



ЦЕНТР МОЛОДЁЖНОЙ  
РОБОТОТЕХНИКИ  
МГТУ ИМ. Н.Э. БАУМАНА

# Основы reinforcement learning. RL-алгоритмы.

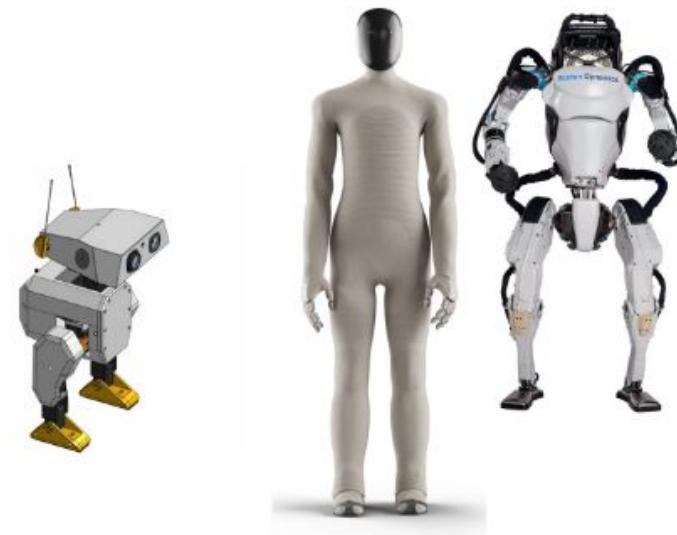
(1) Manipulation



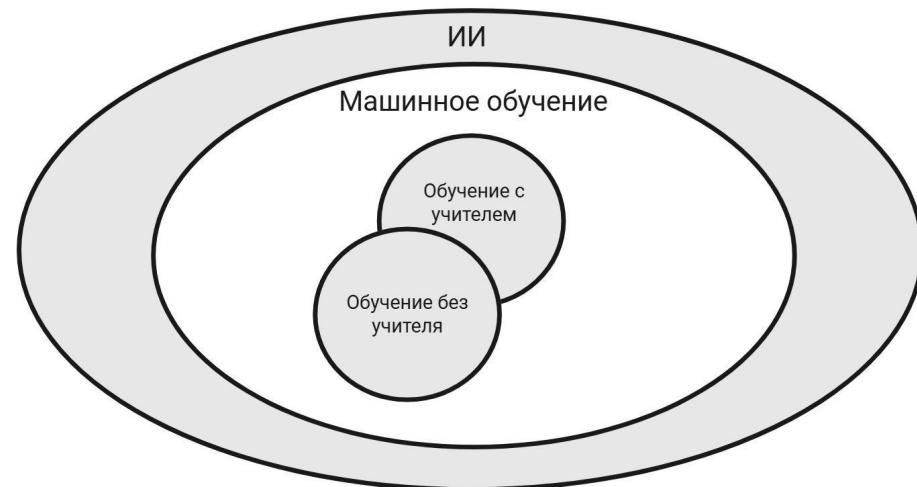
(2) Locomotion



(3) Whole-body control



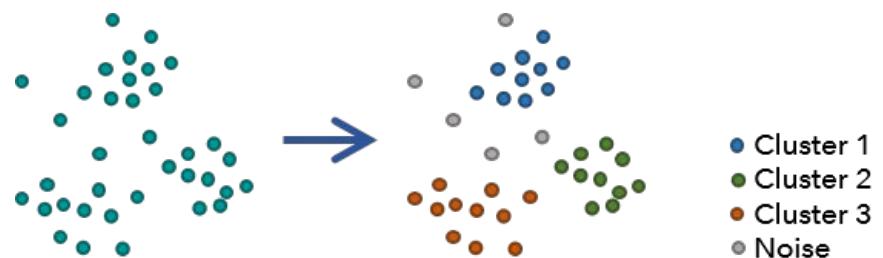
# Основные направления МО

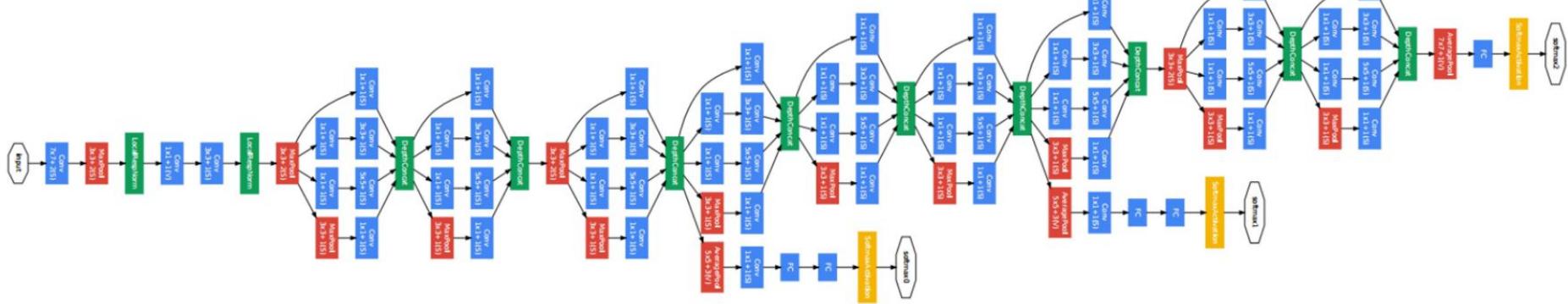


**Обучение с учителем (supervised learning)** –  
построение алгоритмов неявным образом за счет  
взаимодействия с **размеченной выборкой**



