

**Севастопольский государственный университет  
Институт информационных технологий**

# **Методы и системы искусственного интеллекта**

**Бондарев Владимир Николаевич**

**Лекция**  
**Обучение с подкреплением**  
**(reinforcement learning, RL)**  
**Q-обучение с аппроксимацией.**

**Бондарев Владимир Николаевич**

# Обучение с подкреплением

По-прежнему среда, описывается Марковским процессом принятия решений (MDP), который определяется:

1. Множеством состояний  $s \in S$ ;
2. Множеством действий  $a \in A$ ;
3. Моделью переходов  $T(s,a,s')$ ;
4. Функцией вознаграждения  $R(s,a,s')$ .

Цель решения: также заключается в поиске политики  $\pi(s)$ ;

**Особенности задачи обучения с подкреплением:**

1. В отличие от задачи решения MDP, которая является задачей **офф-лайн планирования**, где агент обладает полной информацией, необходимой для предварительного вычисления оптимального действия без его реального выполнения, задача **RL** является задачей **он-лайн планирования**, где агент выполняет реальные действия.
2. При этом у агента нет априорных знаний функции переходов  $T$  и вознаграждений  $R$ . Поэтому агент должен осуществлять исследование среды, выполняя действия и получая обратную связь в виде последующих состояний  $s'$  и соответствующих вознаграждений  $r$

**Основная идея :** Выполнять усреднение всех наград, используя результаты выборов

# Сопоставление: MDP и RL

## Известный MDP: офф-лайн решение

### Цель

Вычисление  $V^*$ ,  $Q^*$ ,  $\pi^*$

Оценка фиксированной  $\pi$

### Метод/алгоритм

Итерации по значениям/политикам

Оценка политики

## Неизвестный MDP, основанный на модели

### Цель

Вычисление  $V^*$ ,  $Q^*$ ,  $\pi^*$

Оценка фиксир.  $\pi$

### Метод

VI/PI по аппр. MDP

PE по аппрок. MDP

## Неизвестный MDP, без модели

### Цель

Вычисление  $V^*$ ,  $Q^*$ ,  $\pi^*$

Оценка фиксир.  $\pi$

### Метод

Q-обучение

Обучение значений

# Аналогия: Средний возраст

Цель: Вычислить средний возраст группы

Известная  $P(A)$

$$E[A] = \sum_a P(a) \cdot a = 0.35 \times 20 + \dots$$

Без  $P(A)$ , просто накапливаем выборки  $[a_1, a_2, \dots, a_N]$

Неизв.  $P(A)$ : “На основе модели”

Почему работает? Т.к. в итоге обучаемся правильной модели.

$$\hat{P}(a) = \frac{\text{num}(a)}{N}$$
$$E[A] \approx \sum_a \hat{P}(a) \cdot a$$

Неизв.  $P(A)$ : “Без модели”

$$E[A] \approx \frac{1}{N} \sum_i a_i$$

Почему работает? Т.к. выборки появляются с определенной частотой.

# Оценивание политики на основе выборок

- Мы хотим улучшить оценку  $V$ , вычислив среднее:

$$V_{k+1}^{\pi}(s) \leftarrow \sum_{s'} T(s, \pi(s), s') [R(s, \pi(s), s') + \gamma V_k^{\pi}(s')]$$

- Идея:** Взять выборки результатов переходов и усреднить

$$sample_1 = R(s, \pi(s), s'_1) + \gamma V_k^{\pi}(s'_1)$$

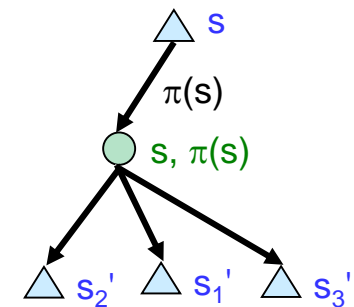
$$sample_2 = R(s, \pi(s), s'_2) + \gamma V_k^{\pi}(s'_2)$$

...

$$sample_n = R(s, \pi(s), s'_n) + \gamma V_k^{\pi}(s'_n)$$

$$V_{k+1}^{\pi}(s) \leftarrow \frac{1}{n} \sum_i sample_i$$

- Таким образом мы смогли бы выполнить шаг обновления значения  $V$



Не работает, т.к.  
мы не можем  
обратить время:  
чтобы получить  
новую выборку после  
уже совершенного  
перехода из  $s$  и  $s'$

# Обучение, без модели

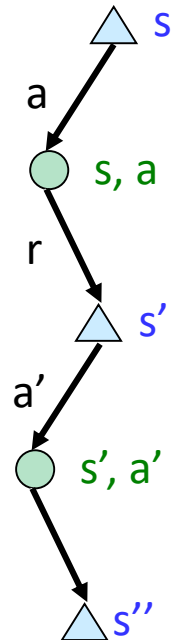
---

- Обучение без модели (временное различие)

- Получить опыт, реализуя эпизоды

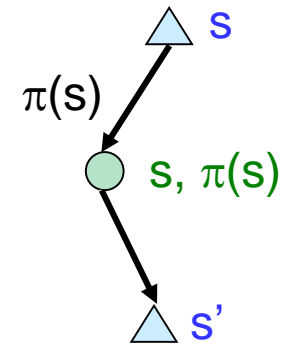
$(s, a, r, s', a', r', s'', a'', r'', s'''' \dots)$

- Обновить оценки каждого перехода  $(s, a, r, s')$
  - Для всех моментов времени, обновления соответствуют обновлениям Беллмана



# Обучение на основе временных различий (temporal difference -TD)

- Основная идея: обучаться на основе каждого опыта (попытки)!
  - Обновлять  $V(s)$  каждый раз, при испытании перехода  $(s, a, s', r)$
  - Более вероятные исходы будут вносить вклад в обновления более часто
- TD – обучение
  - Политика фиксируется, пока выполняется оценивание!
  - Выполнять скользящее усреднение выборок



Выборка  $V(s)$ :  $sample = R(s, \pi(s), s') + \gamma V^\pi(s')$

Обновление  $V(s)$ :  $V^\pi(s) \leftarrow (1 - \alpha)V^\pi(s) + (\alpha)sample$

Или:  $V^\pi(s) \leftarrow V^\pi(s) + \alpha(sample - V^\pi(s))$



# Пример: TD-обучение

Состояния

|   |   |   |
|---|---|---|
|   | A |   |
| B | C | D |
|   | E |   |

Примеч:  $\gamma = 1$ ,  $\alpha = 1/2$

Наблюдаемые переходы

B, east, C, -2

|   |   |   |
|---|---|---|
|   | 0 |   |
| 0 | 0 | 8 |
|   | 0 |   |

C, east, D, -2

|    |   |   |
|----|---|---|
|    | 0 |   |
| -1 | 0 | 8 |
|    | 0 |   |

|    |   |   |
|----|---|---|
|    | 0 |   |
| -1 | 3 | 8 |
|    | 0 |   |

$$V^\pi(s) \leftarrow (1 - \alpha)V^\pi(s) + \alpha [R(s, \pi(s), s') + \gamma V^\pi(s')]$$

$$0 + 1/2(-2 + 0) = -1$$

$$0 + 1/2(-2 + 8) = 3$$

# Аппроксимация ценностей с помощью выборок

---

- Оценка политики:

$$V_{k+1}^{\pi}(s) \leftarrow \sum_{s'} T(s, \pi(s), s') [R(s, \pi(s), s') + \gamma V_k^{\pi}(s')]$$



