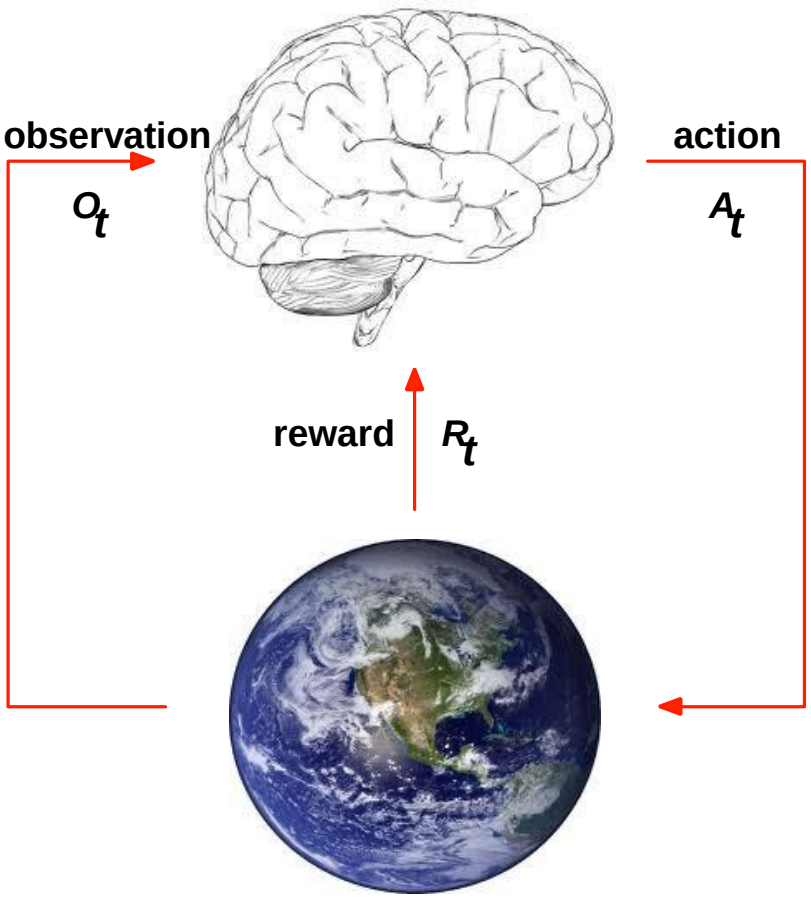
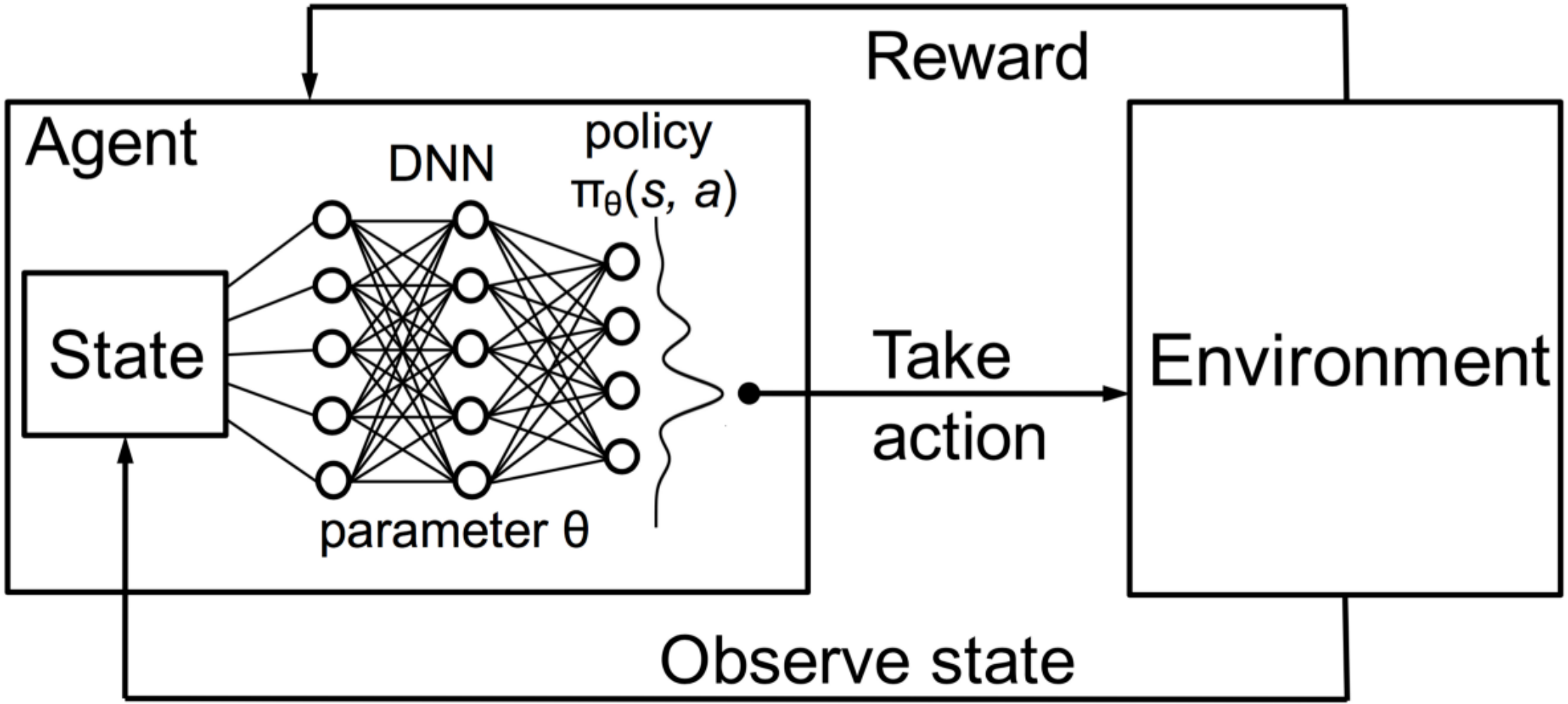
RL: DEEP RL

План

- ❑ Deep Q learning
- ❑ Experience replay
- ❑ Double Q-learning
- ❑ Dueling Network

Взаимодействие среды и агента. MDP




Reward
Agent
DNN
policy
$\pi_\theta(s, a)$
State
Take
action
Environment
parameter θ
Observe state

Базовые определения

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = r_{t+1} + \gamma(r_{t+2} + \gamma r_{t+3} + \cdots) = r_{t+1} + \gamma G_{t+1}$$