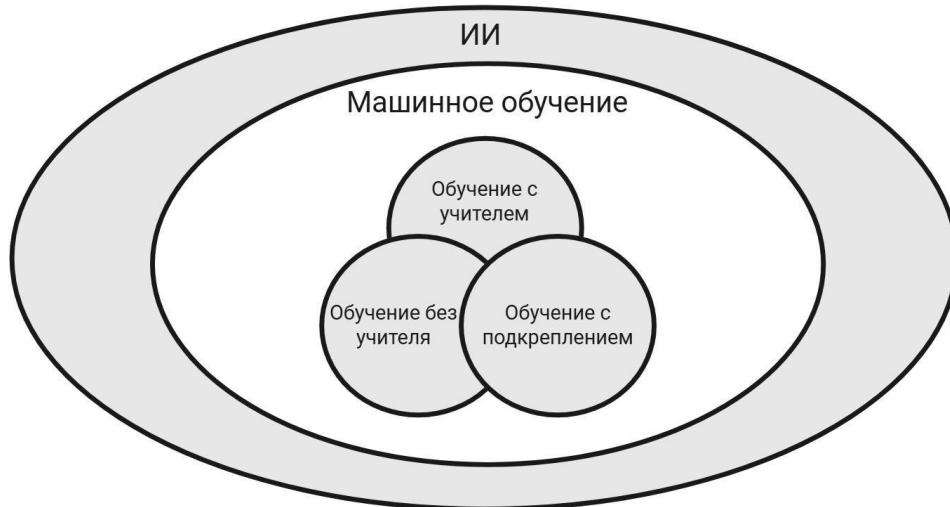
**Обучение без учителя (unsupervised learning)** –  
построение алгоритмов неявным образом за счет  
взаимодействия с **неразмеченной выборкой**



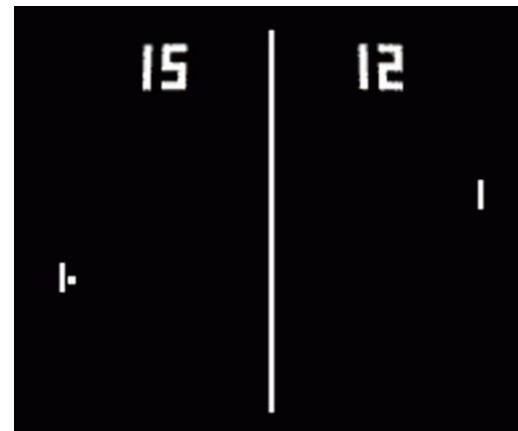


$$f(x) = y$$

# Основные направления МО

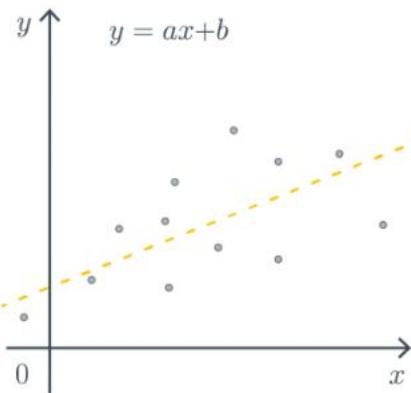


**Обучение с подкреплением** - построение алгоритмов неявным образом за счет взаимодействия со средой **методом проб и ошибок**