- Итерации по значениям ценности состояний:

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V_k(s')]$$



- Итерации по Q-ценностям:

$$Q_{k+1}(s, a) \leftarrow \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma \max_{a'} Q_k(s', a')]$$



# Q-обучение

- Q-обучение - итерации по Q-ценностям с использованием выборок

$$Q_{k+1}(s, a) \leftarrow \sum_{s'} T(s, a, s') \left[ R(s, a, s') + \gamma \max_{a'} Q_k(s', a') \right]$$

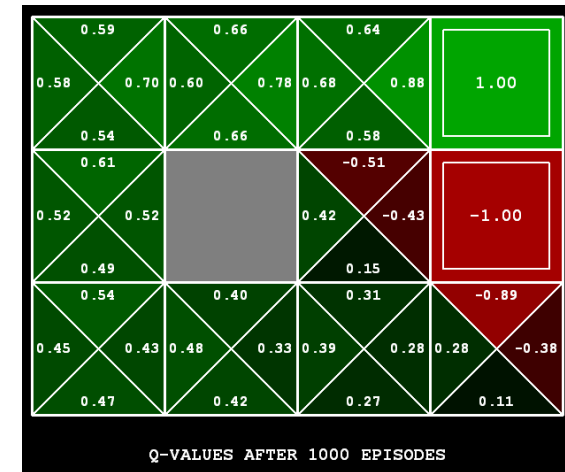
- Шаги Q-обучения:

- Получить выборку  $(s, a, s', r)$
- Определить предыдущую оценку q-состояния  $(s, a)$ :  $Q(s, a)$
- Получить новую выборочную оценку:

$$sample = R(s, a, s') + \gamma \max_{a'} Q(s', a')$$

- Выполнить обновление оценки q-состояния на основе скользящего усреднения выборки:

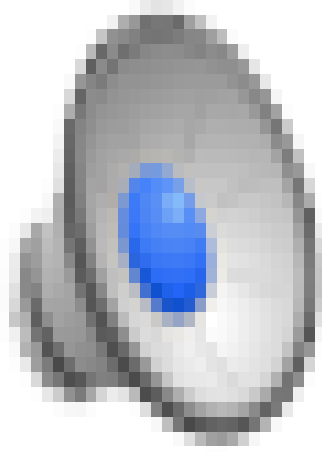
$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + (\alpha) [sample]$$



Избавляет от  
оценивания  
политики

# Q-обучение в мире Gridworld

---



# Свойства Q-обучения

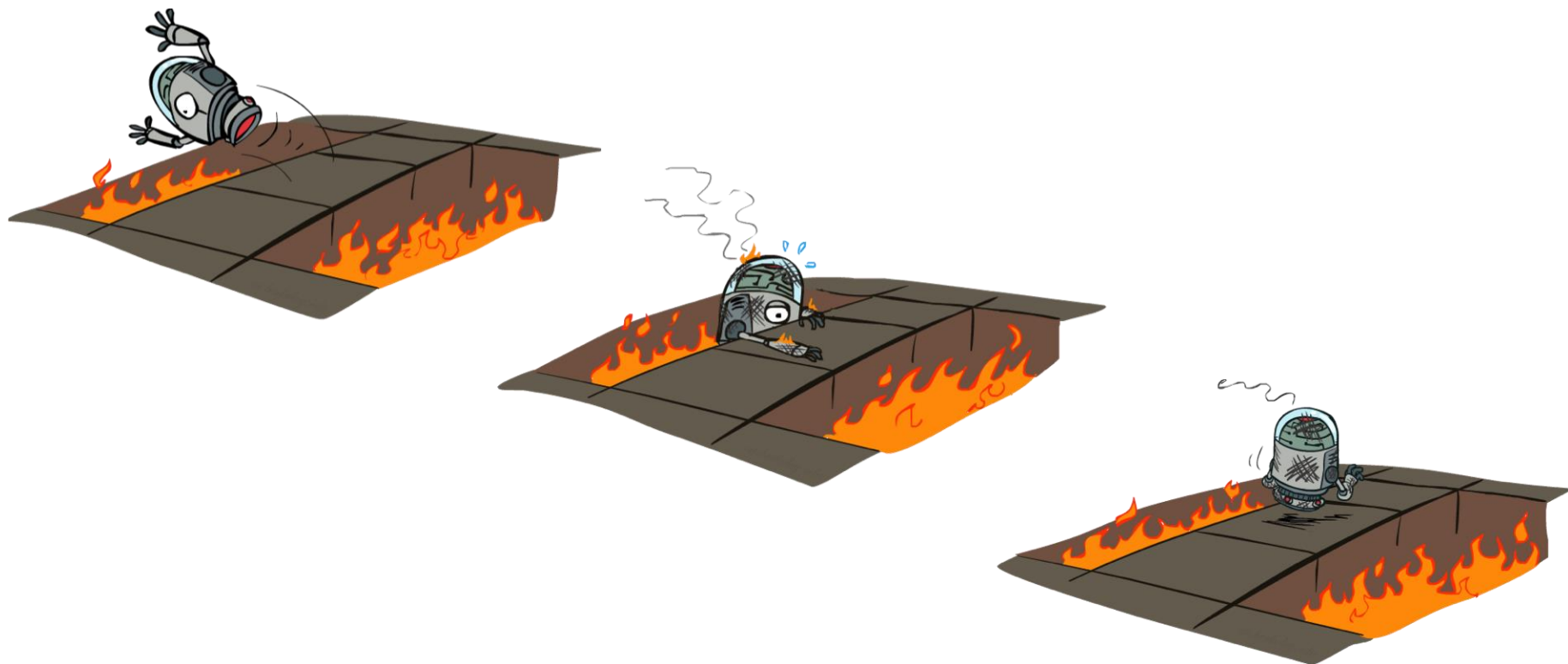
- **Впечатляющий результат:** Q-обучение сходится к оптимальным значениям, т.е. обучается оптимальным q-значениям – даже если агент действует не всегда оптимально! Это делает Q-обучение таким революционным !



- Q-обучение относится **к off-policy обучению, т.е. к обучению без привязки к политике** (в то время как TD-обучение и прямая оценка изучают ценности состояний в рамках политики, следуя политике, прежде чем определять оптимальность политики с помощью других методов).
  - Предостережения:
    - Необходимо, чтобы объем исследований был достаточным
    - Скорость обучения должна стать достаточно малой
    - ... но она не должна снижаться слишком быстро
- По сути, в пределе, не имеет значения как выбираются действия!