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

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

Рекуррентные формулы

$$Q^\pi(s) = E_{s_{t+1}}\left[r_t + \gamma V^\pi(s_{t+1})\right]$$

$$Q^\pi(s) = E_{s_{t+1}, a_{t+1} \sim \pi}[r_t + Q^\pi(s_{t+1}, a_{t+1})]$$

Оптимальная политика

Для всех π, s, a: $Q^{\pi^*}(s,a) \geq Q^{\pi}(s,a)$

$$\pi^*(s) = argmax_a Q^{\pi^*}(s,a)$$

Уравнение Беллмана

$$Q^*(s_t, a) = E_{s_{t+1}}\left[r_t + \max_{a'} Q^*(s_{t+1}, a')\right]$$

Q - learning

Шаг обучения

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

Q-learning как минимизация MSE

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

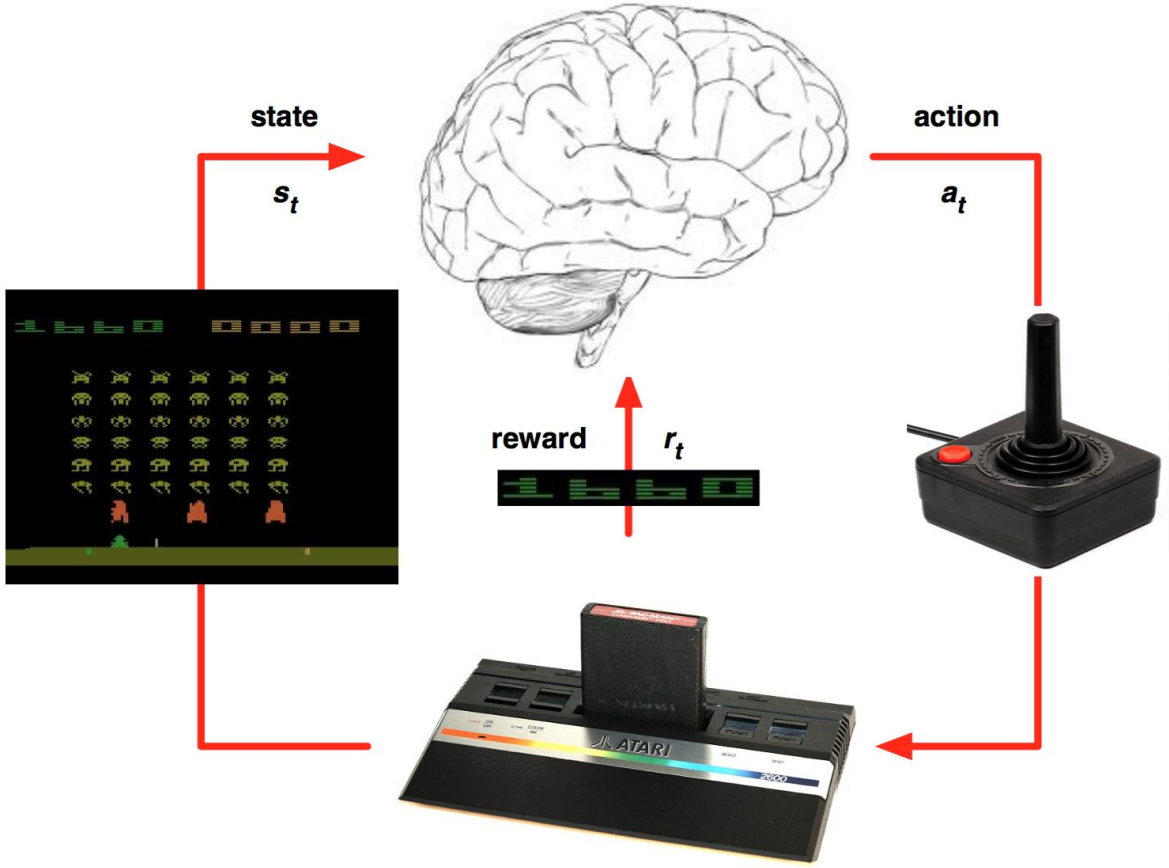
$$\nabla L = 2 \left(r_t + \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right)$$

Какая проблема?

Реальный мир

Сколько приблизительно состояний?

$$S = 2^{210*160*8*3}$$



Проблема

- Пространство состояний может быть очень большим или непрерывным

- Два решения:
  - Бинаризация пространства состояний
  - Функциональная аппроксимация агента

- Что из этого лучше подойдет для Atari?

От таблицы к аппроксимации

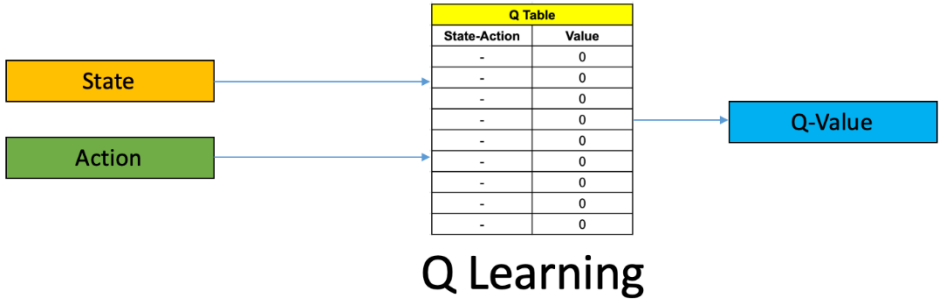
- Для таблицы:
  - Для всех состояний и действий мы должны запомнить $Q(s, a)$

- Аппроксимация:
  - Аппроксимация $Q(s, a)$ некоторой функцией
  - Например линейной функцией

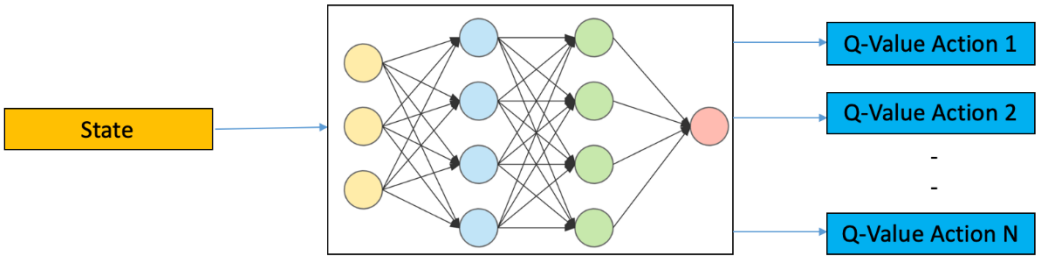
$$argmin_{w,b}\left(r_t + \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)\right)^2$$

Вопрос: что мы используем, классификация или регрессию?