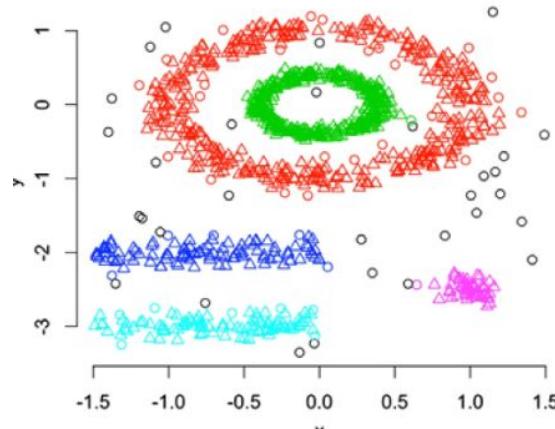
## Обучение с учителем

Модель делает  
прогноз на основе  
**размеченных** данных



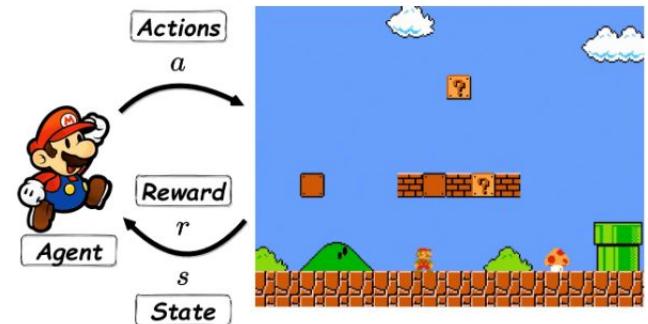
## Обучение без учителя

Модель напрямую не  
взаимодействует с  
правильными ответами

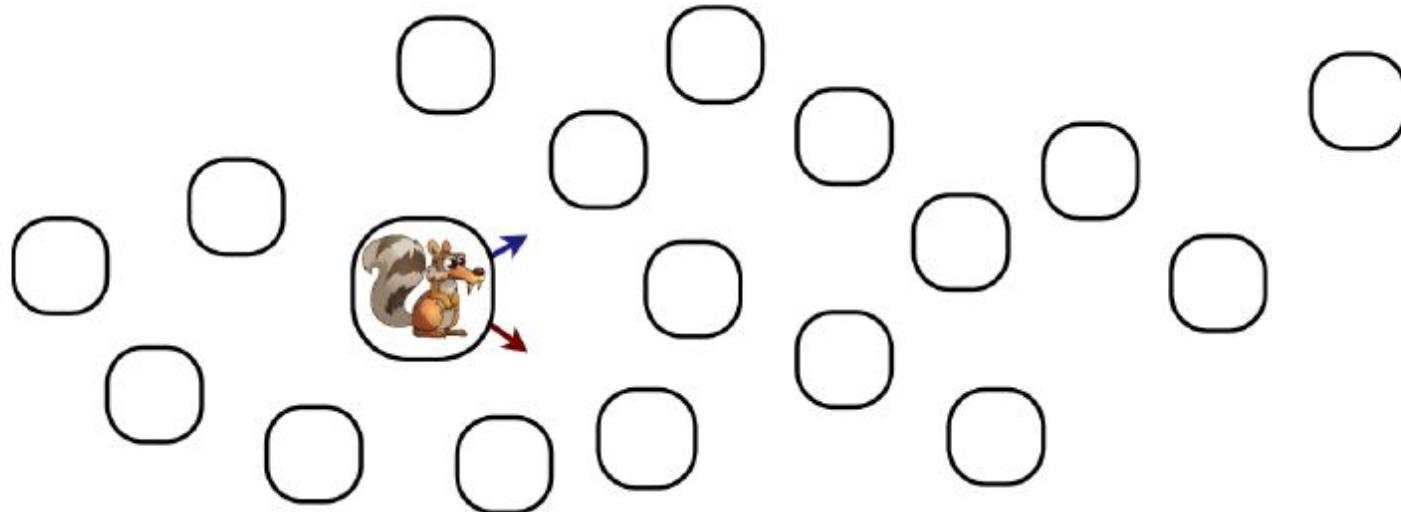


## Обучение с подкреплением

Модель получает **награду**  
за правильное действие на  
основании строгих  
критериев



# Агент и среда



**Агент** = Контроллер (код, принимающий решение, что делать дальше)

**Среда** = Всё остальное:

- "Железо" (сервомоторы, двигатели, клешня, лапы, ноги и так далее),
- Физический симулятор,
- Объекты в зоне досягаемости (или другой агент)

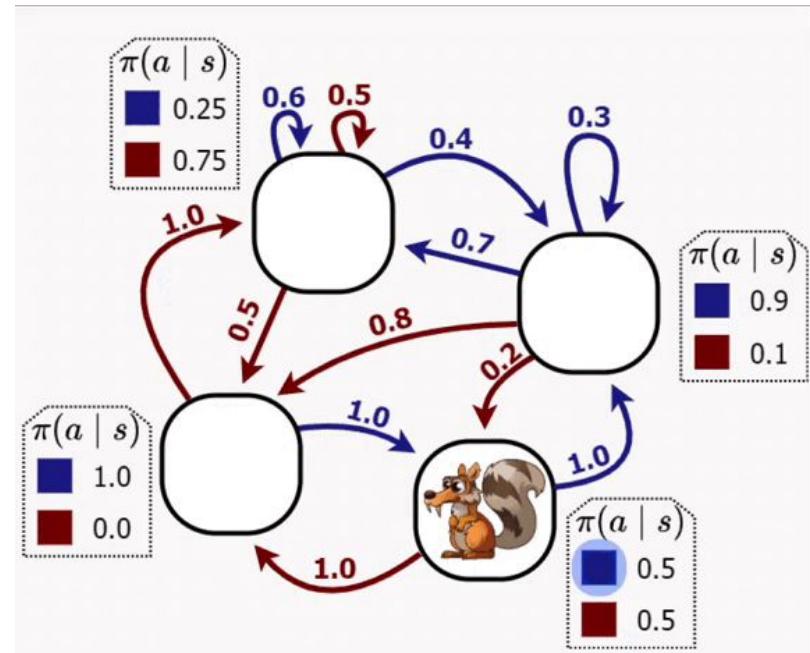
# Политика

Задача найти стратегию  $\pi$ ,  
максимизирующую среднюю награду:

$$\mathbb{E}_{\mathcal{T} \sim \pi} \sum_{t \geq 0} r_t \rightarrow \max_{\pi},$$