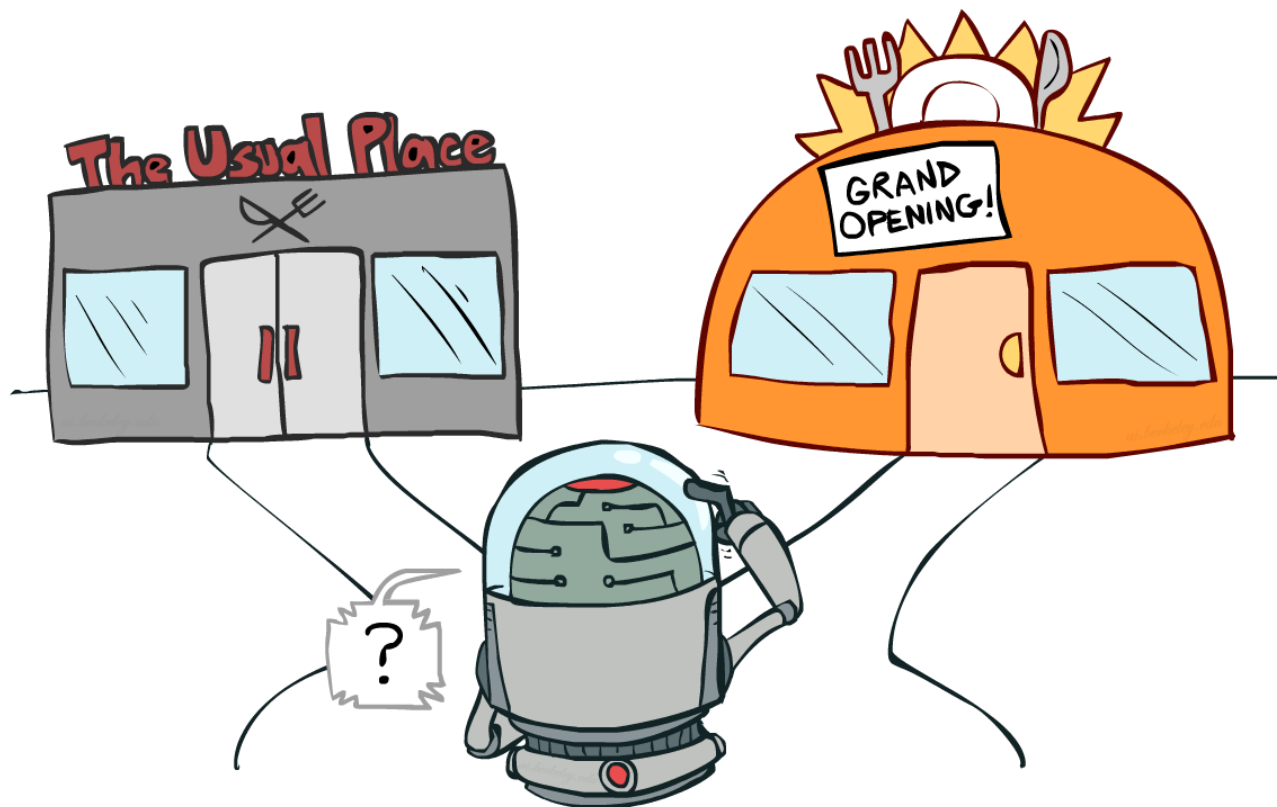
# Активное обучение с подкреплением

---



# Исследование против Эксплуатации

---

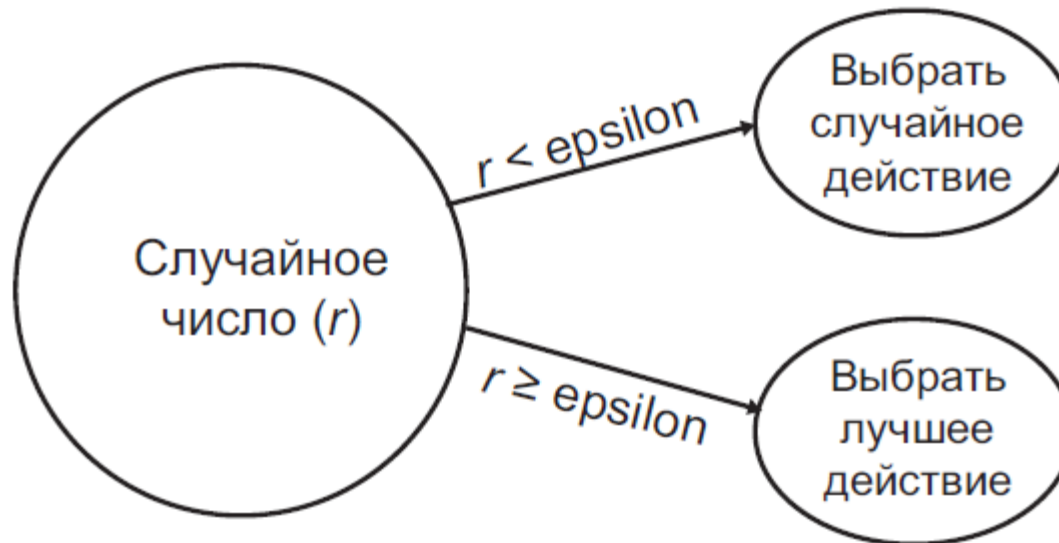


## ε-жадная стратегия

*Жадная стратегия* выбирает оптимальное действие **среди уже исследованных**.

Что лучше искать: любое лучшее действие или действие, лучшее из всех исследованных действий? Это называется *дилеммой между исследованием и эксплуатацией*.

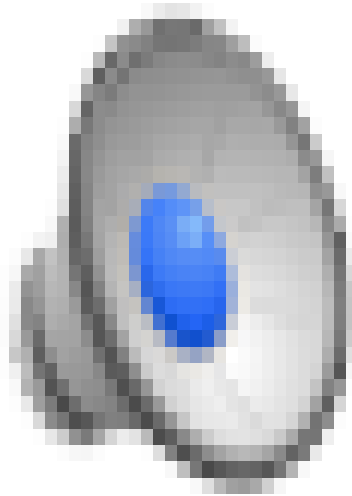
Чтобы разрешить дилемму, вводится **ε-жадная стратегия**: действие выбирается на основе текущей политики с вероятностью  $1 - \epsilon$  и с небольшой вероятностью  $\epsilon$  будет выполняться опробывание новых случайных действий (**исследование**). Значение  $\epsilon$  должно уменьшаться со временем, поскольку заниматься исследованиями до бесконечности незачем. Таким образом, со временем политика переходит **на эксплуатацию** «хороших» действий:





# Q-обучение – эпсилон жадная стратегия – Crawler

---



# Функция разведки (исследования)

---

Проблему ручного выбора  $\epsilon$  можно решить с помощью функции разведки

- **Функция разведки  $f(s,a)$**

- Использует оценку q-ценности состояния и счетчик числа посещений  $N(s,a)$  этого состояния:

$$f(s,a) = Q(s,a) + \frac{k}{N(s,a)}$$

- Функция предоставляет «бонус» некоторым редко выполняемым действиям и обеспечивает выбор действий с более высокими значениями q

Обычное Q-обновление:

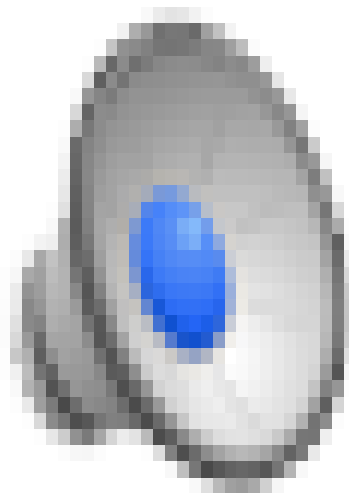
$$Q(s,a) \leftarrow_{\alpha} R(s,a,s') + \gamma \max_{a'} Q(s',a')$$

Модифицированное Q-обучение:

$$Q(s,a) \leftarrow_{\alpha} R(s,a,s') + \gamma \max_{a'} f(Q(s',a'), N(s',a'))$$

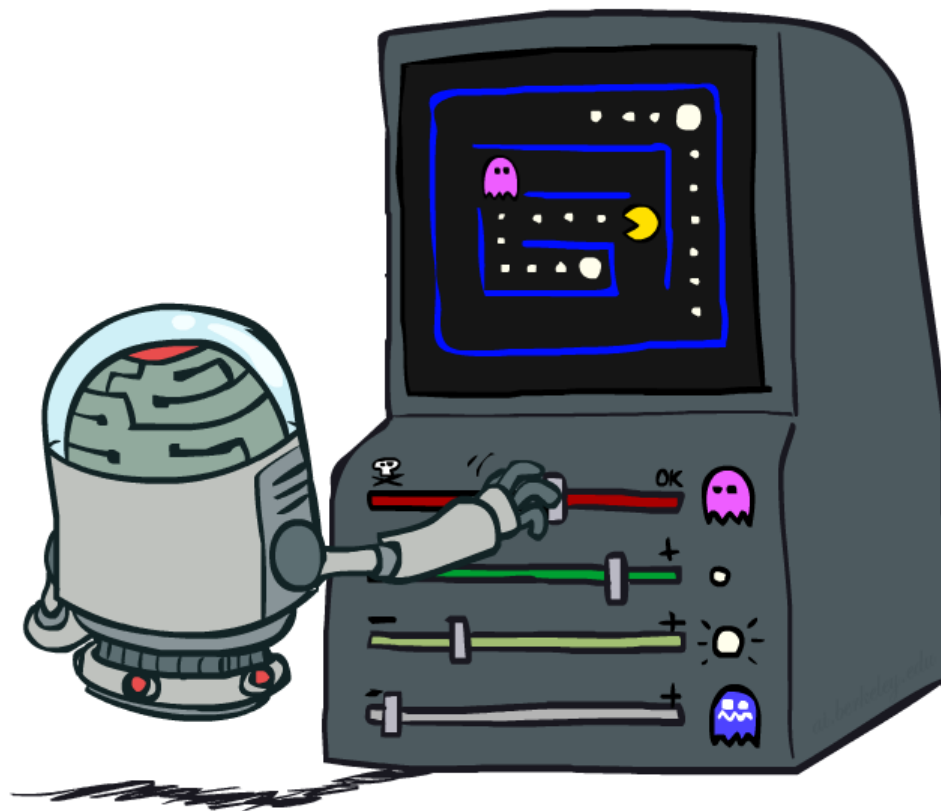
# Q-обучение – функция разведки – Crawler

---



# Q-обучение с аппроксимацией

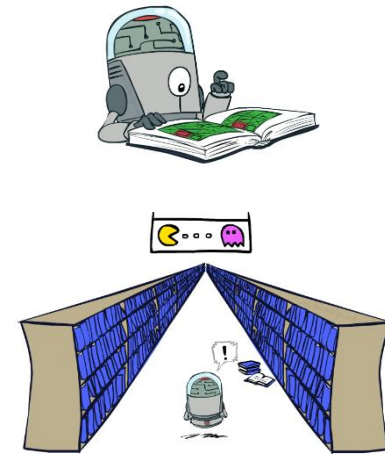
---



# Обобщение представления состояний

---

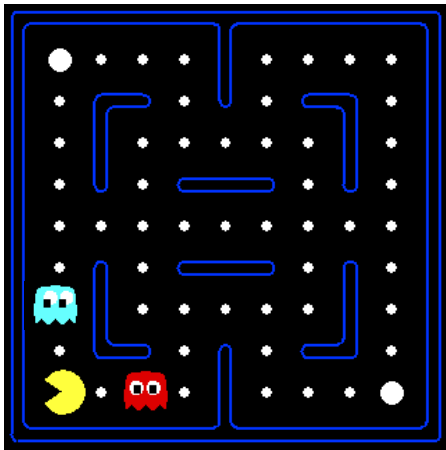
- Базовое Q-обучение требует хранения таблицы всех  $q$ -состояний
- В реальной ситуации мы не сможем обучаться по данным каждого отдельного состояния:
  - Слишком много состояний для посещений во время обучения
  - Слишком много состояний для запоминания в  $q$ -таблицах
- Вместо этого, мы хотели бы ввести обобщения:
  - Изучить небольшое количество обучающих состояний на опыте;
  - Обобщить этот опыт на новые похожие ситуации;
  - Это фундаментальная идея машинного обучения, с которой мы будем сталкиваться и далее.



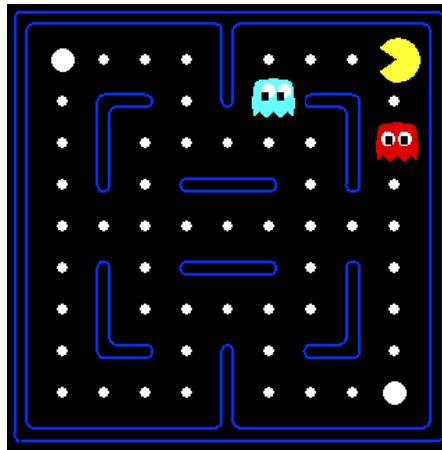
# Пример: Пакман

---

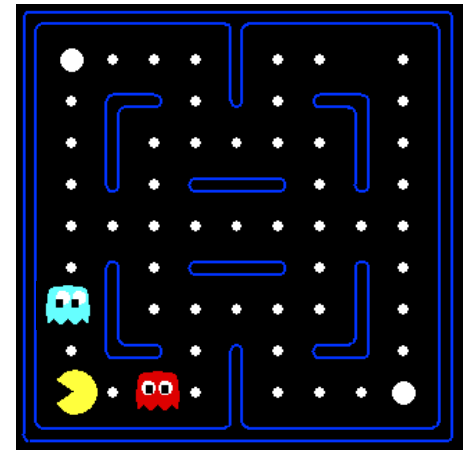
На собственном опыте Пакман обнаружил, что это состояние-ловушка плохое:



В наивном q-обучении Пакман не догадается, что это состояние тоже плохое:

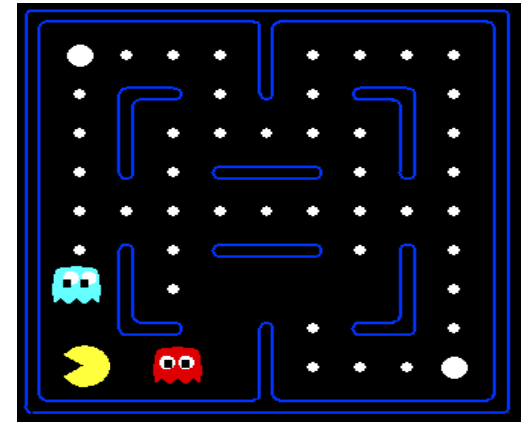


И даже не догадается, что и это состояние неблагоприятно!