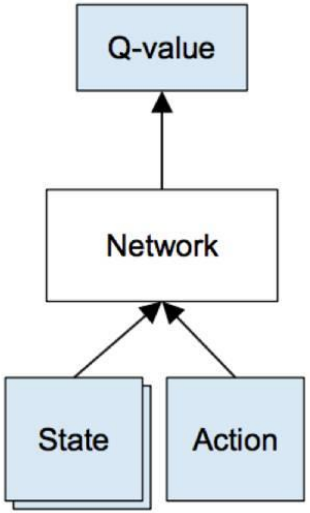
Возможные архитектуры

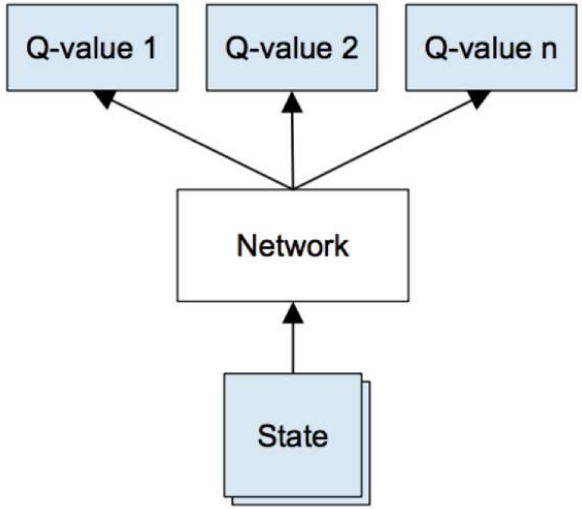


Q Learning



Deep Q Learning

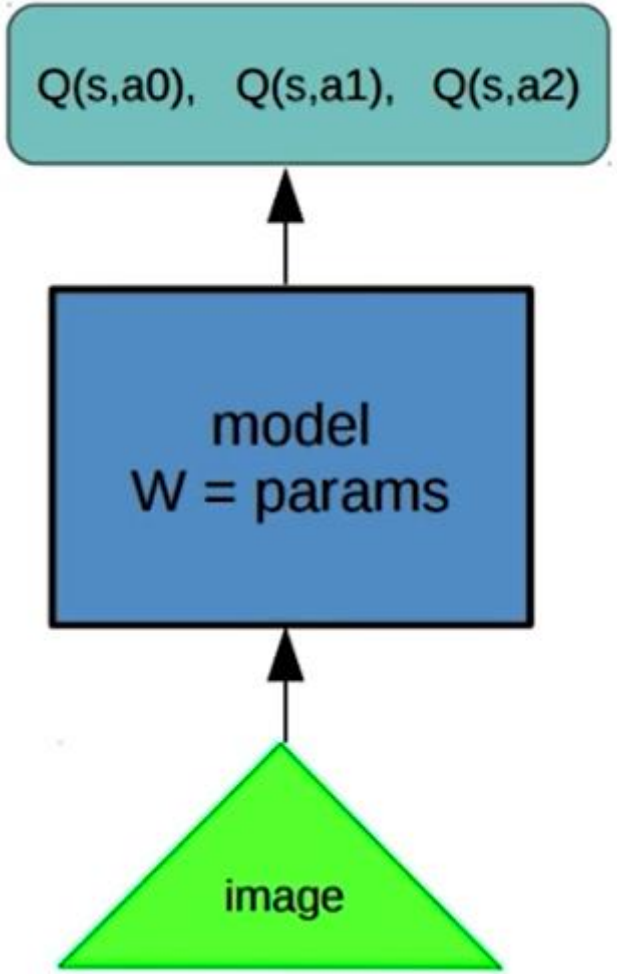




Аппроксимация Q-learning

- Q-values
  - $\hat{Q}(s_t, a_t) = r_t + \max_{a'} Q(s_{t+1}, a')$
- Целевая функция:
  - $\mathrm{L} = \left(\mathrm{Q}(s_t, a_t) - \left[r_t + \max_{a'} Q(s_{t+1}, a')\right]\right)^2$
- Шаг градиента:

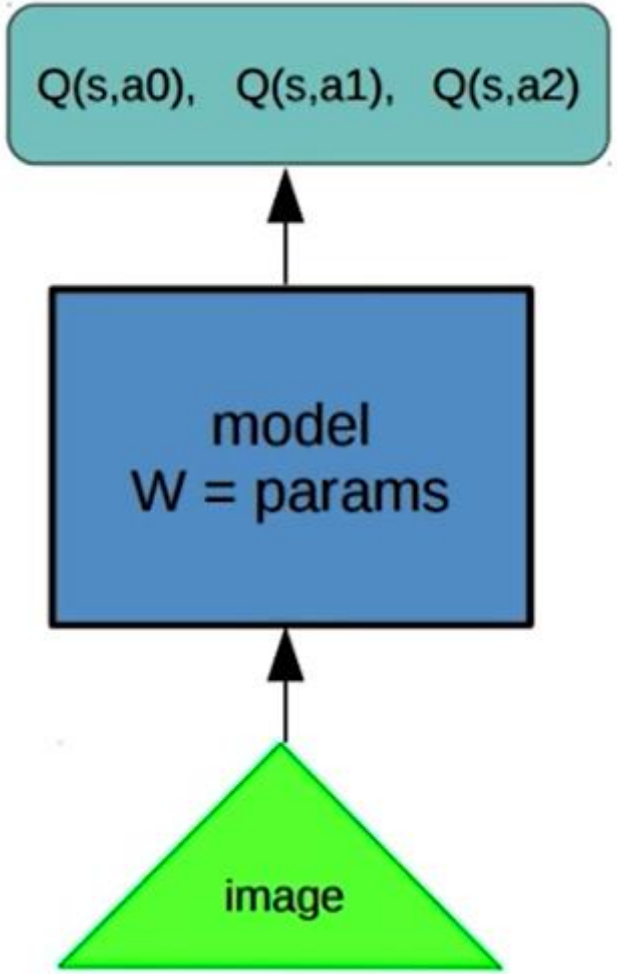
$$w_{t+1} = w_t - \alpha \frac{\partial L}{\partial w}$$



Аппроксимация Q-learning

- Целевая функция:
  - $\mathrm{L} = \left(\mathrm{Q}(s_t, a_t) - \hat{Q}(s_t, a_t)\right)^2$
- Q-learning
  - $\hat{Q}(s_t, a_t) = r_t + \gamma \max_{a'} Q(s_{t+1}, a')$
- SARSA
  - $\hat{Q}(s_t, a_t) = r_t + \gamma\, Q(s_{t+1}, a_{t+1})$
- Expacted Value SARSA:

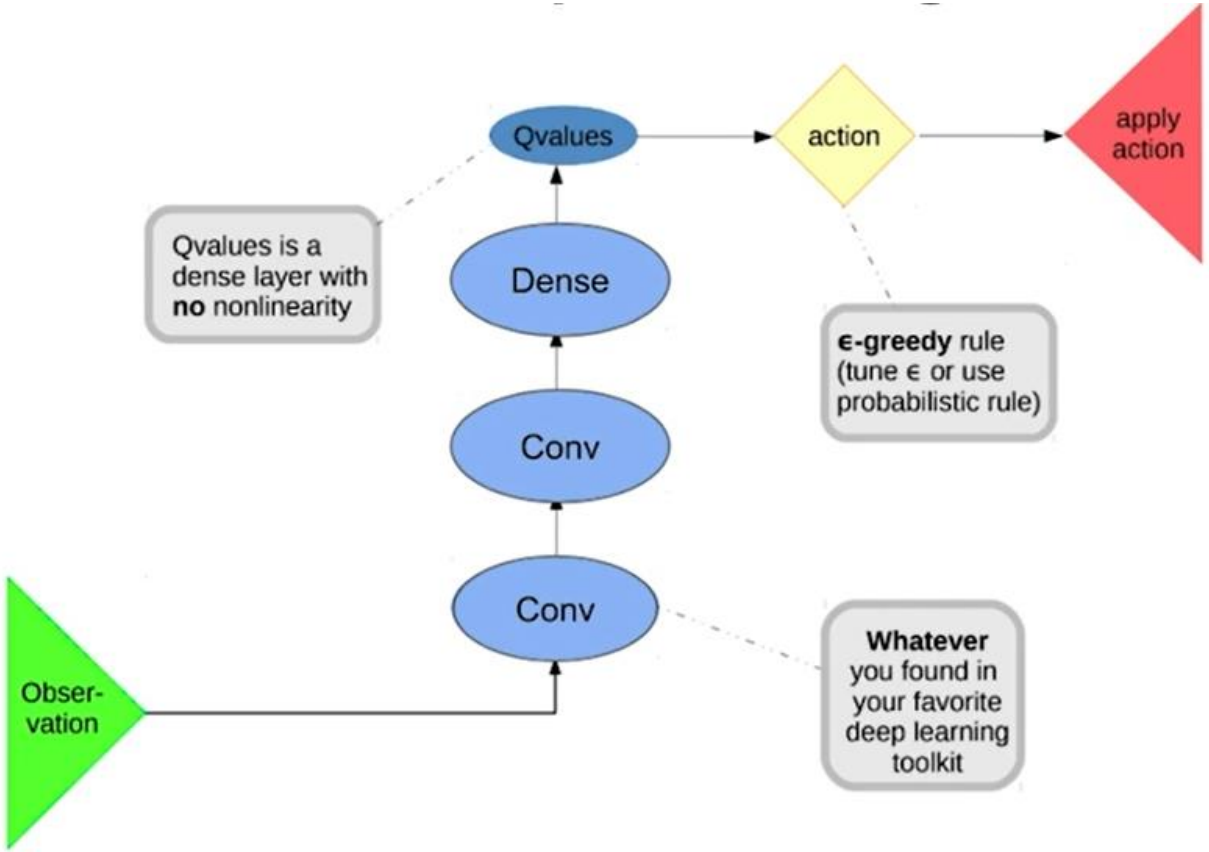
$$\hat{Q}(s_t, a_t) = r_t + \gamma\, E_\pi Q(s_{t+1}, a_{t+1})$$