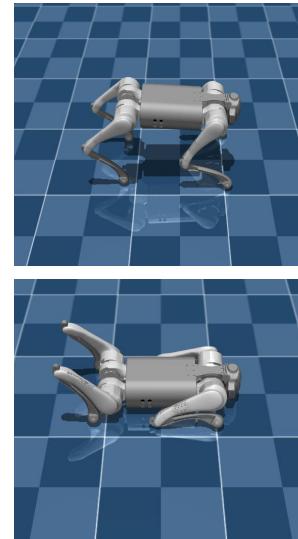
Награда за стратегию:

$$J(\pi) := \mathbb{E}_{\mathcal{T} \sim \pi} R(\mathcal{T})$$



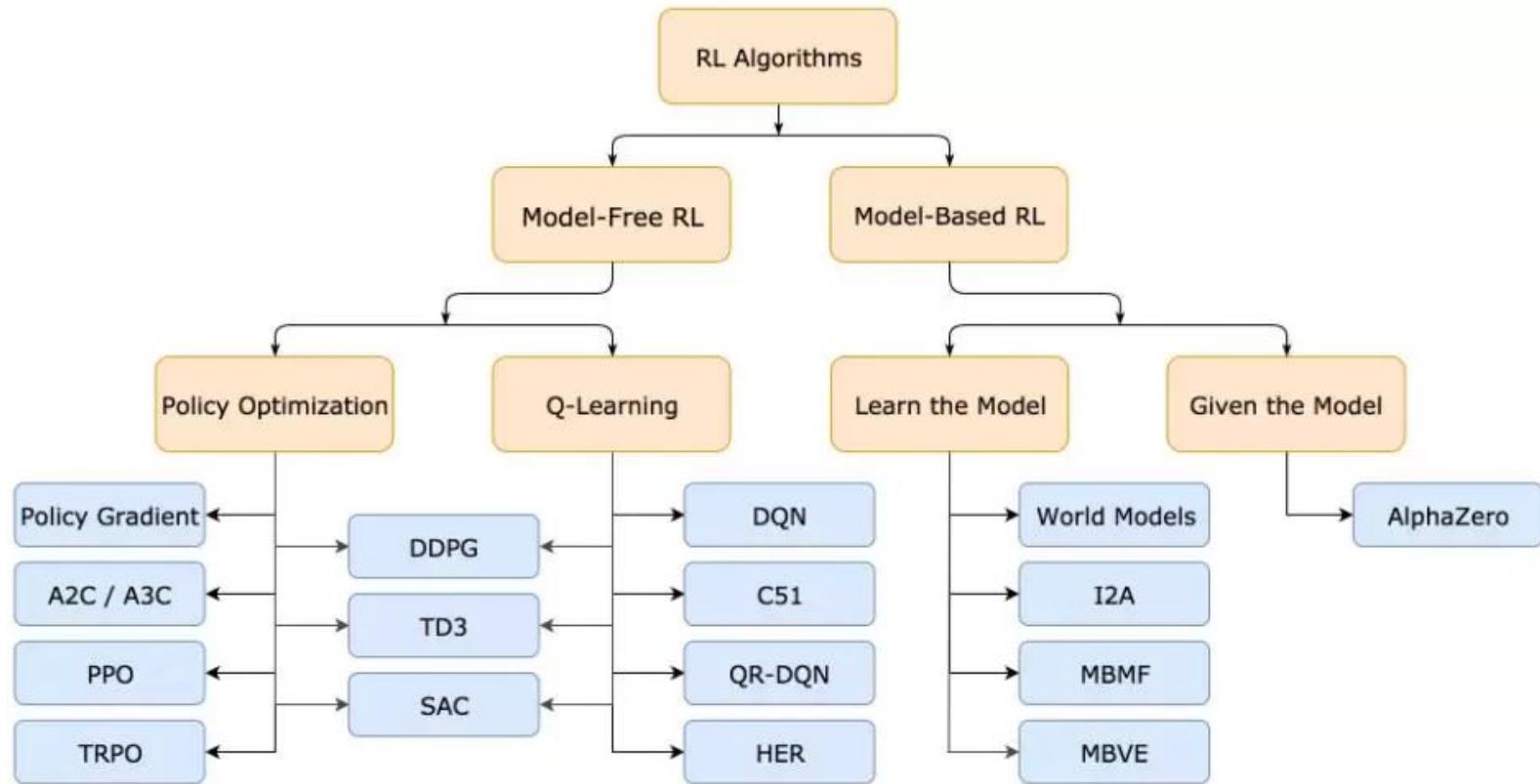
Дисконтирование награды – домножение суммарной награды на коэффициент для учета долгосрочной выгоды

# Итеративность обучения



- Терминальное состояние – состояние завершения эпизода
- Эпизод – цикл от начального состояния до терминального

# Виды RL-алгоритмов



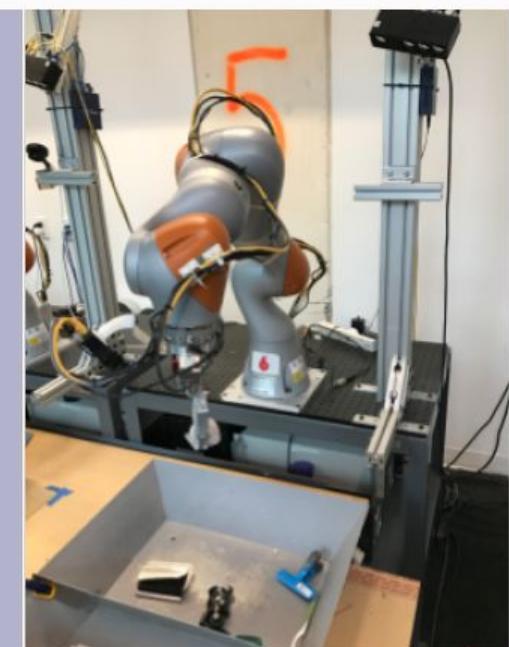
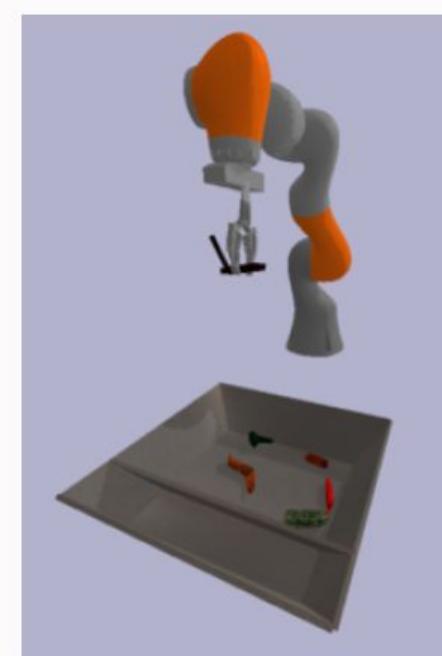
# Model-based



