tau-t} r_{\tau}$
- State-value функция:  $V^{\pi}(s) = \mathbb{E}_{\pi} [R(t) | s_t = s]$
- Action-value функция:  $Q^{\pi}(s, a) = \mathbb{E}_{\pi} \{R(t) | s_t = s, a_t = a\}$


# MARKOV DECISION PROCESS


Пример MDP

actions = {

1. right 

2. left 

3. up 

4. down 

}

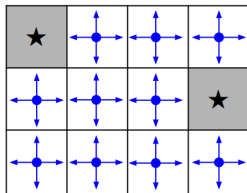
states

★			
			★

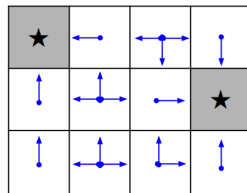
Начинаем в случайном месте, цель - попасть в квадратики с звездами за наименьшее количество шагов

# MARKOV DECISION PROCESS

Пример MDP



Random Policy



Optimal Policy

[Demo by Andrej Karpathy](#)



Табличный алгоритм. Все q-values  $Q(s, a)$  хранятся в памяти.

## Алгоритм

**Init:**  $\forall s, a : Q(s, a) = 0$

**Loop:**

1. Сэмплируем  $\langle s_t, a_t, r_t, s_{t+1} \rangle$  из среды
2. Вычисляем  $\hat{Q}(s_t, a_t) = r_t + \gamma \max_{a_i} Q_{prev}(s_{t+1}, a_i)$
3. Обновляем  $Q_{new}(s_t, a_t) = \alpha \hat{Q}(s_t, a_t) + (1 - \alpha) Q_{prev}(s, a)$

**До сходимости**

Табличный алгоритм. Все q-values  $Q(s, a)$  хранятся в памяти.

## Алгоритм

**Init:**  $\forall s, a : Q(s, a) = 0$

**Loop:**

1. Сэмплируем  $\langle s_t, a_t, r_t, s_{t+1} \rangle$  из среды
2. Вычисляем  $\hat{Q}(s_t, a_t) = r_t + \gamma \max_{a_i} Q_{prev}(s_{t+1}, a_i)$
3. Обновляем  $Q_{new}(s_t, a_t) = \alpha \hat{Q}(s_t, a_t) + (1 - \alpha) Q_{prev}(s, a)$

**До сходимости**

## Проблема

Что делать с многомерными и (или) непрерывными состояниями?

Давайте попробуем аппроксимировать  $Q(s, a) \approx Q(s, a|\theta)$ , где  $\theta$  – некоторые параметры

$$\arg \min_{\theta} \left( Q(s_t, a_t) - [r_t + \gamma \max_{a_i} Q(s_{t+1}, a_i)] \right)^2$$

Давайте попробуем аппроксимировать  $Q(s, a) \approx Q(s, a|\theta)$ , где  $\theta$  – некоторые параметры

$$\arg \min_{\theta} \left( Q(s_t, a_t) - [r_t + \gamma \max_{a_i} Q(s_{t+1}, a_i)] \right)^2$$

## Много различных аппроксиматоров

- Linear combinations of features
- Decision tree
- Nearest neighbour
- Fourier / wavelet bases
- ...

# Q-LEARNING АППРОКСИМАЦИЯ

Давайте попробуем аппроксимировать  $Q(s, a) \approx Q(s, a|\theta)$ , где  $\theta$  – некоторые параметры

$$\arg \min_{\theta} \left( Q(s_t, a_t) - [r_t + \gamma \max_{a_i} Q(s_{t+1}, a_i)] \right)^2$$

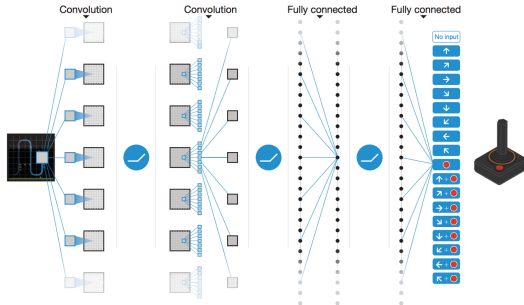
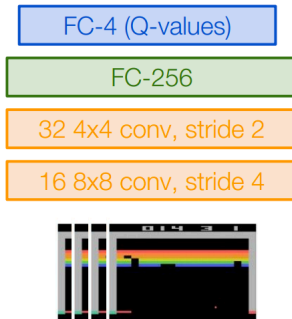


Рис. 3: DQN



Состояние – последние 4 фрейма  
Последний слой имеет 4 выхода (действия)

---

**Algorithm 1** Deep Q-learning with Experience Replay

---

```
Initialize replay memory  $\mathcal{D}$  to capacity  $N$ 
Initialize action-value function  $Q$  with random weights
for episode = 1,  $M$  do
  Initialise sequence  $s_1 = \{x_1\}$  and preprocessed sequenced  $\phi_1 = \phi(s_1)$ 
  for  $t = 1, T$  do
    With probability  $\epsilon$  select a random action  $a_t$ 
    otherwise select  $a_t = \max_a Q^*(\phi(s_t), a; \theta)$ 
    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  $\mathcal{D}$ 
    Sample random minibatch of transitions  $(\phi_j, a_j, r_j, \phi_{j+1})$  from  $\mathcal{D}$ 
    Set  $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$ 
    Perform a gradient descent step on  $(y_j - Q(\phi_j, a_j; \theta))^2$  according to equation 3
  end for
end for
```

---

Состояние – последние 4 фрейма

Последний слой имеет 4 выхода (действия)

Demo by DeepMind



# EXPLORATION VS EXPLOITATION PROBLEM

- **Exploration** исследование среды, получение новой информации о среде
- **Exploitation** взаимодействие со средой оптимальным образом с учетом известной информации
- Обычно важно соблюдать некий баланс между исследованием и оптимальным поведением

Самая простая опция — использовать случайный exploration.

Другие интересные вещи, про которые стоит прочитать в первую очередь:

**Ключевые слова:**

- Double DQN, Experience replay
- Policy gradients methods, TPRO
- A3C — Asynchronous Advantage Actor-Critic Agents
- RL and POMDP (Partially observable Markov decision process)