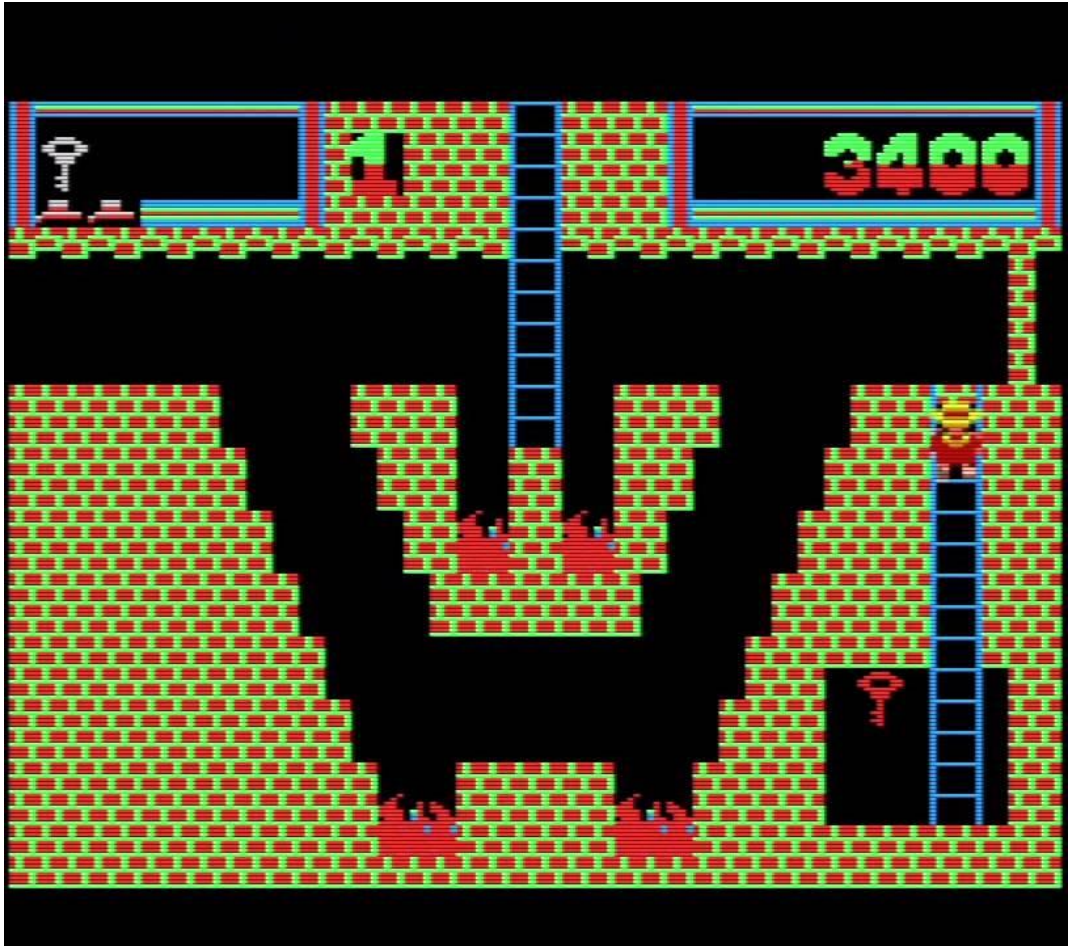
Deep Q-learning

Какую нейросеть применить для этой задачи?

Deep Q-learning




Вы уже играете в лигу 8 лет, я так полагаю, что вы хороший игрок
memesmix.net


3400

Проблема

- Обучаемая выборка не независима, т.е. изображения не являются независимыми одинаково распределенными величинами
- Модели забывают о частях среды, которые они не посещали в течение определенного времени
- Снижение темпов обучения

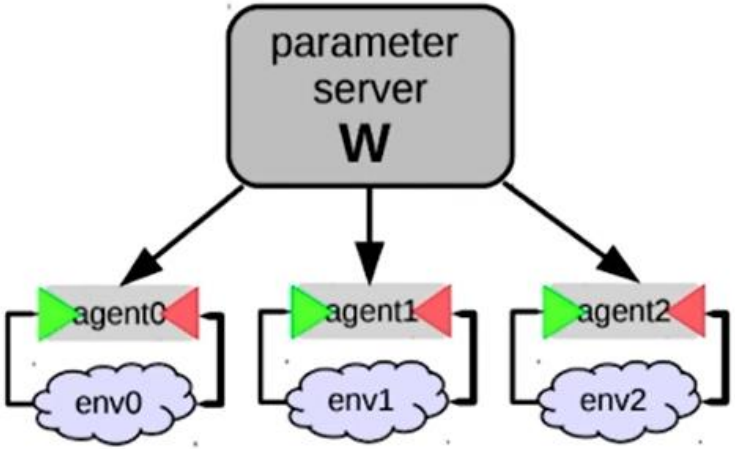
- Идеи?

Много агентов

- Идея: Использовать несколько агентов
- скорее всего, они будут исследовать разные части среды
- Более стабильное обучение
- Требует множество взаимодействий

Вопрос: агент автономный транспорт.

Какие проблемы?



