Продолжение Q-learning. DQN

Tinkoff.ru

25.06.2019

Резяпкин Вячеслав

Нейронные сети и RL

- Double Q-Learning
- Нейронные сети
- Deep Q Network (DQN)
- Continuous actions

Q-обучение

$Q -$ обучение $(Q - learning)$:

Есть какая $-$ то $Q(s, a)$. Рассмотрим пример $s_t, a_t \rightarrow r_t \rightarrow s_{t+1}$

Можем посчитать $Q(s_t, a_t)$ и $r_t + \gamma \max_{a'} Q(s_{t+1}, a')$

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

Для Q^* стратегия тривиальна:

$$\pi^*(a \mid s) = \operatorname*{argmax}_a Q^*(s, a)$$

Что делать для произвольной Q в процессе обучения?

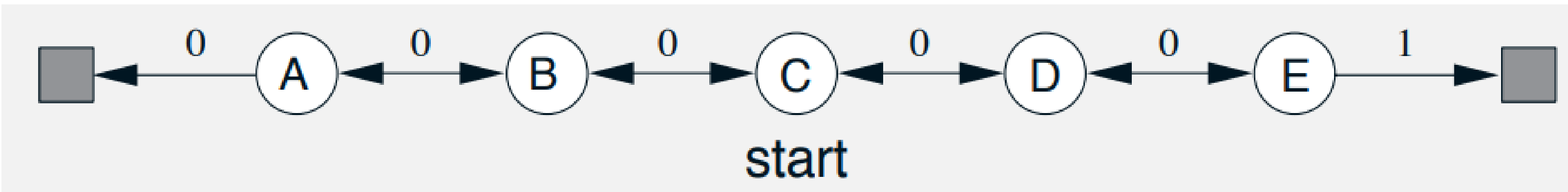
$\pi(a \mid s) =$ с некоторой вероятностью не оптимальное действие(в смысле Q функции)

Exploration-exploitation dilemma

$\boldsymbol{\varepsilon}$ – жадная стратегия:

$$\pi(a \mid s) = \begin{cases} \underset{a}{\operatorname{argmax}} Q(s,a) & \text{с вероятностью } 1-\varepsilon \\ \forall a \neq \underset{a}{\operatorname{argmax}} Q(s,a) & \text{с вероятностью } \dfrac{\varepsilon}{|A|-1} \end{cases}$$

Пример вычисления функции ценности



$$V^*(s) = \max_{\pi} \mathop{\mathrm{E}}_{\tau\sim\pi} [R(\tau)\,|s_0 = s]$$

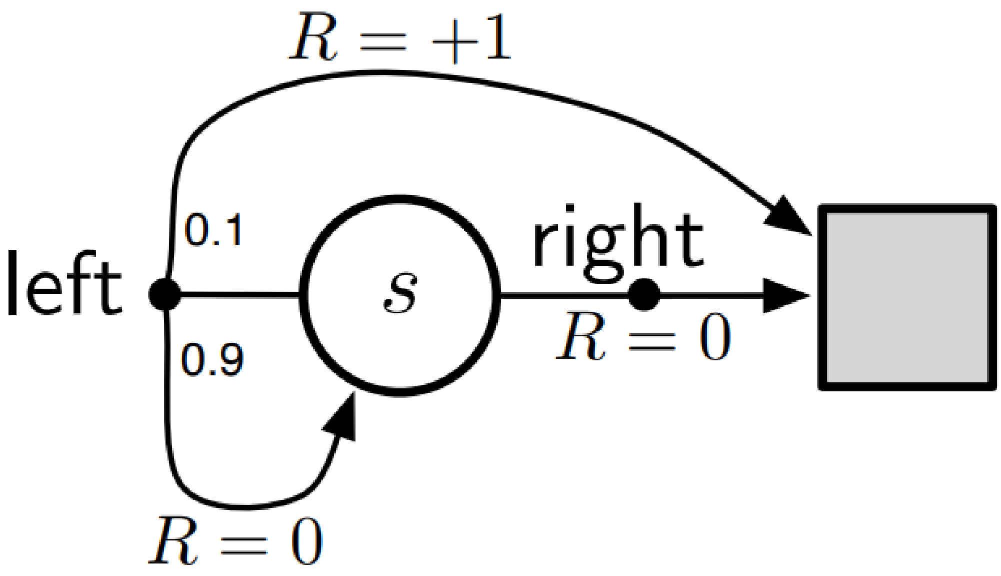
$$V^*(s) = \max_{a} \mathop{\mathrm{E}}_{s'\sim P} [r(s,a) + \gamma V^*(s')]$$

