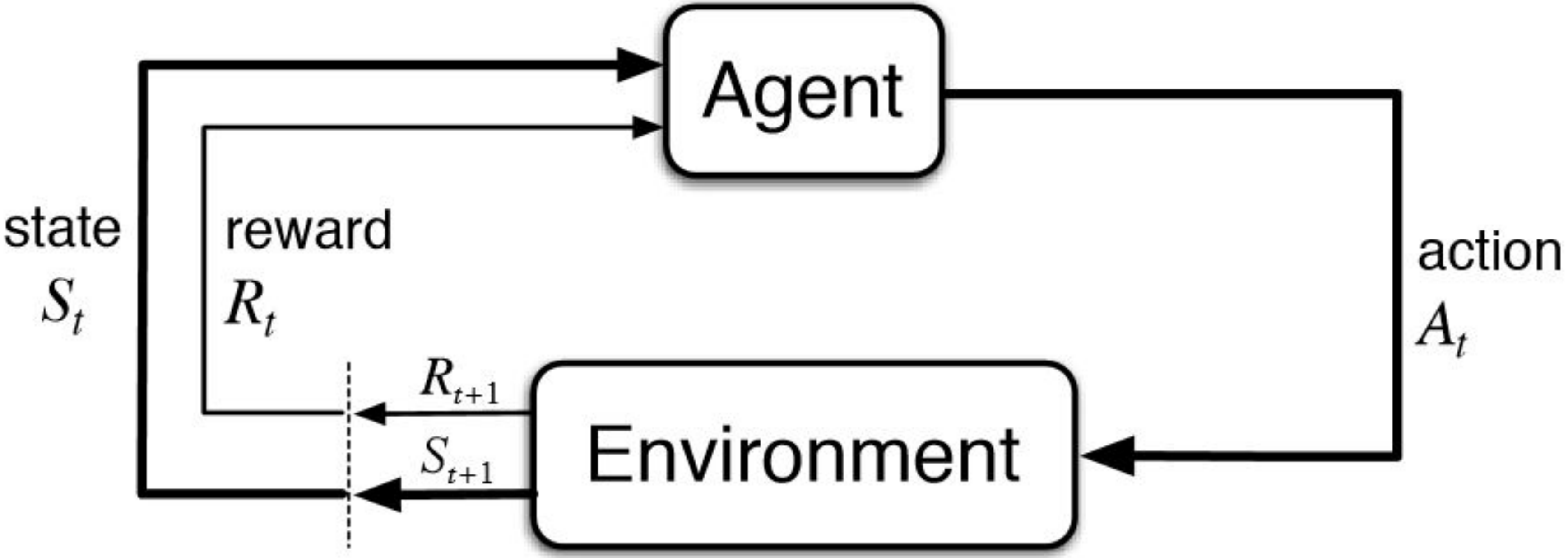
RL 3

Q-learning + всякие ТРЮКИ

Сори за уродскую презу

В предыдущих сериях…



В предыдущих сериях…

$$G_t = \sum_{t'=t}^{T} \gamma^{(t'-t)} r_{t'}$$

$$Q^\pi(s,a) = E_\pi[G_t | s_t = s, a_t = a]$$

$$V^\pi(s) = E_\pi[G_t | s_t = s]$$

$Q^*(s,a)$ – q функция с оптимальной политикой

$V^*(s)$ – v функция с оптимальной политикой

Value iteration

Инициализируем *V(s)* случайно

Повторяем, пока *V(s)* не сойдется:

Для всех состояний *s*:

Для всех действий *a*:

$$Q(s,a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a] = \sum_{s_{next}} \boxed{P^a_{ss_{next}} \cdot (r(s,a)} + \gamma V(s_{next}))$$

$$V(s) = \max_a Q(s,a)$$

Итоговая политика:

$$\pi(s) = \arg\max_a Q(s,a)$$

Policy iteration

1. **Policy Evaluation (считаем value-функцию для текущей политики)**
   Повторяем, пока *V(s)* не сойдется:
   Для всех состояний *s*:
   $$V(s) = \sum_{s_{next}} \boxed{P_{ss_{next}}^{a=\pi(s)}(r(s, a = \pi(s))} + \gamma V(s_{next}))$$
2. **Policy Improvement (обновляем политику, чтобы максимизировать value-функцию)**
   Обновляем политику:
   $$Q(s, a) = \sum_{s_{next}} \boxed{P_{ss_{next}}^{a}(r(s, a)} + \gamma V(s_{next}))$$
   $$\pi(s) = \arg\max_a Q(s, a)$$
3. **Переходим к шагу 1, пока политика не сойдется**

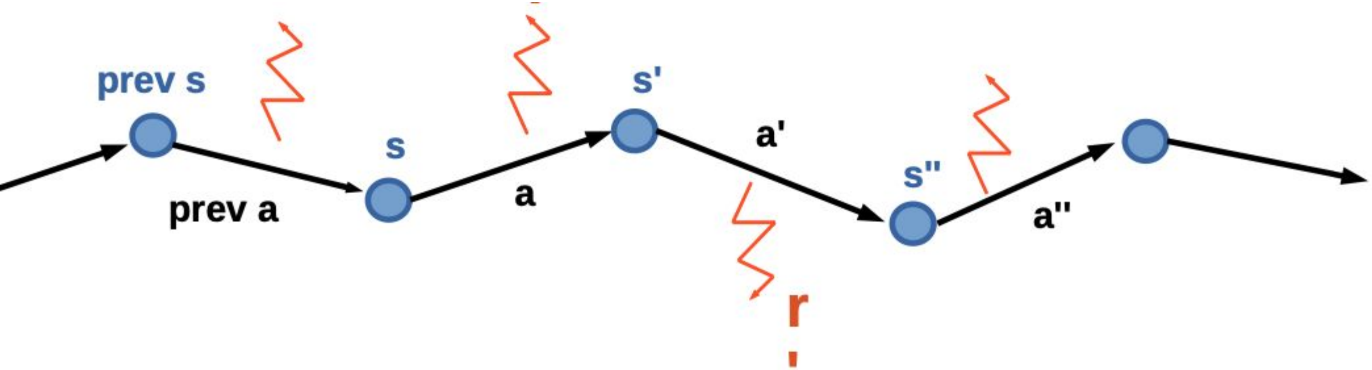
Model-free reinforcement learning:

Мы не знаем

$P(s', r \mid s, a)$

F

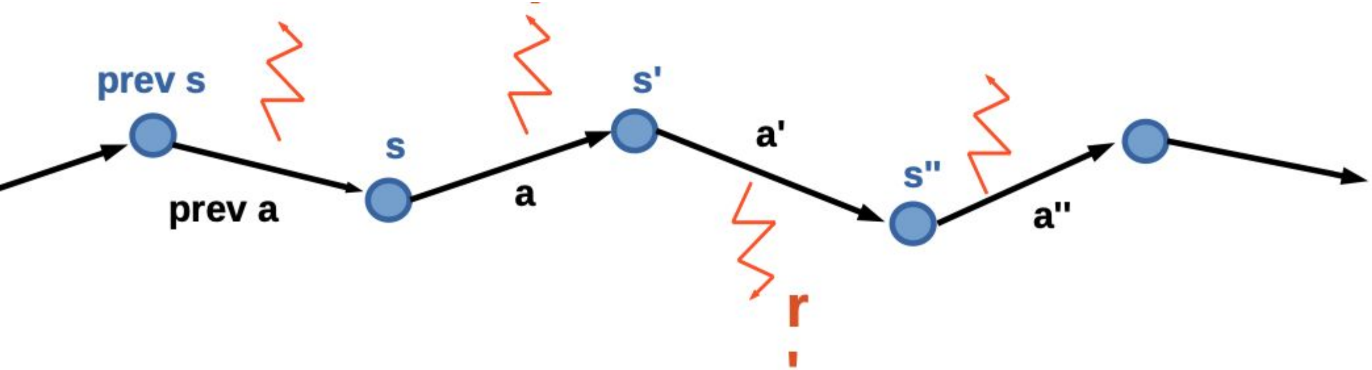
Обучение по траекториям



Model-based: вы знаете P(s', r | s, a)
- можно применять динамическое программирование
- можно планировать заранее

Model-free: вы можете пробовать траектории
- можно пробовать
- страховки нет

Траектории



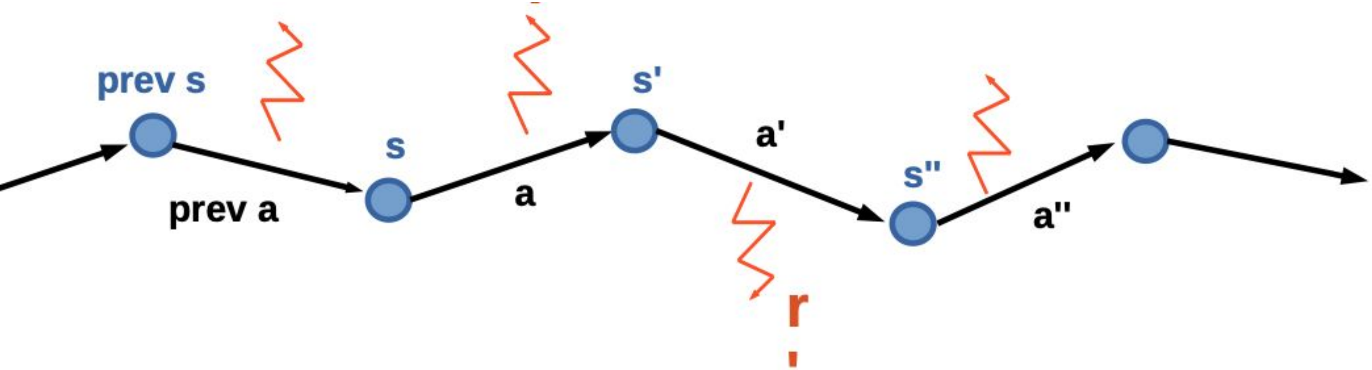
- Траектория - это последовательность
  - Состояний (s)
  - Действия (a)
  - Вознаграждения (r)

- Мы можем только выбирать траектории

Что учить?

V(s) или <u>Q(s, q)</u>

Траектории



- Траектория - это последовательность
  - Состояний (s)
  - Действия (a)
  - Вознаграждения (r)

- Мы можем только выбирать траектории

Что учить?

V(s) или Q(s, q)

V(s) – не работает без P(s’ | s, a)

Monte Carlo

- Получим все траектории, содержащие конкретные (s,a)
- Оценим G(s,a) для каждой траектории
- Усредним и получим матожидание

Monte Carlo

- Получим все траектории, содержащие конкретные (s,a)
- Оценим G(s,a) для каждой траектории
- Усредним и получим матожидание

требует много сессий

Temporal difference

- Вспомним, что мы можем улучшать Q(s, a) итеративно

$$Q(s_t, a_t) \leftarrow \mathop{E}_{r_t, s_{t+1}} r_t + \gamma \cdot max_{a'} Q(s_{t+1}, a')$$

Temporal difference

- Вспомним, что мы можем улучшать Q(s, a) итеративно

$$Q(s_t, a_t) \leftarrow \mathop{E}_{r_t, s_{t+1}} r_t + \gamma \cdot max_{a'} Q(s_{t+1}, a')$$

Все вопросы к Максиму Казадаеву БПМИ202

Temporal difference

- Вспомним, что мы можем улучшать Q(s, a) итеративно

$$Q(s_t, a_t) \leftarrow \mathop{E}_{r_t, s_{t+1}} r_t + \gamma \cdot max_{a'} Q(s_{t+1}, a')$$

А вот этого мы не знаем

Temporal difference

- Заменим матожидание на среднее

$$\underset{r_t, s_{t+1}}{E} r_t + \gamma \cdot max_{a'} Q(s_{t+1}, a') \approx \frac{1}{N} \sum_i r_i + \gamma \cdot max_{a'} Q(s_i^{next}, a')$$

Temporal difference

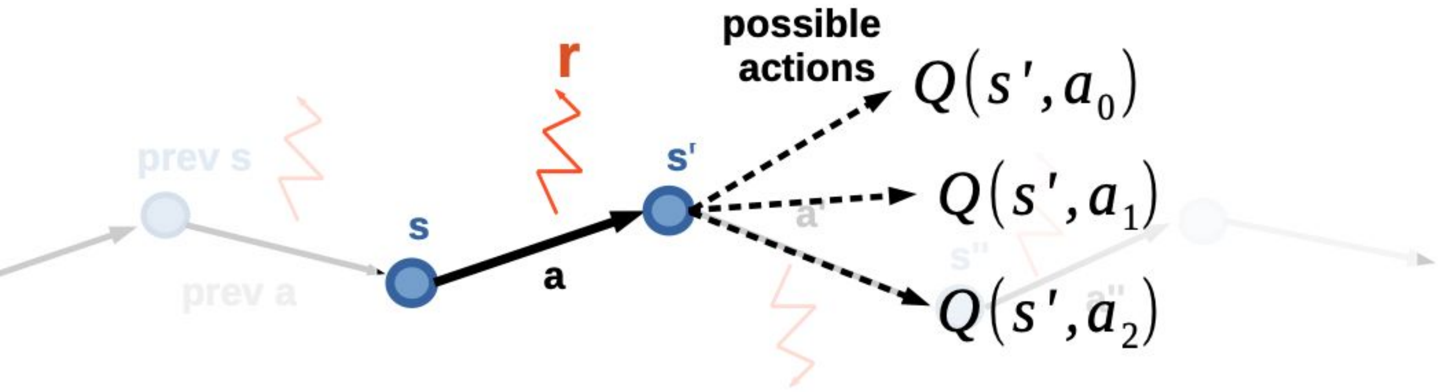
- Заменим матожидание на среднее

$$\underset{r_t, s_{t+1}}{E} r_t + \gamma \cdot max_{a'} Q(s_{t+1}, a') \approx \frac{1}{N} \sum_i r_i + \gamma \cdot max_{a'} Q(s_i^{next}, a')$$