Experience replay

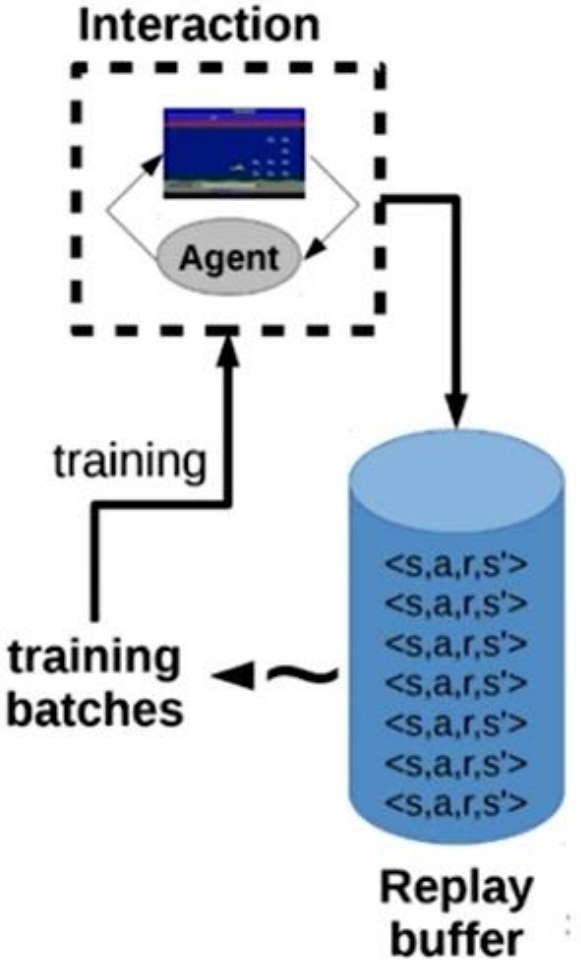
- Идея: сохранить несколько прошлых итераций

<s,a,r,s'>

Обучение проходит на случайных подвыборках

- *Подвыборки ближе к независимым*

- *более старые взаимодействия были получены при более слабой политике*



Experience replay

- Мы аппроксимируем Q(s,a) нейронной сетью
- Используем *experience replay* для обучения

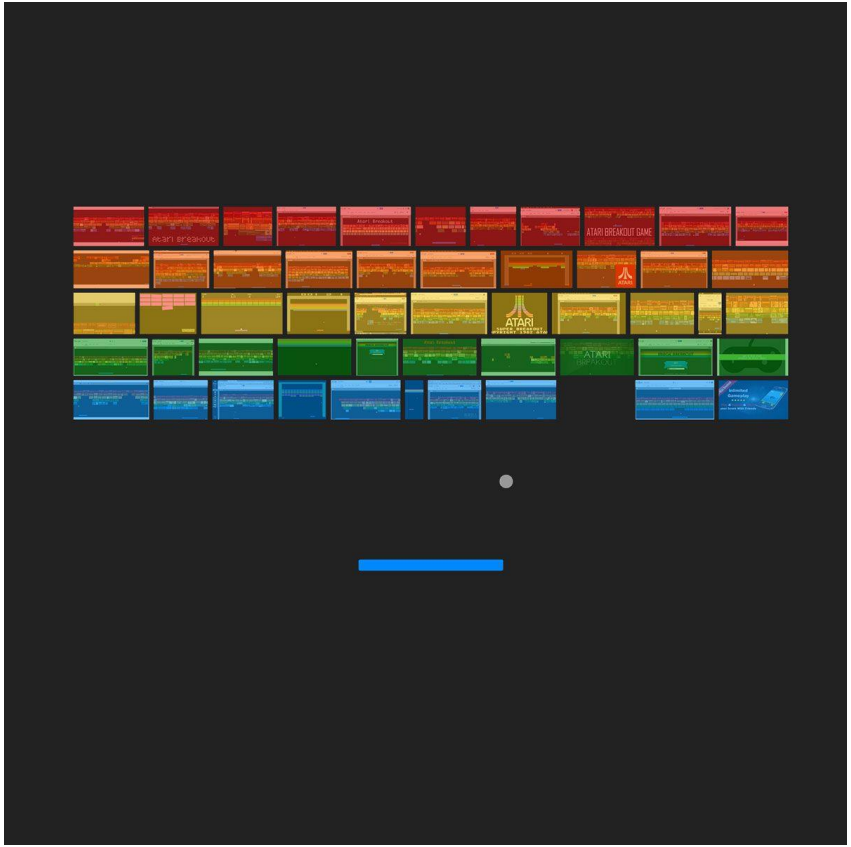
- *Вопрос: какие из перечисленных алгоритмов можем использовать?*

- Q-learning
- SARSA
- CEM
- Expected Value SARSA

При обучение on-policy алгоритмов

- Не использовать (или использовать не большой) experience replay
- Можно компенсировать сессиями параллельных игр

Лево или право?

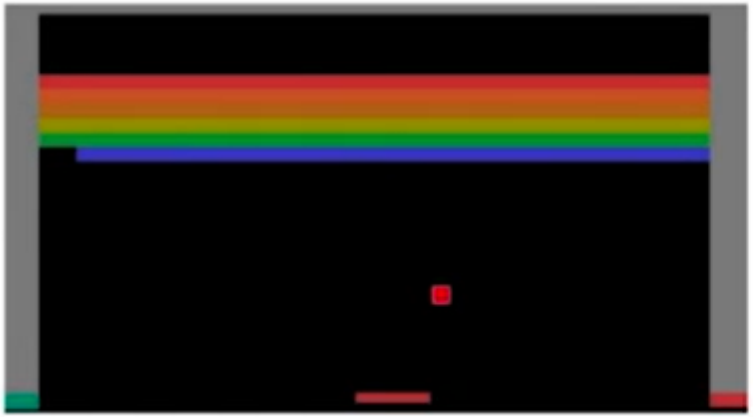


N-gram trick

Идея:

$s_t \neq o(s_t)$

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



Одна игра

Несколько игр

N-gram trick

- Предполагается марковость n-го порядка:
- Работает для скоростей/ускорений
- Потерпим неудачу, если информация выходит за рамки N кадров
- Непрактично для больших N

Автокорреляция

Target network

- Идея: использовать сеть с замороженными весами для расчета таргета.

$$\mathrm{L}(\theta) = E_{s\sim S, a\sim A}\left(\mathrm{Q}(s_t, a_t, \theta) - \left[r_t + \gamma \max_{a'} Q(s_{t+1}, a', \bar{\theta})\right]\right)^2$$

где $\bar{\theta}$ - замороженные веса

- *Hard target network:*

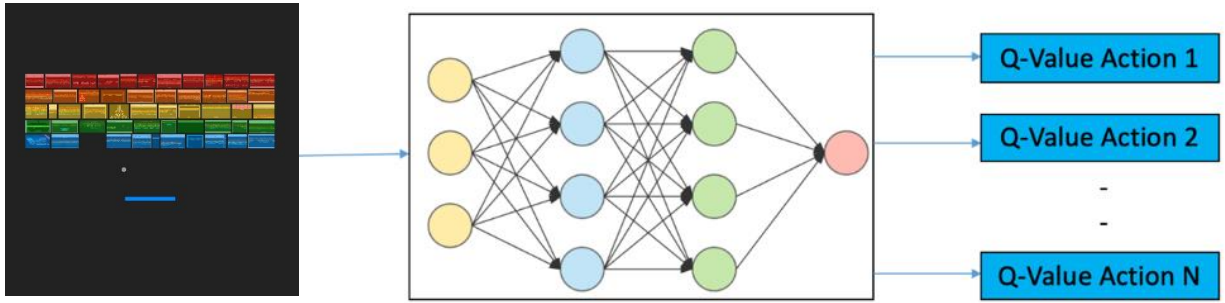
Обновлять $\bar{\theta}$ каждые n шагов и устанавливать значения θ