$$V^*(C) = \frac{1}{2}$$
$$V^*(A) = 0.5 * 0 + 0.5 * \big(0 + V^*(B)\big)$$
$$V^*(B) = 0.5 * \big(0 + V^*(C)\big) + 0.5 * \big(0 + V^*(A)\big)$$

$$V^*(C) = \frac{1}{2}$$
$$2V^*(A) = V^*(B)$$
$$4V^*(B) = 2V^*(A) + 1$$

$$V^*(B) = \frac{1}{3}$$

Ещё пример



Политики:

$$\pi(\text{left}|s) = 1$$

$$b(\text{left}|s) = \frac{1}{2}$$

Найти $V^{\pi}(s)$ и $V^{b}(s)$:

- С помощью уравнения Беллмана
- Через определение

Double Q-Learning

$$\mathbb{E}\max(X,Y) \geq \max(\mathbb{E}X, \mathbb{E}Y)$$

$$\mathbb{E}\max(X,Y) = \int \max(x,y)\, f(x)f(y)dxdy \geq \int xf(x)f(y)dxdy = \mathbb{E}\mathrm{X}$$

$$\mathbb{E}\max(X,Y) = \int \max(x,y)\, f(x)f(y)dxdy \geq \int yf(x)f(y)dxdy = \mathbb{E}\mathrm{Y}$$

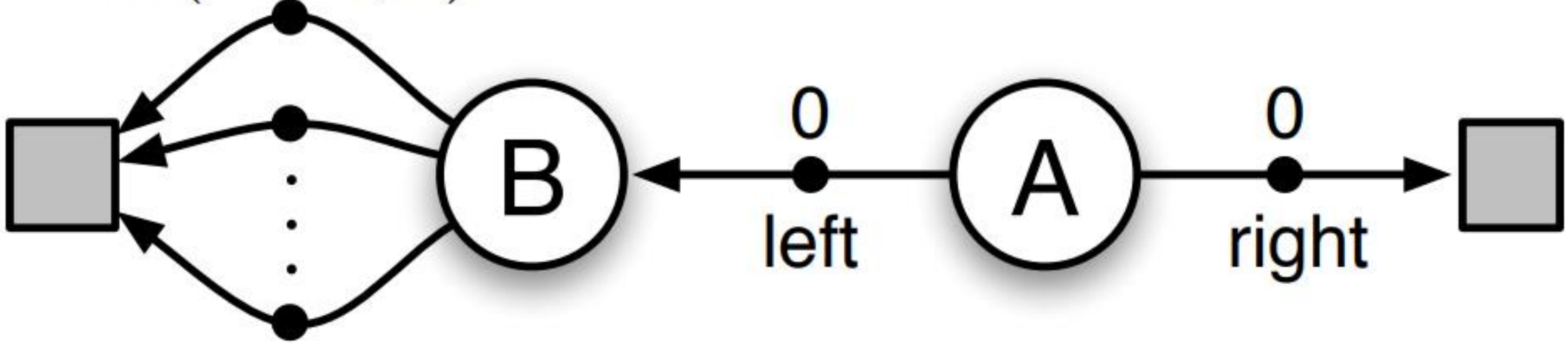
Пусть $X, Y \sim U[0,1], i.i.d$

$$\max(\mathbb{E}X, \mathbb{E}Y) = 0.5$$

$$\mathbb{E}\max(X,Y) = \int_{\square} \max(x,y)\, dxdy = 2\int_{\triangle} xdxdy = 2\int_0^1 x^2dx = 2/3$$