- Используем скользящее среднее с только с одним объектом

$$Q(s_t, a_t) \leftarrow \alpha \cdot (r_t + \gamma \cdot max_{a'} Q(s_{t+1}, a')) + (1 - \alpha) Q(s_t, a_t)$$

Q-learning

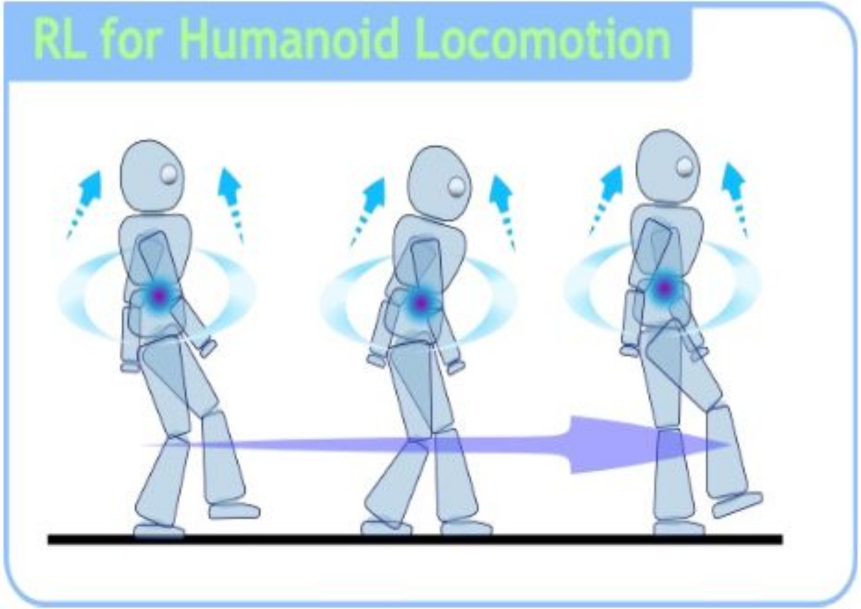


Инициализируем Q(s, a) нулями

- В цикле:
  - Берем <s, a, r, s’> из среды
  - Считаем $\hat{Q}(s,a)=r(s,a)+\gamma \max_{a_i} Q(s',a_i)$
  - Обновляем $Q(s,a) \leftarrow \alpha \cdot \hat{Q}(s,a)+(1-\alpha)Q(s,a)$

Что может пойти не так?

Допустим наш робот учится ходить



Инициализируем Q(s, a) нулями

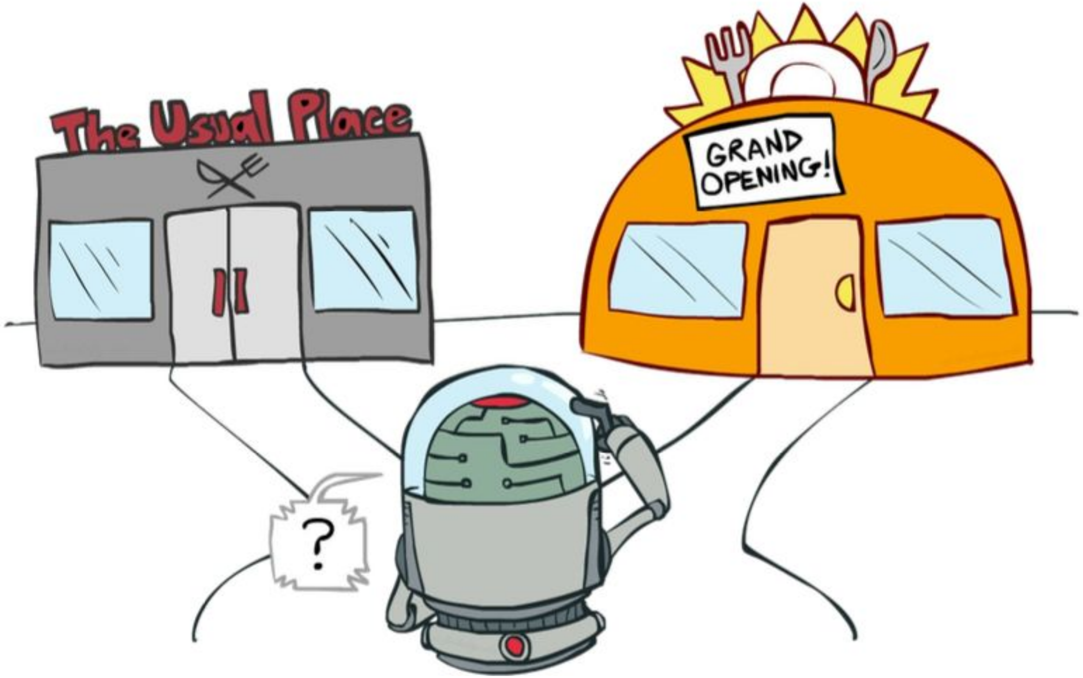
наш робот выбирает argmax Q(s, a)

Теперь он никогда не научится ходить прямо =(

Exploration Vs Exploitation

- ε - greedy

  - С вероятностью ε совершать случайное действие



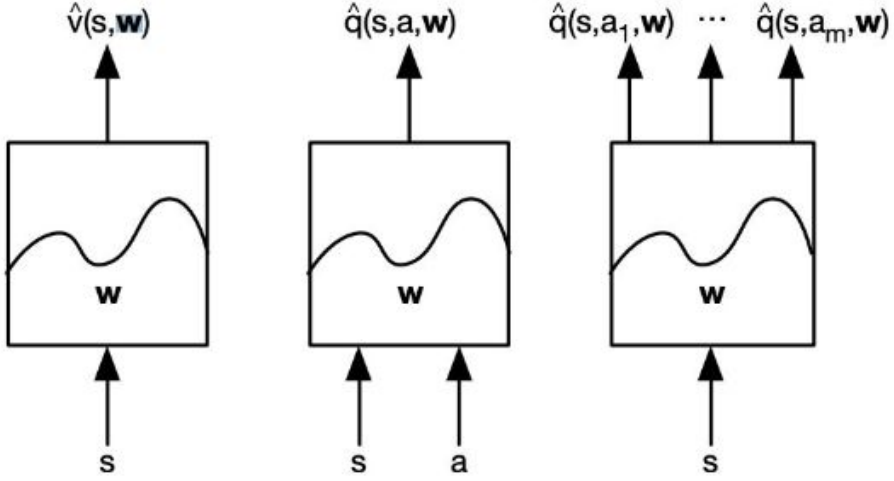
Проблемы с количеством состояний

$$|S| = 2^{210 \cdot 160 \cdot 8 \cdot 3}$$

Хотим приблизить Q(s, a) какой-нибудь функцией

$$argmin_{w,b}(Q(s_t,a_t)-[r_t+\gamma\cdot max_{a'}Q(s_{t+1},a')])^2$$

- По факту – задача регресии
- Можем использовать что угодно, главное – оценить Q