- *Soft target network:*

Обновлять $\bar{\theta}$ каждый шаг:

$$\bar{\theta} = (1-\alpha)\bar{\theta} + \alpha\theta$$

Playing Atari with Deep Reinforcement Learning (2013, DeepMind)



4 последних кадра

Experience replay

- Обновление весов:

$$\mathrm{L}(\theta) = E_{s\sim S, a\sim A}\left(\mathrm{Q}(s_t, a_t, \theta) - \left[r_t + \gamma \max_{a'} Q(s_{t+1}, a', \bar{\theta})\right]\right)^2$$

Обновление $\bar{\theta}$ каждые 5000 шагов обучения

10^6 последних переходов

Проблема завышенной оценки

- *Мы используем оператор max для расчета таргета*

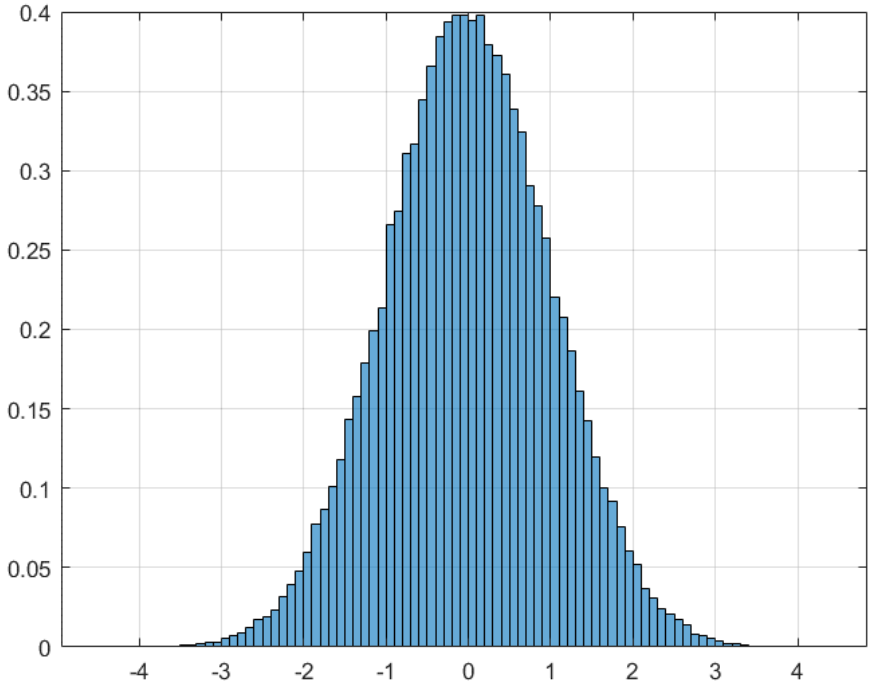
$$L(s,a) = \left(\mathrm{Q}(s,a) - \left[r_t + \gamma \max_{a'} Q(s',a')\right]\right)^2$$

Проблема

(мы хотим чтобы $E_{s\sim S, a\sim A}[L(s,a)] = 0$*)*

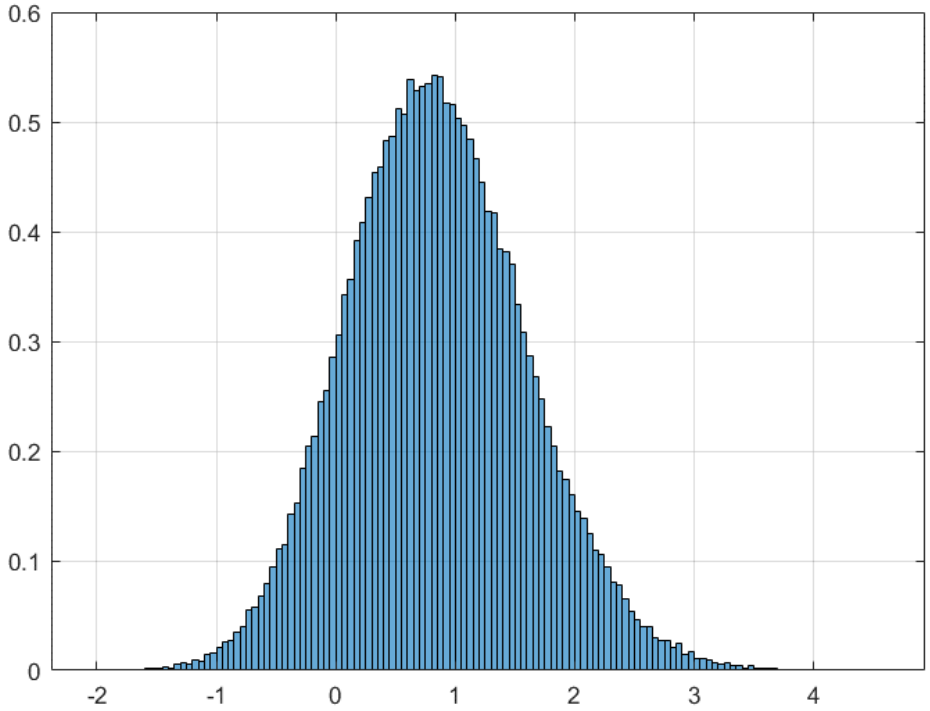
Проблема завышенной оценки

- *Mean(data)~0.001*

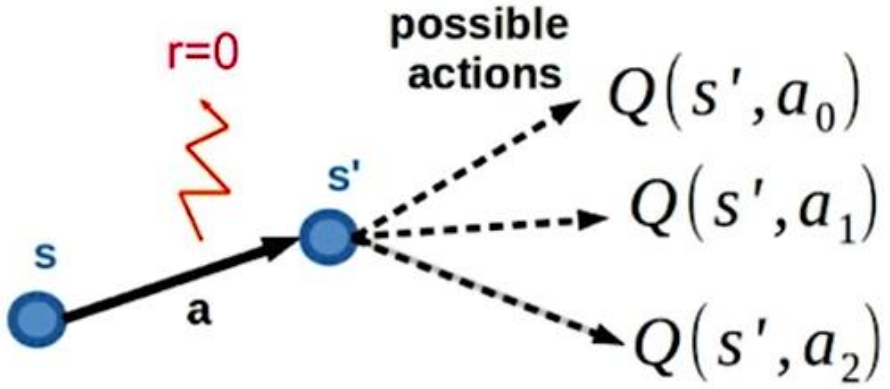


Проблема завышенной оценки

- *Mean(data)~ 0.8482*



Проблема завышенной оценки



- *Если мы обновляем Q(s,a) в соответствии с* $r_t + \gamma \max_{a'} Q(s', a')$, *мы будем иметь завышенную оценку*

$$E[\max_{a'} Q(s', a')] \geq \max_{a'} E[Q(s', a')]$$

Double Q-learning (NIPS 2010)

$$y = r_t + \gamma \max_{a'} Q(s', a')$$ *Q-learning target*

$$y = r_t + \gamma \max_{a'} Q\left(s', argmax_{a'} Q(s', a')\right)$$ *Q-learning target*