Double Q-Learning

$$R_i \sim \mathcal{N}(-0.1, 1)$$



*Посчитаем Q**

$Q^*(\mathrm{A}, \mathrm{right}) = 0$

$Q^*(\mathrm{A}, \mathrm{left}) = \max Q^*(\mathrm{B}, \mathrm{a}) = \max \mathbb{E}(R_i) = -0.1$

Но в начале мы не располагаем большой выборкой, поэтому оценка среднего получается с большой дисперсией

Оценка Q(A, left) после первого шага:

$\max Q(\mathrm{B}, \mathrm{a}) = \max(R_i)$

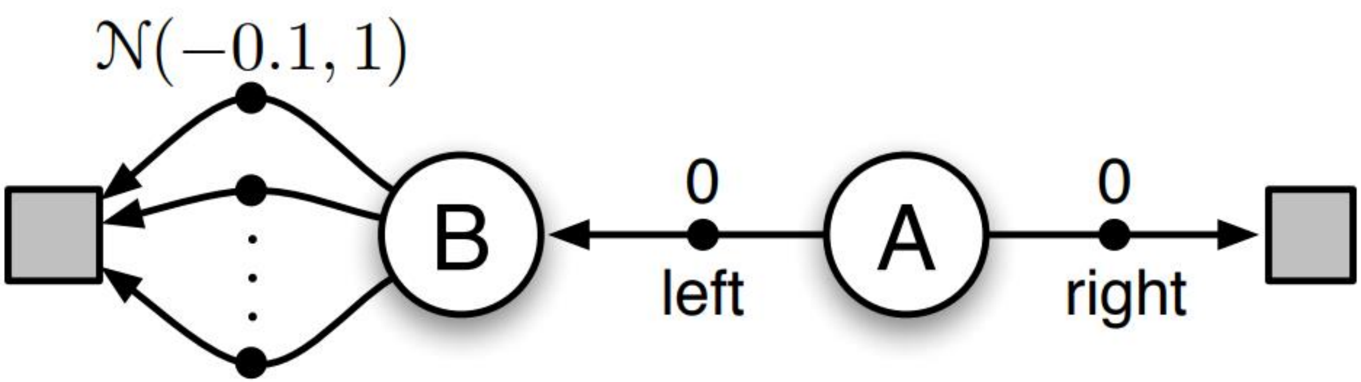
$\mathbb{E}\max(Q(\mathrm{B}, \mathrm{a}) = \mathbb{E}\max(R_i) \geq \max \mathbb{E}(R_i) = -0.1$

Double Q-Learning

<u>Ну и что с того, что переоценивает?</u>

Нам же важно правильно найти лучшее действие в данном состоянии, а с ранжированием (упорядочиванием) всё в порядке

Посмотрим еще раз



$$\mathbb{E}\mathrm{Q}(\mathrm{A}, left) = \mathbb{E}\max(\mathrm{Q}(\mathrm{B}, \mathrm{a}) = \mathbb{E}\max(R_i) \geq -0.1$$

$Q(A, right) = 0$ не сильно больше, чем -0.1

$\Rightarrow$ возможно такое, что в начале $\mathrm{Q}(\mathrm{A}, left) \geq Q(A, right)$

Но это неверно!