# Представления состояний в виде признаков

- Решение проблемы : **описать ценность состояний** с помощью вектора признаков
  - **Признаки** - это функции, которые отображают состояния на действительные числа и которые фиксируют важные свойства состояний.
  -
- Примеры признаков:**
  - Расстояние до ближайшего призрака;
  - Расстояние до ближайшей гранулы;
  - Число призраков;
  - $1/(\text{расстояние до призрака})^2$
  - Находится ли Пакман в ловушке? (0/1)
  - ..... и др..
- Также можно с помощью признаков описывать и q-состояния (например, действие приближающее к еде)



# Линейные функции ценности

---

- Используя признаки, мы можем представить  $q$  - функцию (или функцию ценности состояний) в виде линейной комбинации признаков с весами:

$$V(s) = w_1 f_1(s) + w_2 f_2(s) + \dots + w_n f_n(s)$$

$$Q(s, a) = w_1 f_1(s, a) + w_2 f_2(s, a) + \dots + w_n f_n(s, a)$$

- Преимущество: накопленный опыт может представлен набором нескольких признаков.
- Недостаток: состояния могут характеризоваться общими признаками, но при этом сильно различаются по ценности!



# Q-обучение с аппроксимацией

- Q-обучение с аппроксимацией:

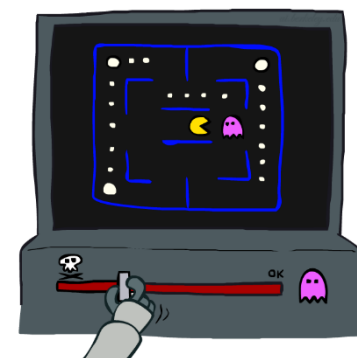
- 1. Выборка  $(s, a, s', r)$

- 2.  $\text{difference} = \left[ r + \gamma \max_{a'} Q(s', a') \right] - Q(s, a)$

$$Q(s, a) \leftarrow Q(s, a) + \alpha [\text{difference}]$$

Ранее точное Q

- 3.  $w_i \leftarrow w_i + \alpha [\text{difference}] f_i(s, a)$  Теперь аппроксимация Q



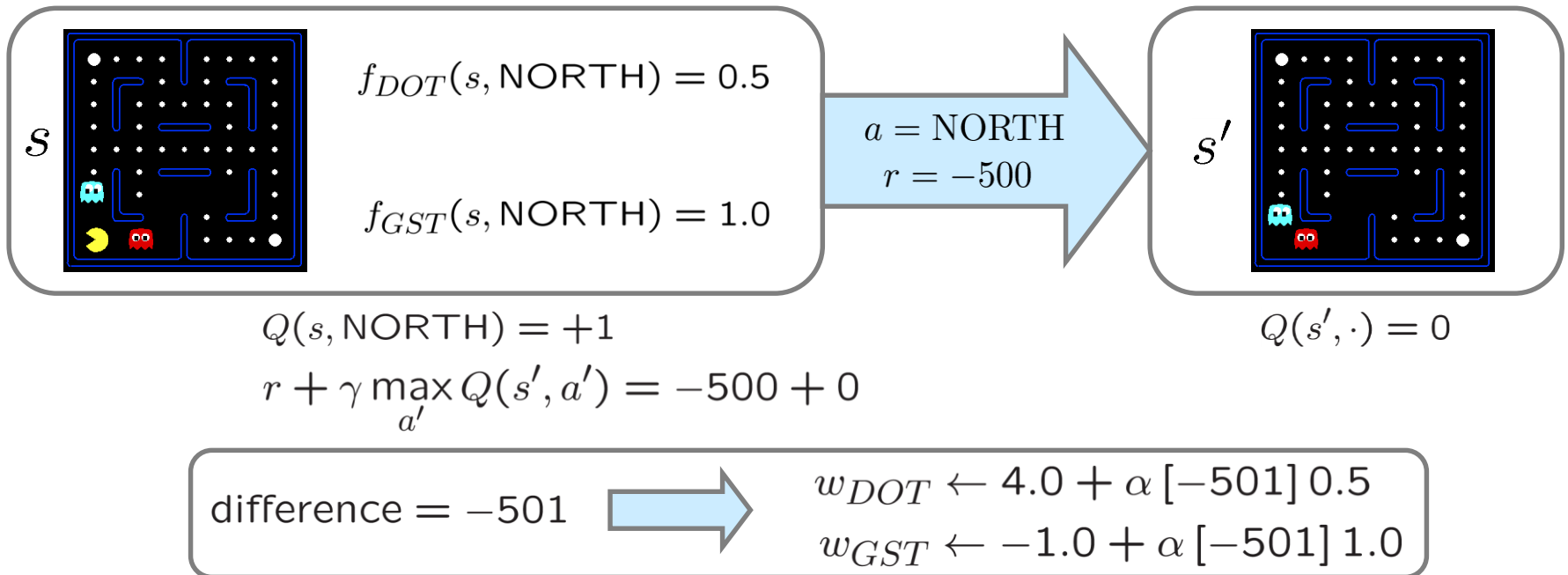
$$Q(s, a) = w_1 f_1(s, a) + w_2 f_2(s, a) + \dots + w_n f_n(s, a)$$

Для каждого состояния Q-обучение с аппроксимацией позволяет нам хранить только один весовой вектор и мы можем вычислять Q-ценности по мере необходимости. В результате это дает нам обобщенную версию Q-обучения.