Идея: использовать 2 оценки q-values: Q^A, Q^B

Q-learning Они должны компенсировать несоответствия друг друга, т.к. они независимы.

$$y = r_t + \gamma \max_{a'} Q^A\left(s', argmax_{a'} Q^B(s', a')\right)$$ *Double Q-learning*

Double Q-learning (NIPS 2010)

Algorithm 1 Double Q-learning

Initialize Q^A,Q^B,s
repeat
  Choose a, based on $Q^A(s,\cdot)$ and $Q^B(s,\cdot)$, observe r, s'
  Choose (e.g. random) either UPDATE(A) or UPDATE(B)
  if UPDATE(A) **then**
    Define $a^* = \arg\max_a Q^A(s',a)$
    $Q^A(s,a) \leftarrow Q^A(s,a) + \alpha(s,a)\left(r + \gamma Q^B(s',a^*) - Q^A(s,a)\right)$
  else if UPDATE(B) **then**
    Define $b^* = \arg\max_a Q^B(s',a)$
    $Q^B(s,a) \leftarrow Q^B(s,a) + \alpha(s,a)(r + \gamma Q^A(s',b^*) - Q^B(s,a))$
  end if
  $s \leftarrow s'$
until end

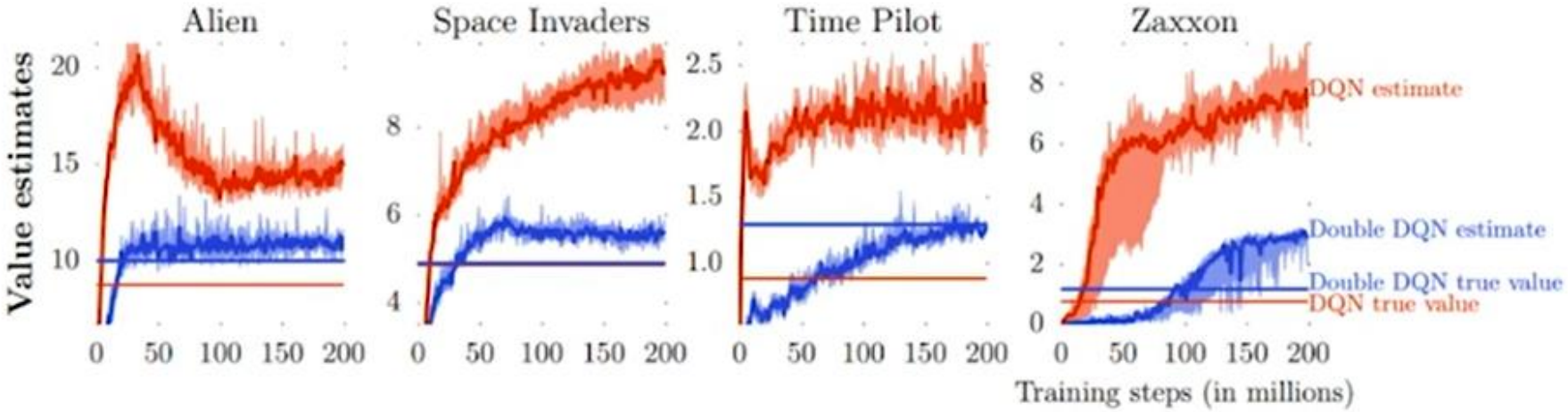
Сможем ли мы скомбинировать этот алгоритм с DQN?

Deep Reinforcement Learning with Double Q-learning (Deepmind, 2015)

Идея: использовать основную сеть для выбора действия

$$y_{dqn} = r_t + \gamma \max_{a'} Q(s', a', \bar{\theta})$$

$$y_{ddqn} = r_t + \gamma Q(s', argmax_{a'} Q(s', a', \theta), \bar{\theta})$$



	DQN	Double DQN	Double DQN (tuned)
Median	47.5%	88.4%	116.7%
Mean	122.0%	273.1%	475.2%

Experience replay

State	Action	Reward	Next state
s_0	a_0	0	s_1
s_1	a_1	0	s_2
...	...	...	...
s_(n-1)	a_(n-1)	0	s_n
s_n	**a_n**	**100**	**s_(n+1)**
s_(n+1)	a_(n+1)	0	s_(n+2)
...	...	...	...

Experience replay: приоритезация (DeepMind, 2016)

Идея: семплировать переходы из буфера более грамотно

Мы хотим назначить вероятности для каждого перехода. Можем использовать абсолютное значение TD ошибки перехода как вероятности

TD-error $\delta = Q(s,a) - (r + \gamma Q(s', argmax_{a'} Q(s', a', \theta), \bar{\theta})$

$p = |\delta|$

$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$ здесь α параметр приоритета (0 – равномерный выбор)

Какую видите проблему?

Experience replay: приоритезация (DeepMind, 2016)

Решение: корректно обучать используя importance-sampling weights

$$w_i = \left(\frac{1}{N}\frac{1}{P(i)}\right)^{\beta} \quad \text{где } \beta \text{ параметр}$$

Таким образом мы семплируем $P(i) = \frac{p_i^{\alpha}}{\sum_k p_k^{\alpha}}$ и умножая ошибку на w_i

Experience replay: приоритезация (DeepMind, 2016)

Дополнительные детали

Нормировка весов на $1/\max_i w_i$

При записи переходов в experience replay, принимаем maximal priority $p_t = \max_{i<t} p_i$

https://youtu.be/UXurvvDY93o

Dueling Network Architectures for Deep Reinforcement Learning (DeepMind, 2016)