Double Q-Learning

Заметим, что $\max Q(S', a) = Q(S', argmax\, Q(S', a))$

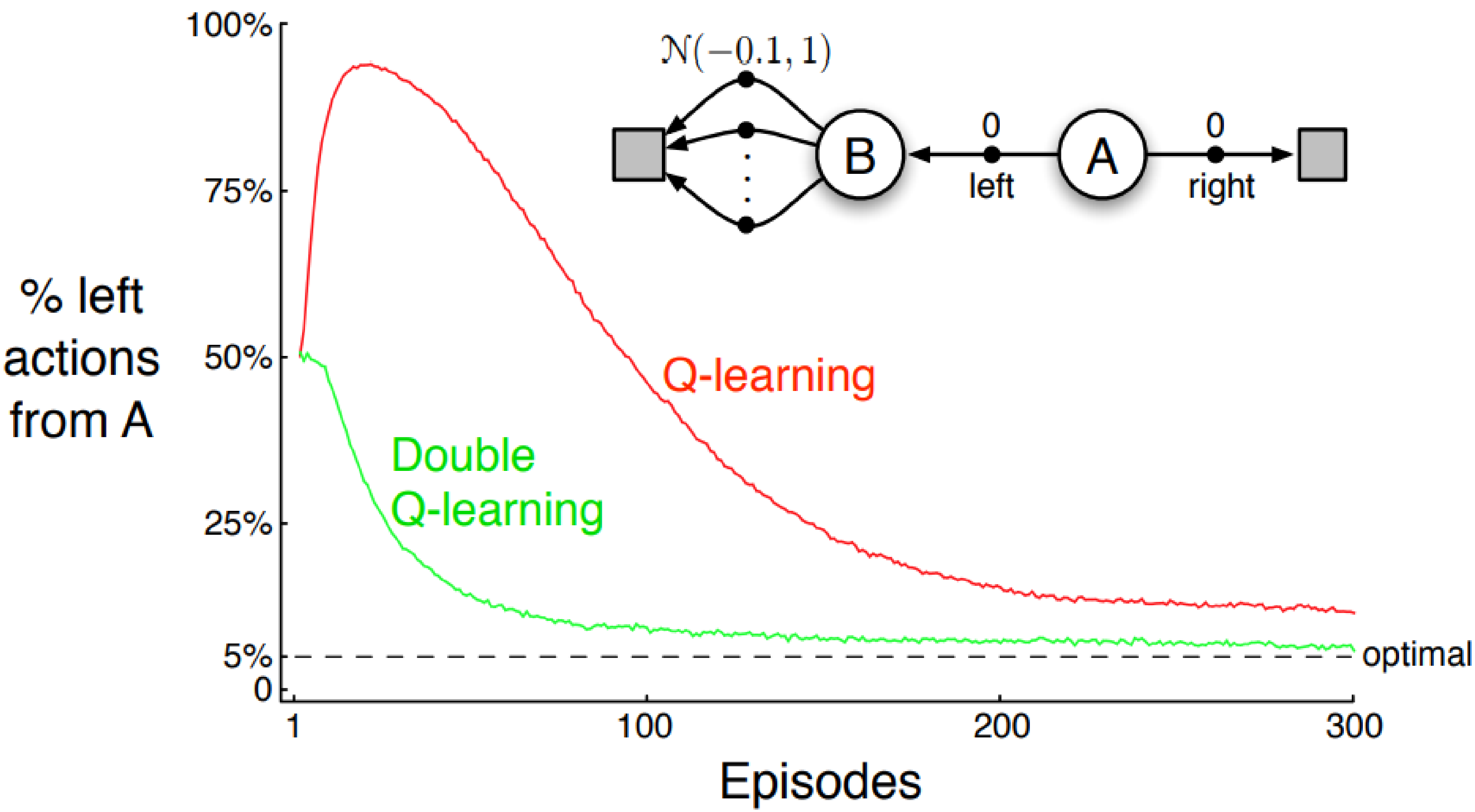
Double Q-learning, for estimating $Q_1 \approx Q_2 \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q_1(s, a)$ and $Q_2(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, such that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using the policy ε-greedy in $Q_1 + Q_2$
    Take action A, observe R, S'
    With 0.5 probabilility:
$$Q_1(S, A) \leftarrow Q_1(S, A) + \alpha\Big(R + \gamma Q_2\big(S', \operatorname{argmax}_a Q_1(S', a)\big) - Q_1(S, A)\Big)$$
    else: Шумы некоррелированны, поэтому переоценивания не происходит
$$Q_2(S, A) \leftarrow Q_2(S, A) + \alpha\Big(R + \gamma Q_1\big(S', \operatorname{argmax}_a Q_2(S', a)\big) - Q_2(S, A)\Big)$$
    $S \leftarrow S'$
  until S is terminal