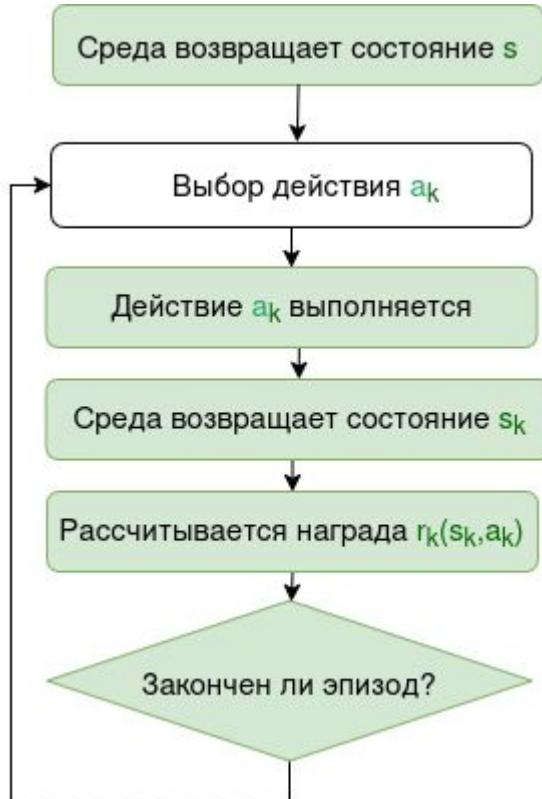
13	2	3	12
9	11	1	10
6	4	14	
15	8	7	5

Знаем и учим будущее  
состояние

# Model-free



Информация о динамике среды  
закрыта



# Цикл RL-агента

Необходимо определить стратегию для **выбора оптимальных действий** — таких, чтобы достичь максимальной награды

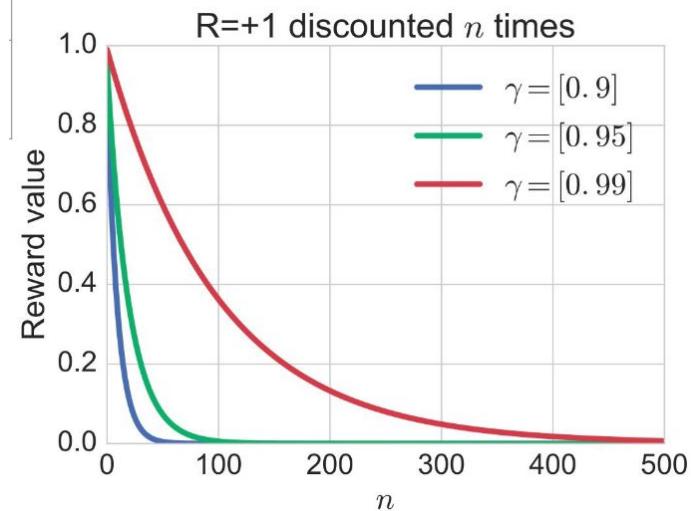
# Оценочная функция

Сумма всех **будущих** наград.

Оценочная функция показывает будущую награду, если в текущем состоянии **s** выбрать действие **a**.

Вклад отдалённых наград меньше, т. к. награды умножаются на **коэффициент дисконтирования**.

$$G_0 = \sum_{k=0}^{\infty} \gamma^k = \frac{1}{1 - \gamma}$$



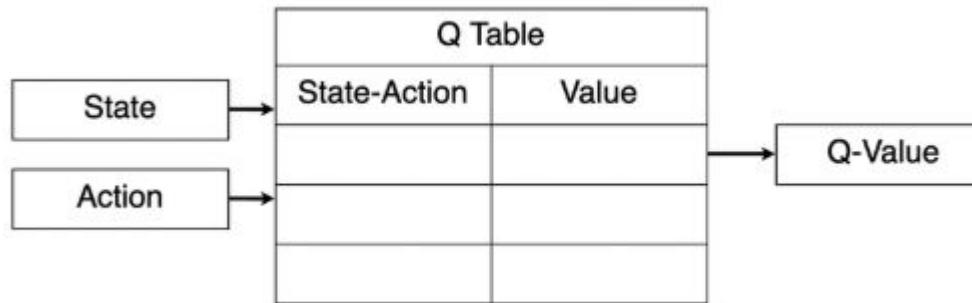
$$G_t \triangleq R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

discount factor

# Q-learning

Агент от среды получает **награды**, на основании которой может **сформировать оценочную функцию Q**.