Q-learning

Раскроем скобки:

$$Q(s_t, a_t) \longleftarrow Q(s_t, a_t) + \alpha(r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t))$$

Q-learning как MSE:

$$L = (r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t))^2$$

$$\nabla L = 2 \cdot (r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t))$$

Q-learning

Раскроем скобки:

$$Q(s_t, a_t) \longleftarrow Q(s_t, a_t) + \alpha \boxed{(r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t))}$$

Q-learning как MSE:

$$L = (r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t))^2$$

$$\nabla L = 2 \cdot \boxed{(r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t))}$$

Q-learning

Раскроем скобки:

$$Q(s_t, a_t) \longleftarrow Q(s_t, a_t) + \alpha(r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t))$$

Q-learning как MSE:

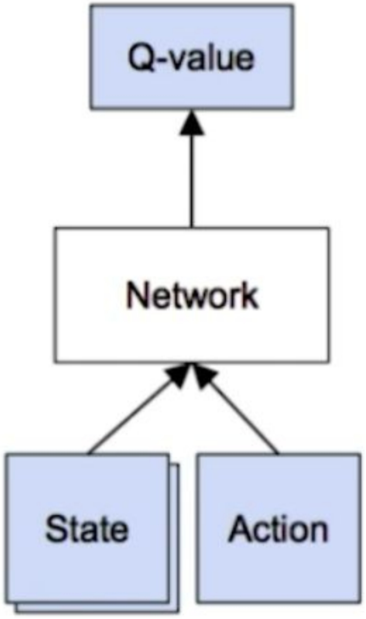
$$L = (r_t + \gamma \max_{a'} \boxed{Q(s_{t+1}, a')} - Q(s_t, a_t))^2$$

Const

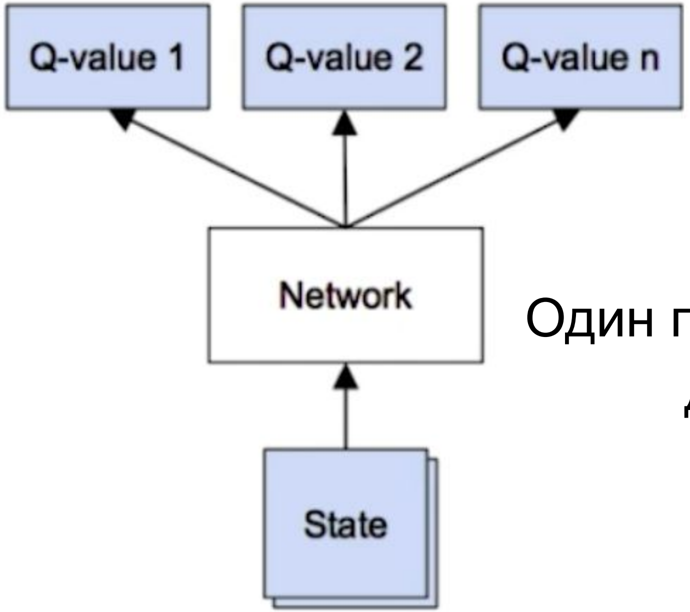
$$\nabla L = 2 \cdot (r_t + \gamma \max_{a'} \boxed{Q(s_{t+1}, a')} - Q(s_t, a_t))$$

Возможные архитектуры

Непрерывный контроль или огромное количество действий



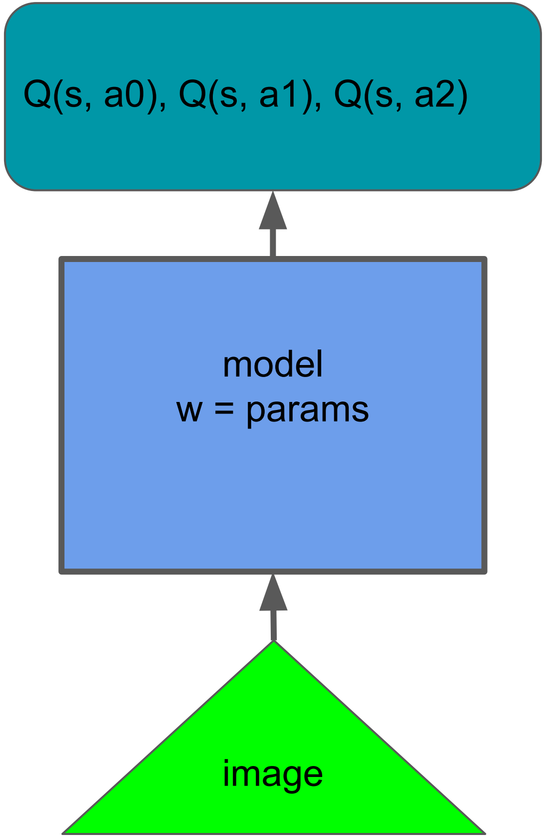
Given **(s,a)**
Predict Q(s,a)



Один проход для всех действий

Given **s** predict all q-values
Q(s,a0), Q(s,a1), Q(s,a2)

Почти Q-learning



target:

$$\hat{Q}(s_t, a_t) = r + \gamma \cdot max_{a'} Q(s_{t+1}, a')$$

loss:

$$L = (Q(s_t, a_t) - [r + \gamma \cdot max_{a'} \underbrace{\boxed{Q(s_{t+1}, a')}}_{\text{Const}}])^2$$