# Пример: Q-Распан

$$Q(s, a) = 4.0 f_{DOT}(s, a) - 1.0 f_{GST}(s, a)$$

$$\text{difference} = \left[ r + \gamma \max_{a'} Q(s', a') \right] - Q(s, a)$$

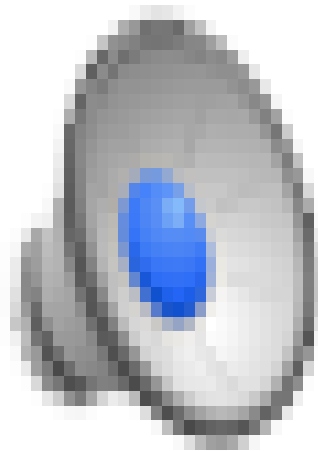


$$Q(s, a) = 3.0 f_{DOT}(s, a) - 3.0 f_{GST}(s, a)$$

$$\alpha = 0.004$$

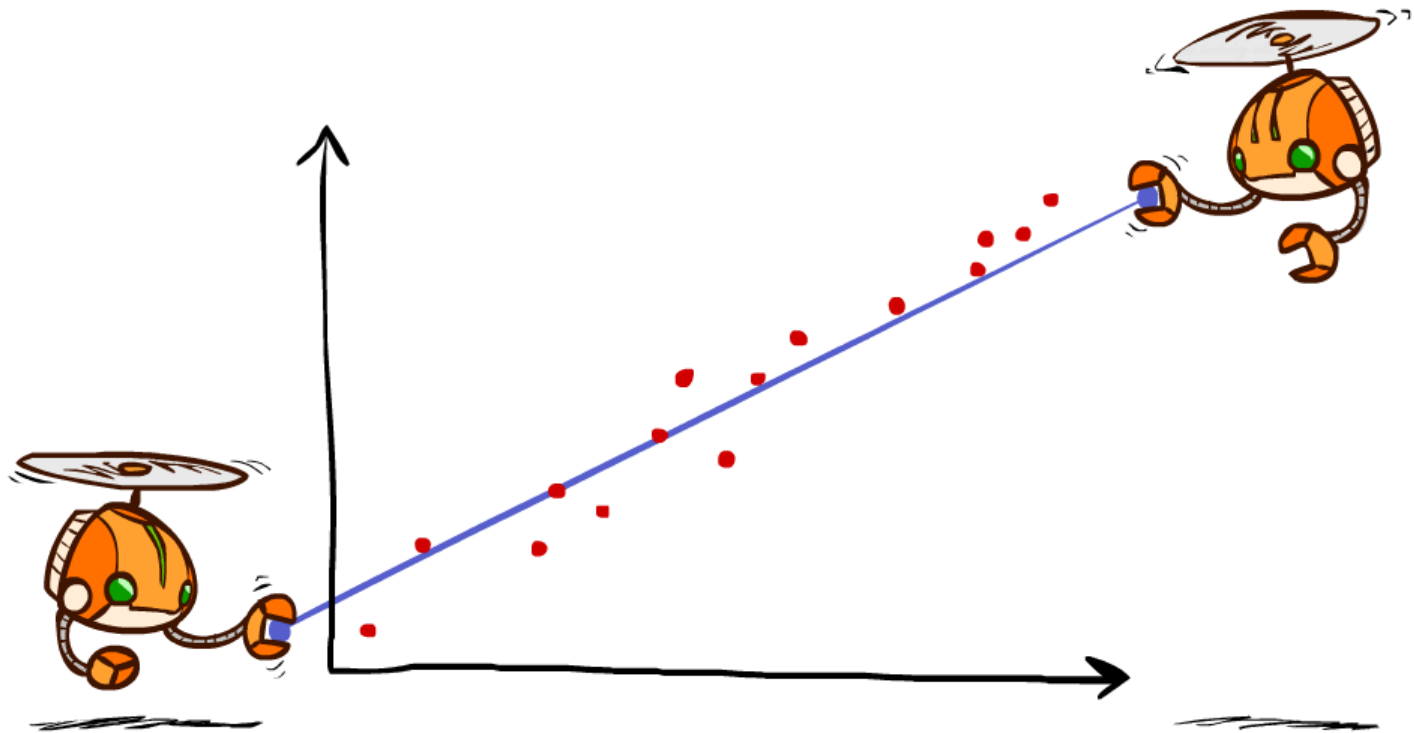
# Q-обучение с аппроксимацией -- Расстан

---

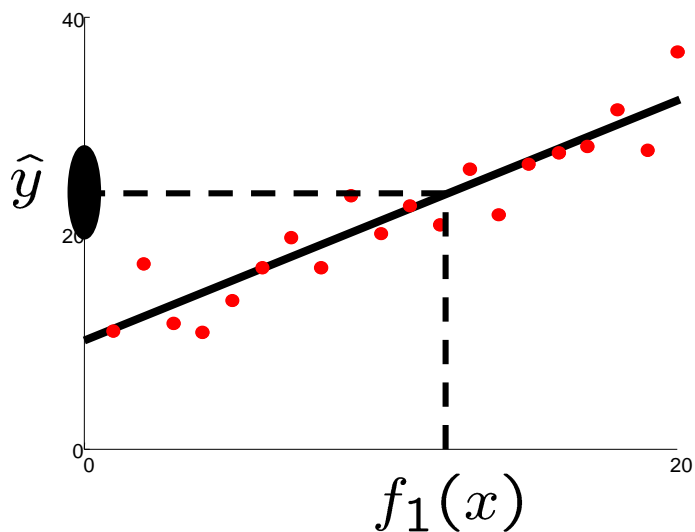


# Q-обучение и МНК

---

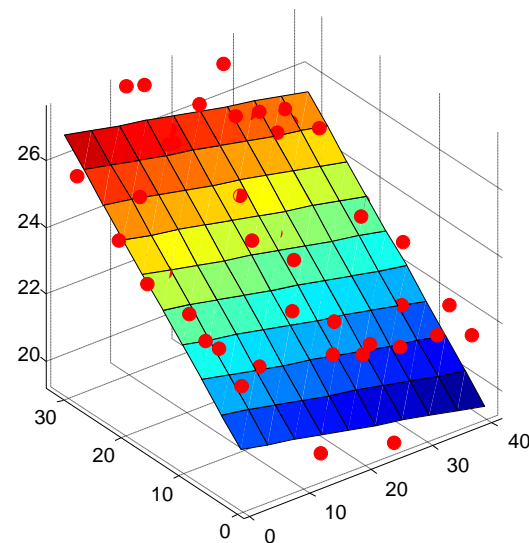


# Линейная аппроксимация: регрессия



Предсказание:

$$\hat{y} = w_0 + w_1 f_1(x)$$

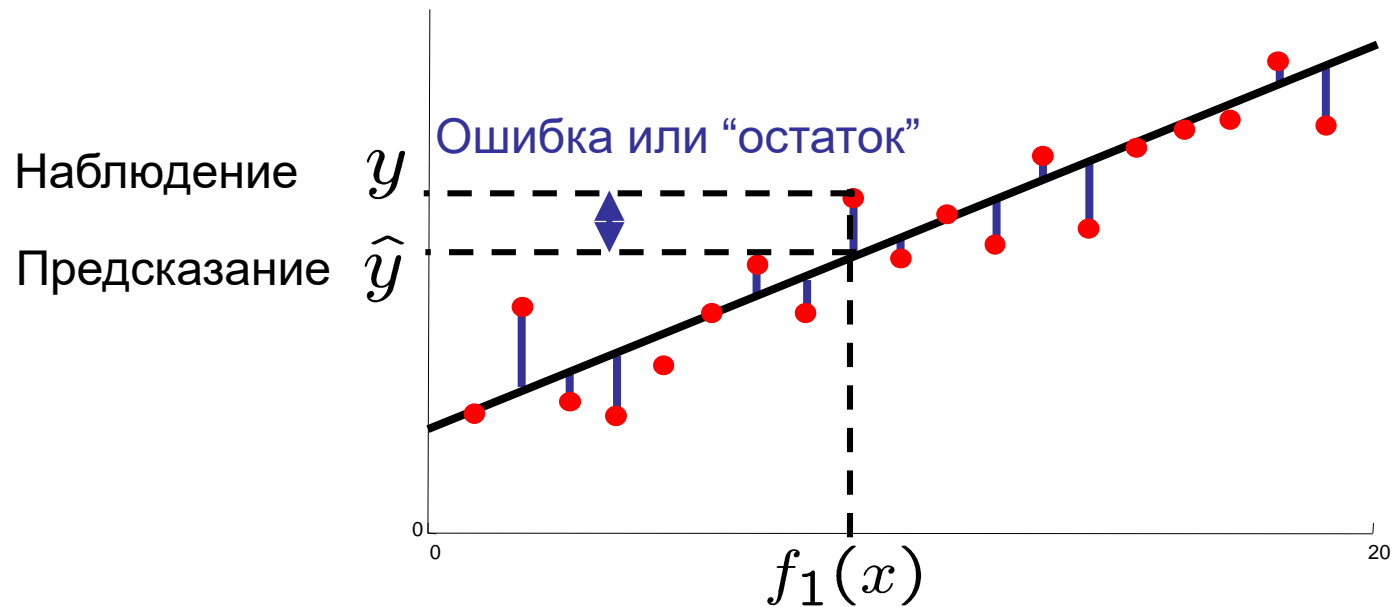


Предсказание:

$$\hat{y}_i = w_0 + w_1 f_1(x) + w_2 f_2(x)$$

# Оптимизация: наименьшие квадраты

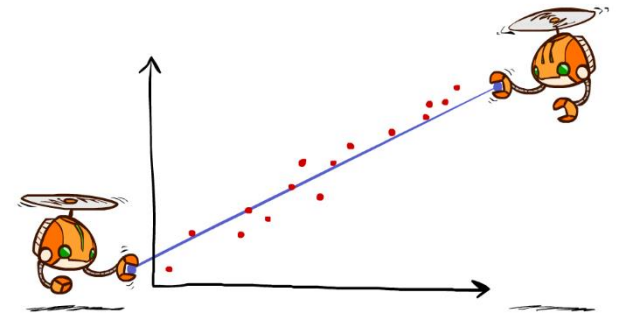
$$\text{total error} = \sum_i (y_i - \hat{y}_i)^2 = \sum_i \left( y_i - \sum_k w_k f_k(x_i) \right)^2$$



# Минимизация ошибки

Представьте, что у нас есть только одна точка  $x$  с признаками  $f(x)$ , целевым значением  $y$  и весами  $w$ :

$$\begin{aligned}\text{error}(w) &= \frac{1}{2} \left( y - \sum_k w_k f_k(x) \right)^2 \\ \frac{\partial \text{error}(w)}{\partial w_m} &= - \left( y - \sum_k w_k f_k(x) \right) f_m(x) \\ w_m &\leftarrow w_m + \alpha \left( y - \sum_k w_k f_k(x) \right) f_m(x)\end{aligned}$$



Объяснение q-обучения с аппроксимацией:


$$w_m \leftarrow w_m + \alpha \left[ r + \gamma \max_a Q(s', a') - Q(s, a) \right] f_m(s, a)$$

“цель”

“предсказание”

# Глубокое Q-обучение

Будем осуществлять аппроксимацию Q- функции в виде:

$$Q^*(s, a) \approx Q(s, a; \theta)$$


параметры (веса)

Если в качестве аппроксиматора будет использоваться глубокая нейронная сеть, то будем говорить **о глубоком q-обучении**. *Сеть, выполняющую указанную аппроксимацию, называют **Q-сетью**.*



# Глубокое Q-обучение

Мы хотим найти Q-функцию, удовлетворяющую уравнению Беллмана:

$$Q^*(s, a) = \mathbb{E}_{s' \sim \mathcal{E}} \left[ r + \gamma \max_{a'} Q^*(s', a') | s, a \right]$$

## Прямое распространение

Q-сеть обучается путем минимизации последовательности потерь  $L_i(\vartheta_i)$ , которые изменяются на каждой итерации  $i$ :

$$L_i(\theta_i) = \mathbb{E}_{s, a \sim p(\cdot)} [(y_i - Q(s, a; \theta_i))^2]$$

где

$$y_i = \mathbb{E}_{s' \sim \mathcal{E}} \left[ r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) | s, a \right]$$

предсказанное q-значение на итерации  $i$ , а  $p(s, a)$  - распределение пар  $(s, a)$ . Параметры предыдущей итерации  $\vartheta_{i-1}$  фиксируются. Также отметим, что предсказываемое значение зависит от весов

## Обратное распространение

Градиент обновления (градиент функции потерь по параметрам  $\vartheta$ ):

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s, a \sim p(\cdot); s' \sim \mathcal{E}} \left[ \left( r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) - Q(s, a; \theta_i) \right) \nabla_{\theta_i} Q(s, a; \theta_i) \right]$$

# Глубокое Q-обучение

Мы хотим найти Q-функцию, удовлетворяющую уравнению Беллмана:

$$Q^*(s, a) = \mathbb{E}_{s' \sim \mathcal{E}} \left[ r + \gamma \max_{a'} Q^*(s', a') | s, a \right]$$

## Прямое распространение

Функция потерь:

$$L_i(\theta_i) = \mathbb{E}_{s, a \sim \rho(\cdot)} \left[ (y_i - Q(s, a; \theta_i))^2 \right]$$

где

$$y_i = \mathbb{E}_{s' \sim \mathcal{E}} \left[ r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) | s, a \right]$$

Итеративно приближаем Q-функцию к целевому значению ( $y_i$ ). Это будет возможно, если Q-функция будет оптимальной  $Q^*$

## Обратное распространение

Градиент обновления (градиент функции потерь по параметрам  $\vartheta$ ):

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s, a \sim \rho(\cdot); s' \sim \mathcal{E}} \left[ \left( r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) - Q(s, a; \theta_i) \right) \nabla_{\theta_i} Q(s, a; \theta_i) \right]$$

Вместо вычисления ожиданий в вышеупомянутом градиенте оптимизируют функцию потерь путем стохастического градиентного спуска. Параметры при этом обновляются на каждом временном шаге пропорционально градиенту обновления на основе выборок из  $\rho(s, a)$  и эмулятора среды  $\mathcal{E}$ . Т.о. мы приходим к знакомому алгоритму Q-обучения

# Игры Atari с использованием DQN

*Deep Q Network (DQN)* — один из популярных алгоритмов **глубокого обучения с подкреплением (DRL)**. Он был предложен исследователями из компании DeepMind, принадлежащей Google, и достиг результатов человеческого уровня в любой игре Atari, получая на входе только экраны игры.



**Цель:** завершить игру с наибольшим количеством очков

**Состояние:** значения пикселей экранных форм игры

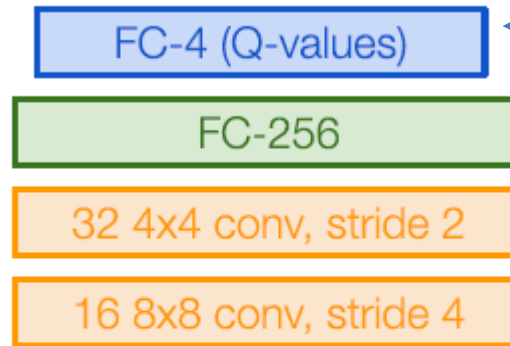
**Действие:** управление игрой, например: влево, вправо, вверх, вниз

**Награда:** увеличивается / уменьшается на каждом временном шаге

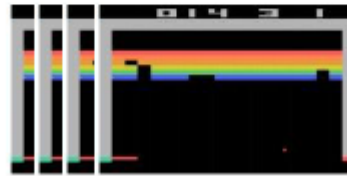
# Архитектура Q-сети

$Q(s, a, \Theta)$ : нейронная сеть с весами  $\Theta$

Один шаг прямого распространения для вычисления Q-функции для всех действий из текущего состояния → эффективно!