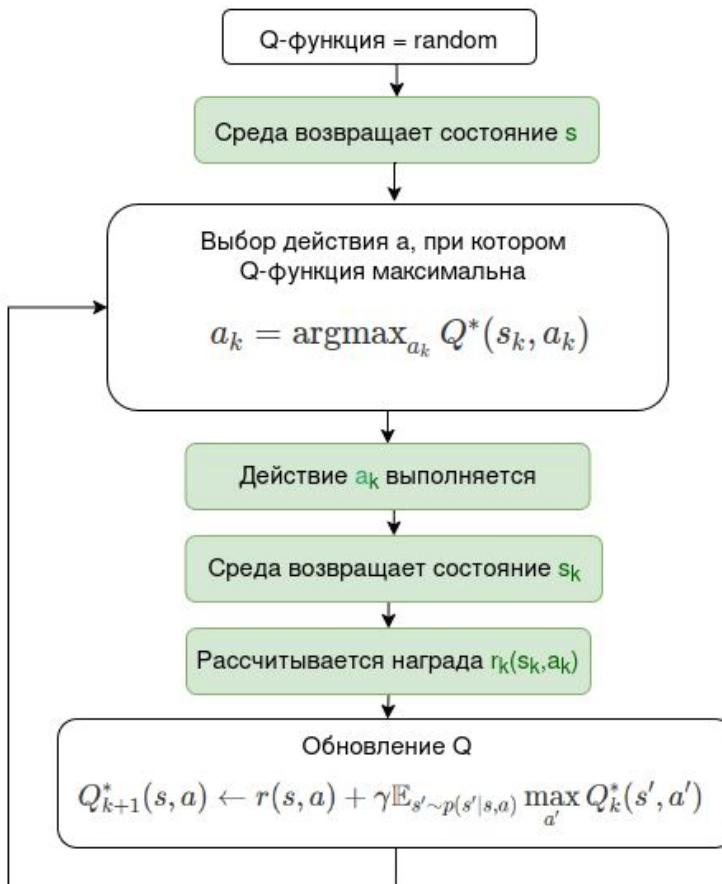
Функция позволяет без модели окружающей среды, оценивать полезность доступных действий.



$$Q^{new}(s_t, a_t) \leftarrow \underbrace{(1 - \alpha) \cdot Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \left( \underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\substack{\text{learned value} \\ \text{estimate of optimal future value}}} \right).$$

### Value-iteration

Итеративный подбор  
Q-функции

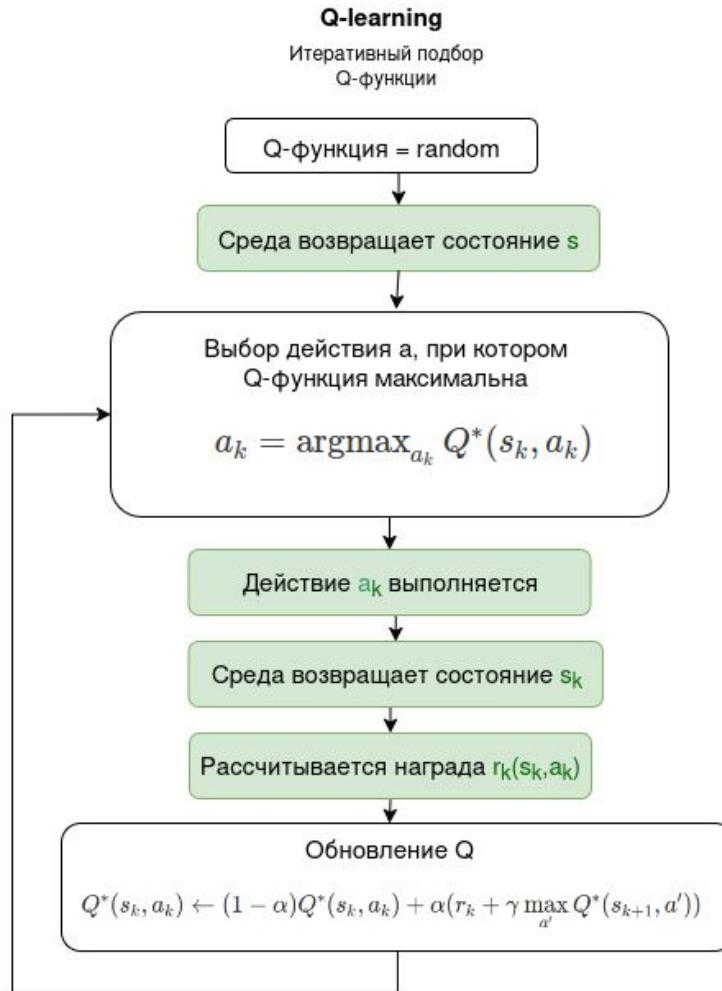


# Value-iteration

Model-based алгоритм

Подбираем **истинную** Q-функцию.

Алгоритм **динамического программирования** для поиска оптимальной стратегии для марковского процесса принятия решений (MDP) путём итеративного обновления значений каждого состояния.