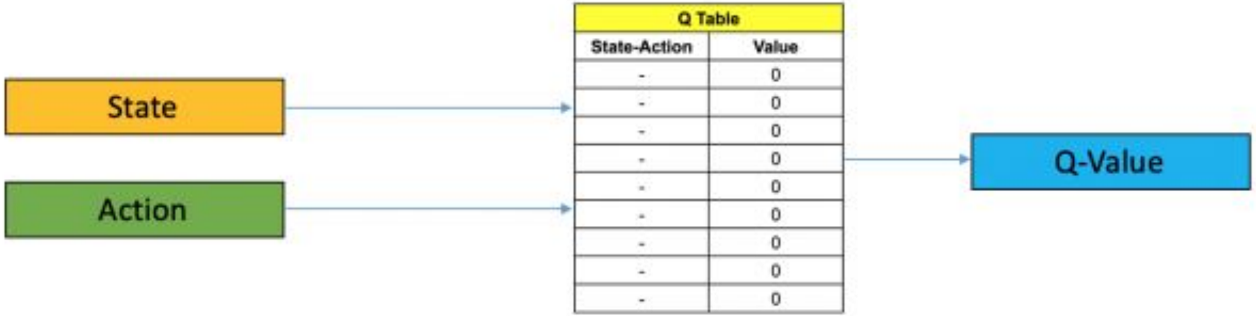
$$L = (Q(s_t, a_t) - \underbrace{\hat{Q}(s_t, a_t)}_{\text{consider const}})^2$$

gradient step:

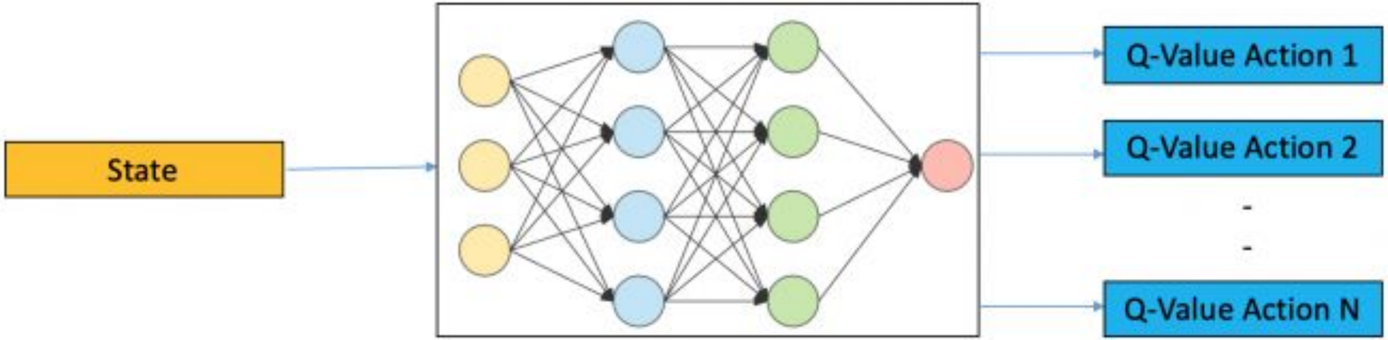
$$w_{t+1} = w_t - \alpha \cdot \frac{\delta L}{\delta w}$$

Q-learning:

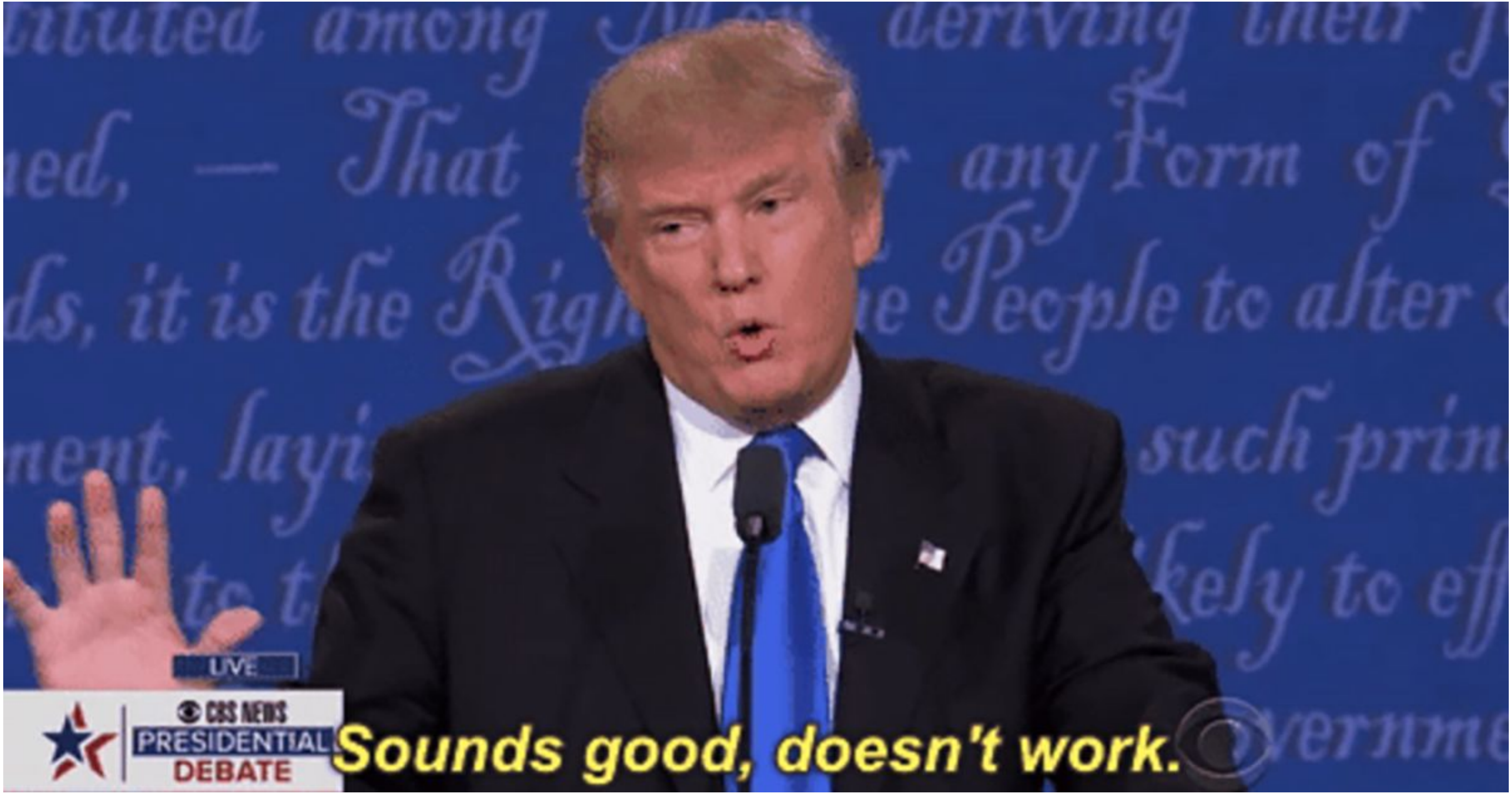
$$\hat{Q}(s_t, a_t) = r + \gamma \cdot max_{a'} Q(s_{t+1}, a')$$