FC слой обеспечивает 4-d выход (если 4 действия), соответствуют  $Q(s, a_1)$ ,  $Q(s, a_2)$ ,  $Q(s, a_3)$ ,  $Q(s, a_4)$



Вход: состояние  $s_t$

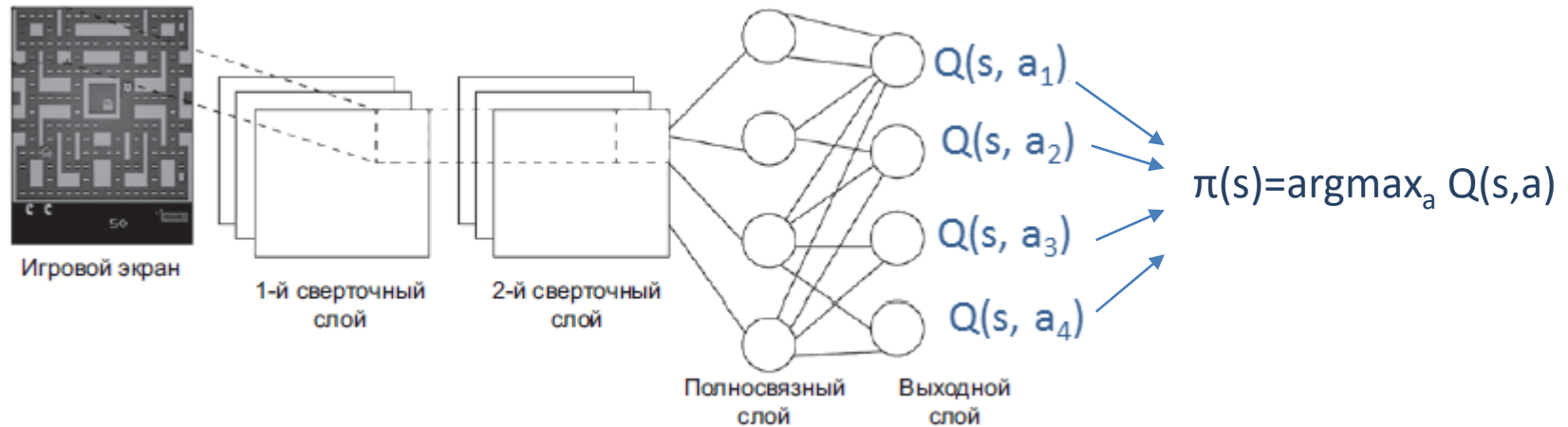
Число действий 4-18 в зависимости от Atari игры

Текущее состояние  $s_t$  : стек из 4 фреймов  $84 \times 84 \times 4$

(после предварительного преобразования  $210 \times 160$  RGB -> полутоновое, прореживания и сокращения)

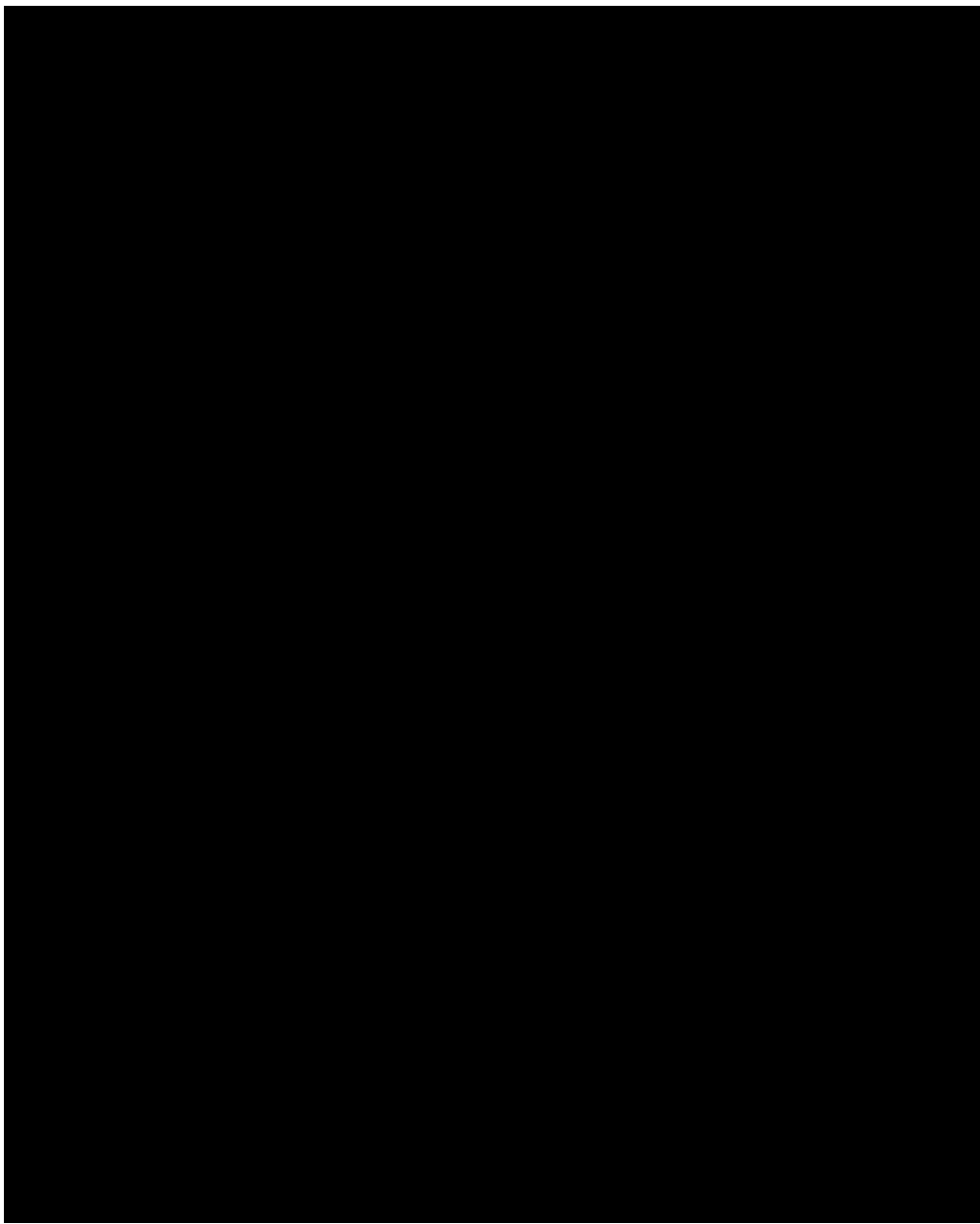
# Архитектура Q-сети

Архитектура DQN показана на следующем рисунке. Мы передаем ей игровой экран, а она выдает значения  $Q$  для всех действий в данном игровом состоянии





<https://www.youtube.com/watch?v=V1eYniJ0Rnk>



<https://www.youtube.com/watch?v=V1eYniJ0Rnk>