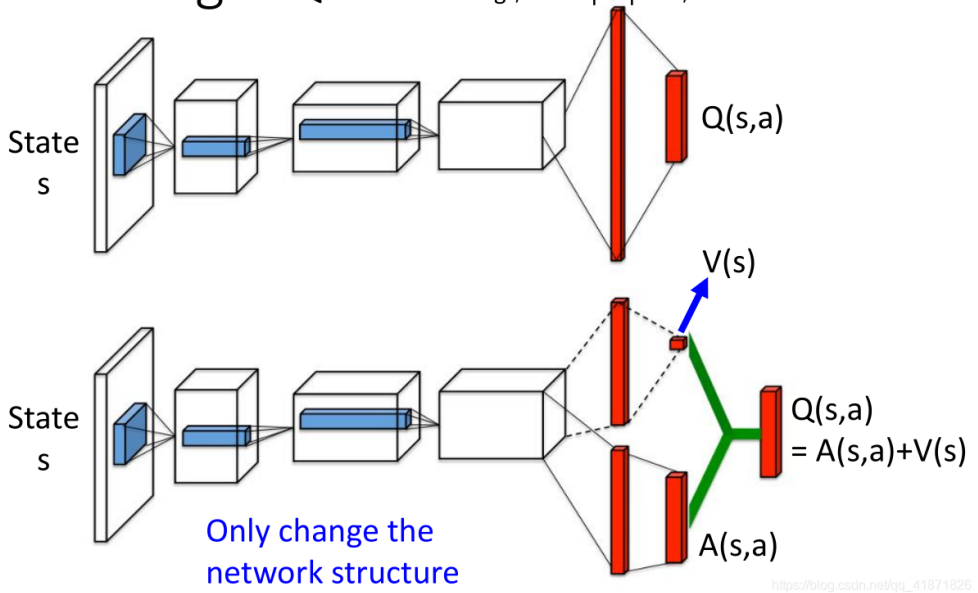
Идея: изменить архитектуру НС

Advantage function $A(s,a) = Q(s,a) – V(s)$

Таким образом $Q(s,a) = A(s,a) + V(s)$

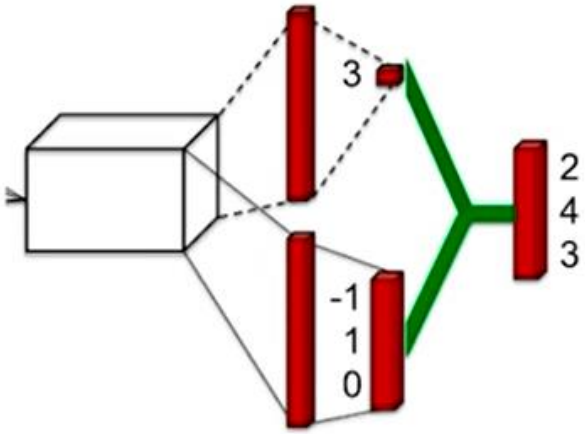
Проблема?

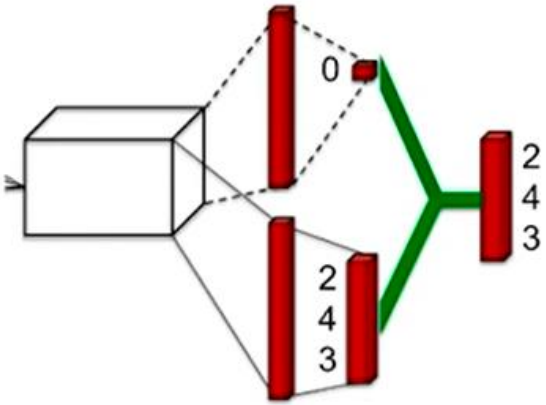
Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, Nando de Freitas, "Dueling Network Architectures for Deep Reinforcement Learning", arXiv preprint, 2015



Dueling Network Architectures for Deep Reinforcement Learning (DeepMind, 2016)

Имеем дополнительную степень свободы





Какой вариант лучше?

Dueling Network Architectures for Deep Reinforcement Learning (DeepMind, 2016)

Решение: требуем $\max_{a' \in |A|} A(s, a'; \theta, \alpha)$ равнялось нулю

Таким образом Q функция рассчитывается как:

$$Q(s, a, \alpha, \beta) = V(s; \theta, \beta) + \left(A(s, a; \theta, \alpha) - \max_{a' \in |A|} A(s, a'; \theta, \alpha) \right)$$

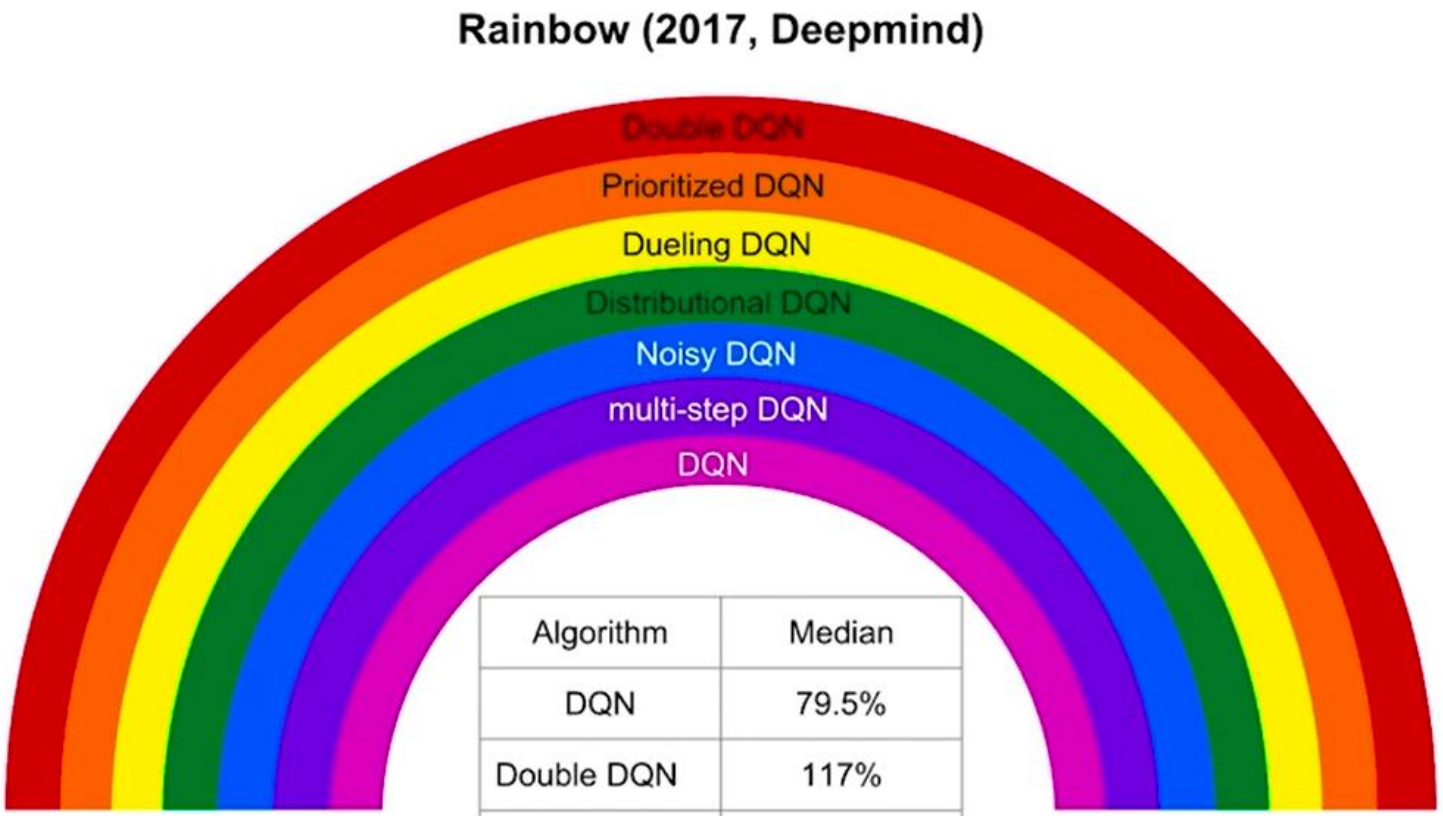
Авторы статьи также предложили следующий способ расчета Q-values:

$$Q(s, a, \alpha, \beta) = V(s; \theta, \beta) + \left(A(s, a; \theta, \alpha) - \frac{1}{|A|} \sum_{a'} A(s, a'; \theta, \alpha) \right)$$

Этот вариант повышает стабильность оптимизации

Асинхронное обучение для DLR (DeepMind, 2016)

Rainbow (DeepMind, 2017)



Algorithm	Median
DQN	79.5%
Double DQN	117%
Rainbow	223%

R2D2: (Deepmind, 2018)

- LSTM
- Distributed Prioritized Experience replay
- N-step DQN
- Reword re-scaling
- Double DQN
- Dueling DQN

Медианная производительность: 1920% от человеческой