Q Learning

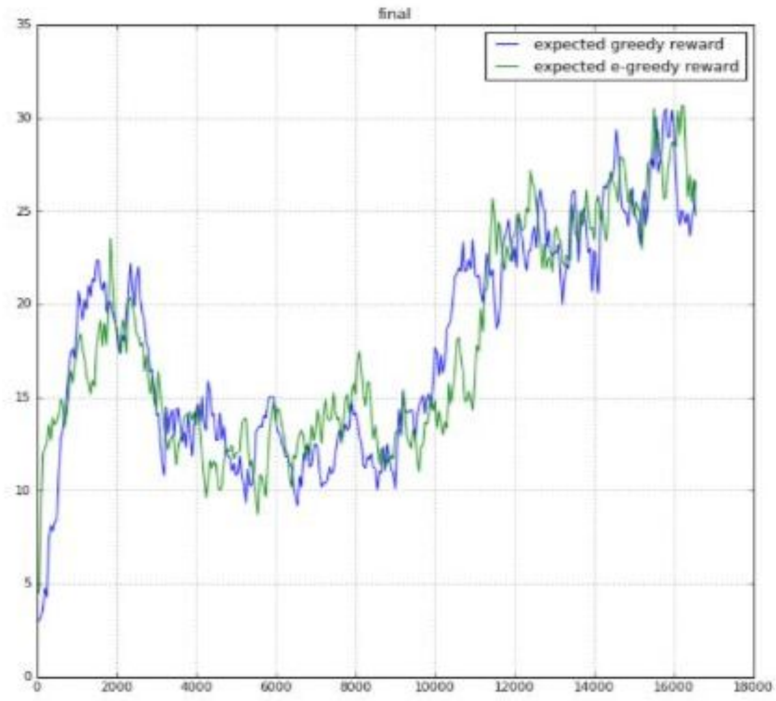


Deep Q Learning


LIVE
CBS NEWS
PRESIDENTIAL
DEBATE
Sounds good, doesn't work.

Проблема – данные всегда упорядочены

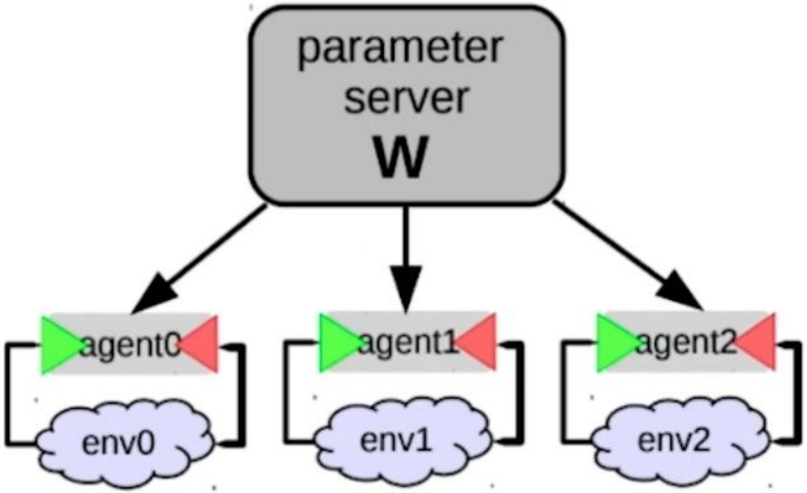
- Обучающие данные не независимые одинаково распределенные случайные величины (н.о.р.с.в)
- Модель забывает часть среды, которую не посещает
- Падает темп обучения



Multiple agent trick

Идея: запускаем несколько агентов с одинаковыми параметрами

- Скорее всего, они будут исследовать различные части окружающей среды,
- Более стабильное обучение
- Требуют много действий

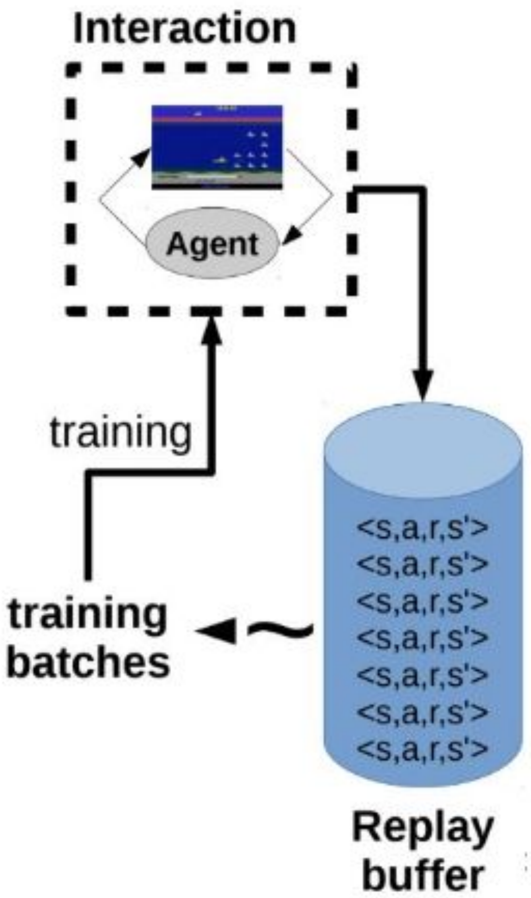



fedex.com
1.800.GoFedEx
TOP LIFT ONLY
SEDALCO

Experience replay

Идея: храним несколько прошлых взаимодействий <s,a,r,s'>

Обучаем на случайных подвыборках



- Ближе к н.о.р.с.в
- Отбрасываем старые взаимодействия, как полученные более слабой стратегией

налево или направо?

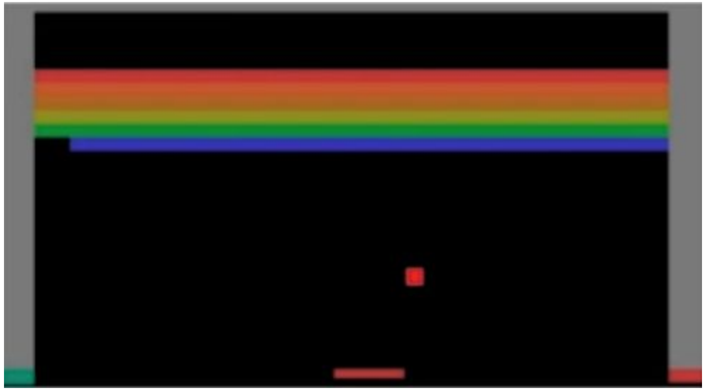
N-gram trick

Идея:

$$s_t \neq o(s_t)$$

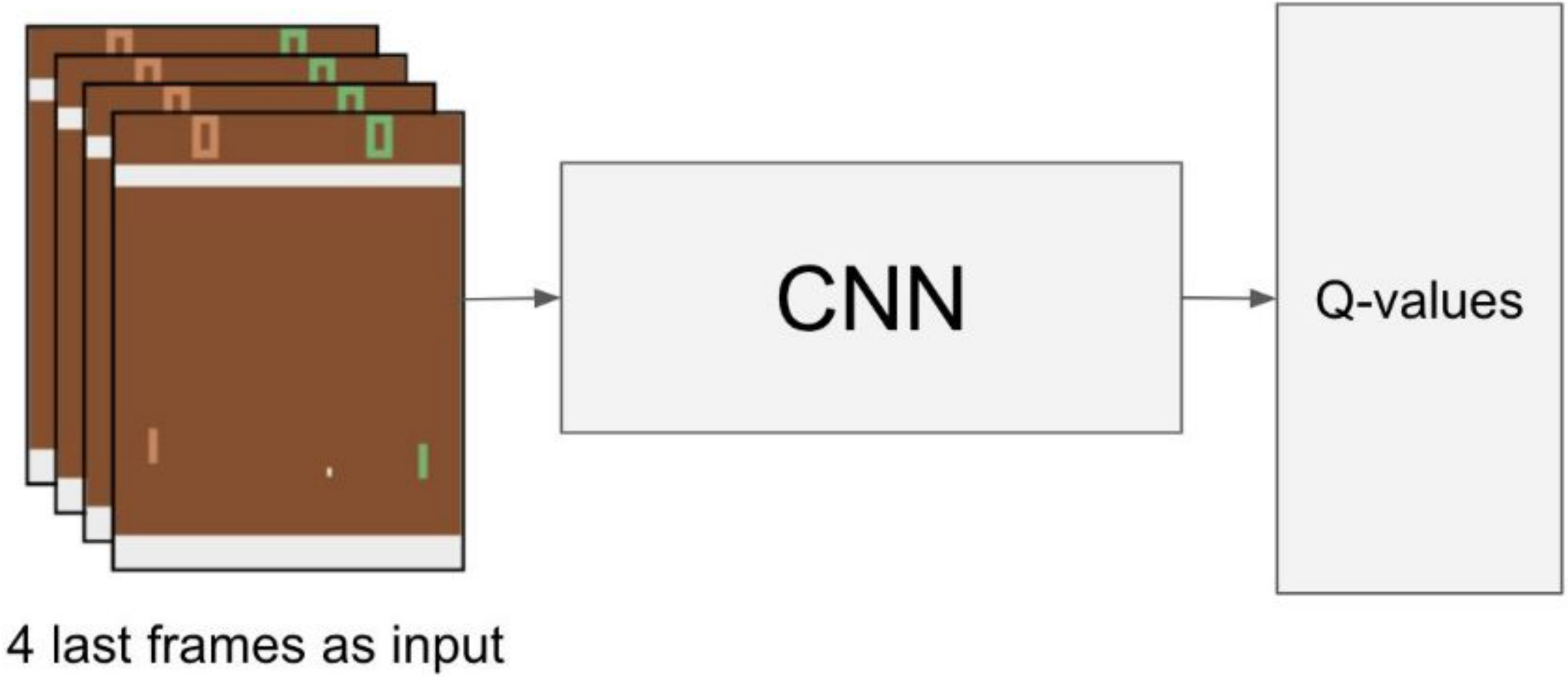
$$s_t \approx (o(s_{t-n}), a_{t-n}, \ldots, o(s_{t-1}), a_{t-1}, o(s_t))$$