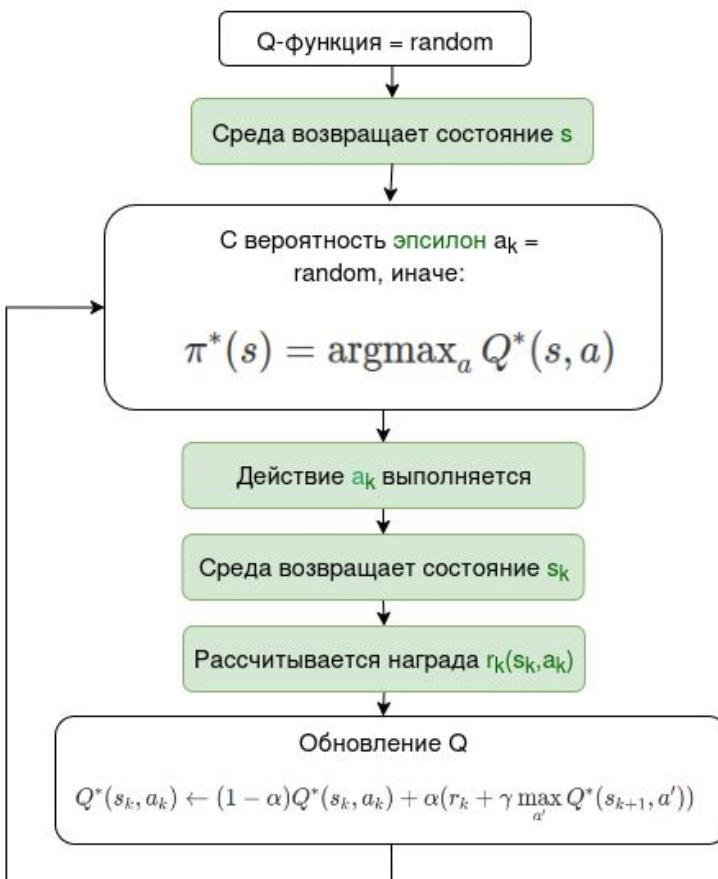
# Q-learning

Model-free алгоритм

Точная матрица переходов  
неизвестна, а значит неизвестны и  
точные будущие вознаграждения.

Подбираем **примерную** Q-функцию  
(стохастичную оценку)

**Q-learning**  
Итеративный подбор  
Q-функции



# Q-learning

Q-функция прогнозирует будущую награду, если в текущем состоянии  $s$  выбрать действие  $a$ .

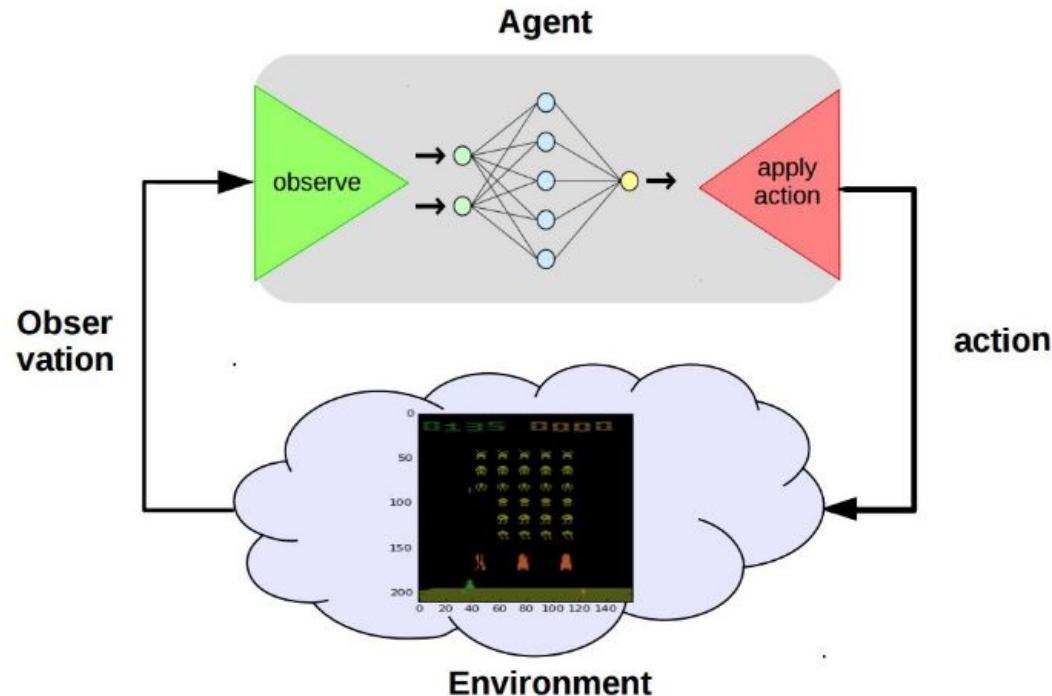
С вероятностью эпсилон выбираем не самые оптимальные действия, а случайные.

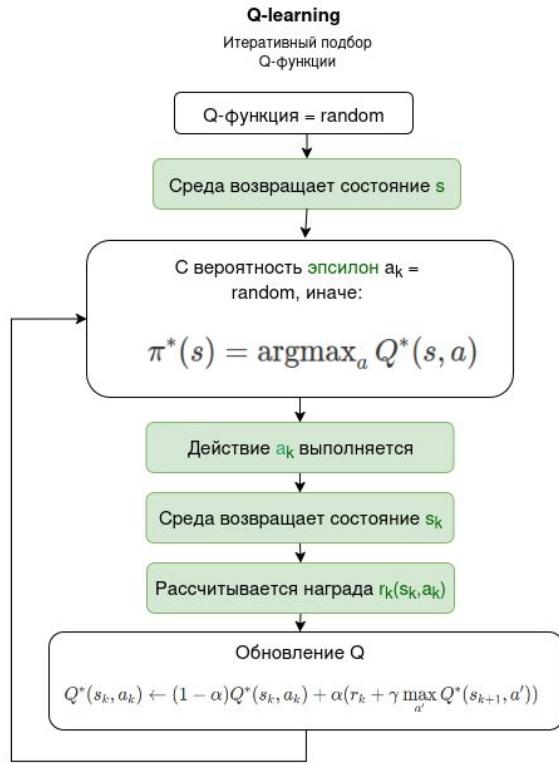
Алгоритм ищет не саму стратегию, а оптимальную Q-функцию для оценки состояний.