например направление движения мяча



Один кадр

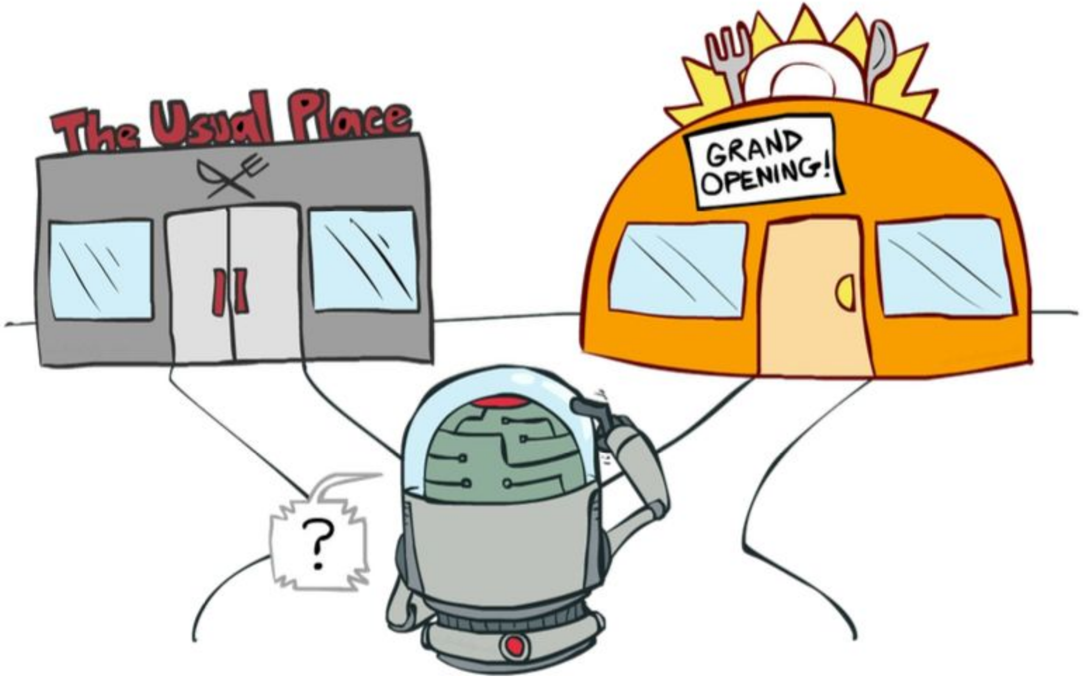
Несколько кадров


CNN
Q-values
4 last frames as input

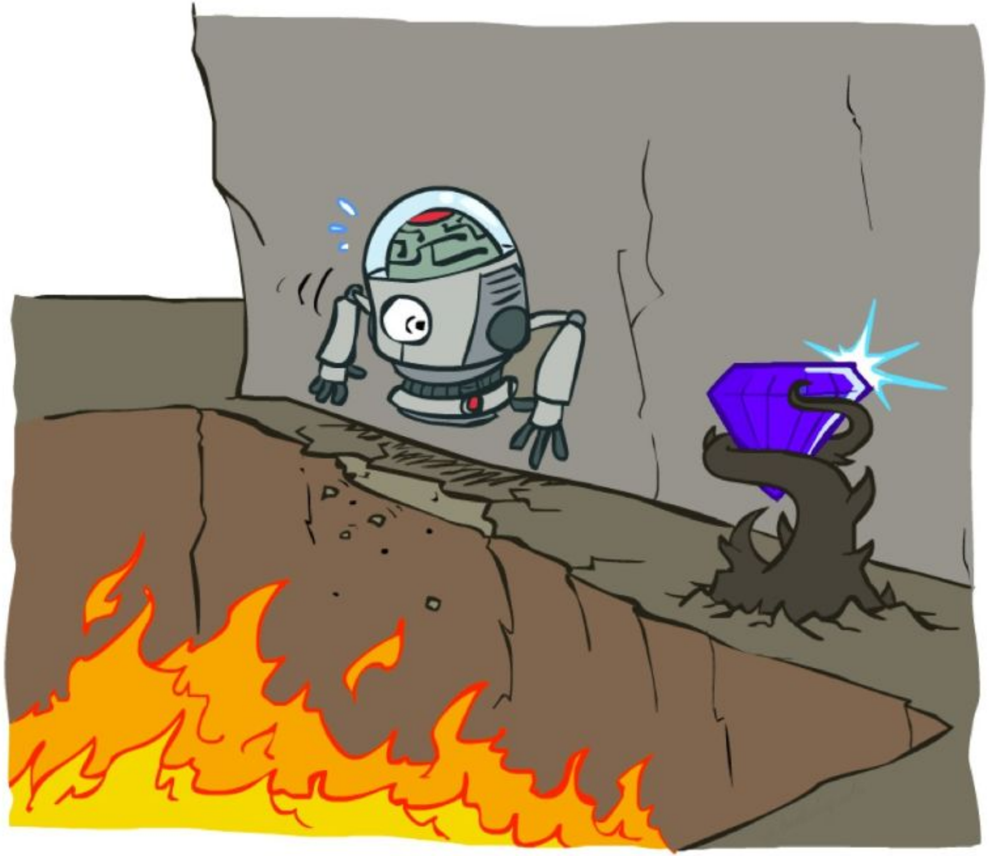
Exploration Vs Exploitation

- ε - greedy

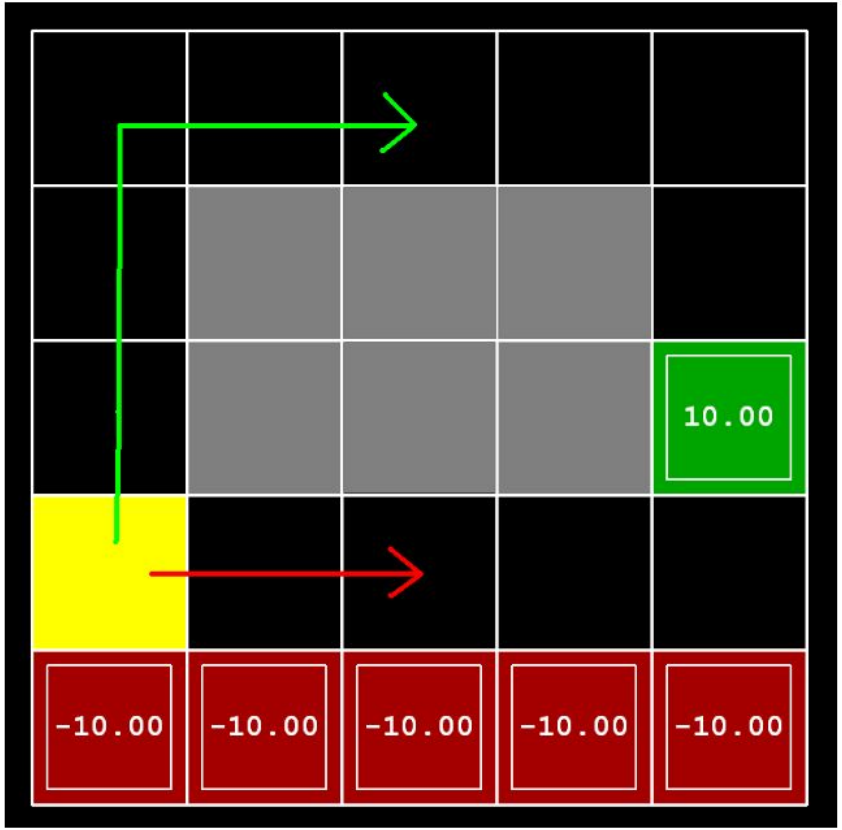
  - С вероятностью ε совершать случайное действие



Cliff world



Cliff world



Условия:

- Q-learning

$$\gamma = 0.99 \quad \epsilon = 0.1$$

Что сделает Q-learning?

Q-learning vs SARSA

Правило обновления (из равенства Беллмана)

$$Q(s_t, a_t) \leftarrow \alpha \cdot \hat{Q}(s_t, a_t) + (1-\alpha) Q(s_t, a_t)$$

Q-learning

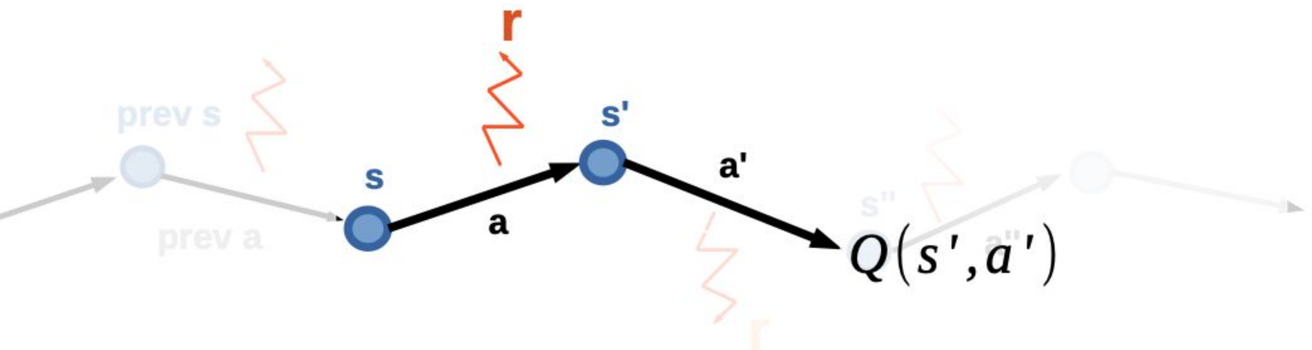
“лучшее Q(s, a)”

$$\hat{Q}(s,a) = r(s,a) + \gamma \cdot \max_{a'} Q(s',a')$$

Sarsa

$$\hat{Q}(s,a) = r(s,a) + \gamma \cdot \underset{a' \sim \pi(a'|s')}{E} Q(s',a')$$

Sarsa



Инициализируем Q(s, a) нулями

- В цикле:
  - Берем <s, a, r, s’, a’> из среды
  - Считаем $\hat{Q}(s,a)=r(s,a)+\gamma Q(s',a')$
  - Обновляем $Q(s,a)\leftarrow\alpha\cdot\hat{Q}(s,a)+(1-\alpha)Q(s,a)$