# Развитие идеи Q-learning

Если пространство состояний слишком большое, хранить  $Q$  как таблицу в памяти

Тогда можно заменить  $Q$ -функцию на  $Q$ -нейросеть



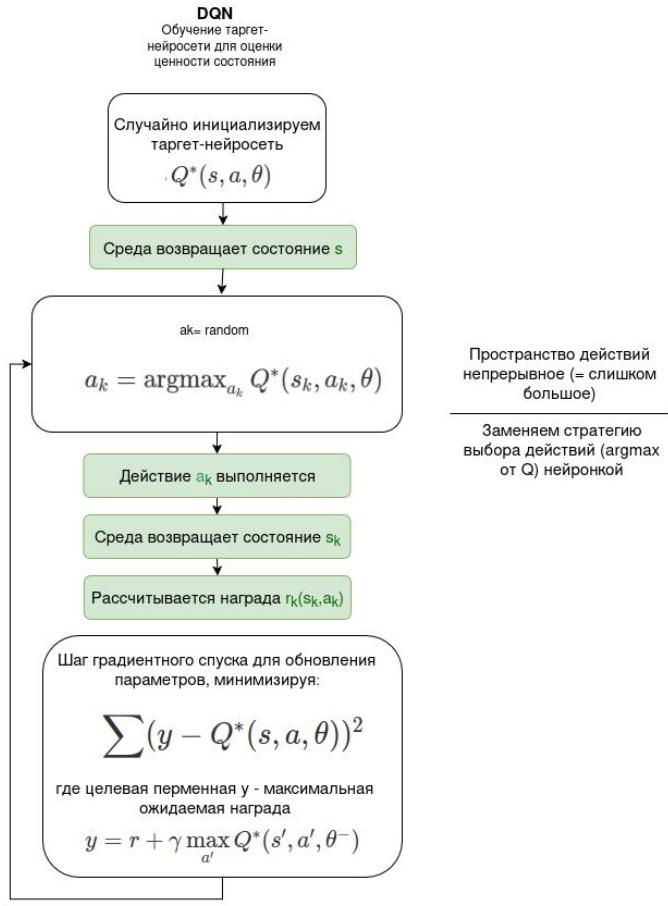


Пространство состояний слишком большое  
Заменяем функцию Q на нейросеть Q



# DQN

Вместо функции ценности Q — таргет-нейросеть для оценки ценности



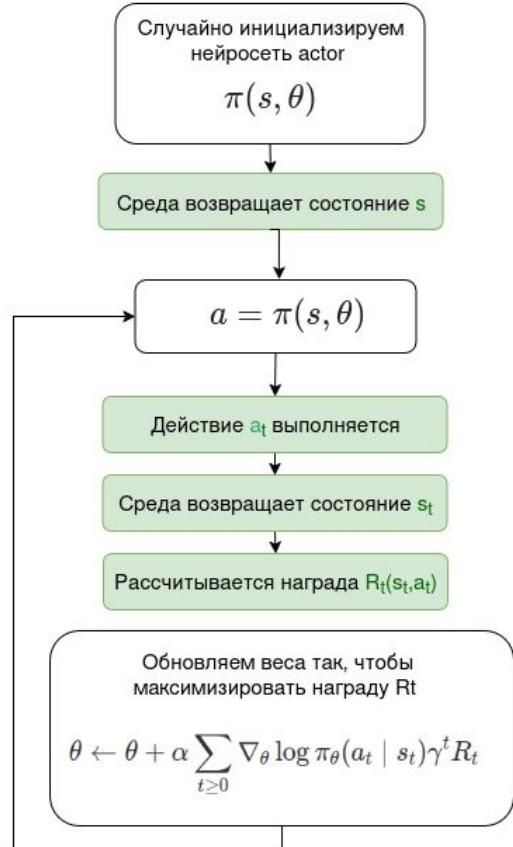
# Actor-Critic

-Вместо функции ценности Q — **Critic**-нейросеть для оценки ценности

-Вместо эпсилон-жадного выбора — **Actor**-нейросеть для предсказания действий

## Policy Gradient

Нейросеть для выбора действий



# Policy Gradient

Напрямую оптимизирует то, что нам нужно: **поведение** агента (а не оценочную функцию).

Примеры: A2C, TRPO, PPO

# Сбор данных в model-free алгоритмах

- (**off-policy**) Value-based подход подразумевает, что алгоритм ищет не саму стратегию, а оптимальную Q-функцию

Примеры: DDPG, TD3 и SAC

- (**on-policy**) Policy Gradient – градиенты по стратегиям. Возвращается вероятностное распределение действий, которое задается выходами нейронной сети.

Примеры: Advantage Actor-Critic (A2C), Trust-Region Policy Optimization (TRPO) и Proximal Policy Optimization (PPO)