Что изменилось для предтерминальных состояний?

Double Q-Learning



Что делать, если состояний много?

- Если пространство состояний является непрерывным, то можно его дискретизировать

- Проблема дискретизации: проклятие размерности
  Количество состояний экспоненциально растет с ростом числа размерностей

- Если их просто много, но не бесконечно, то получается большая табличка

- Из-за большого количества состояний процесс обучения может затянуться

- Хочется использовать пространственную структуру состояний:
  часто близкие состояния имеют близкие значения Q функции

- Не обязательно искать табличное отображение. Можно искать полноценную функцию

Что делать, если состояний много?

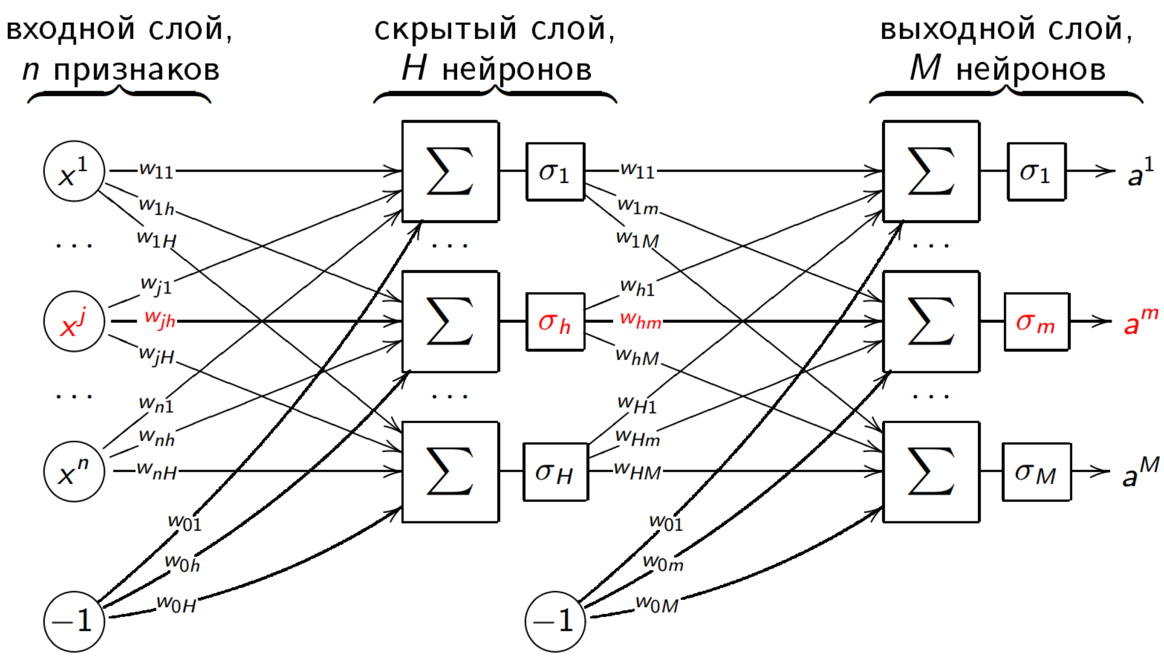
Опять проблема: функций много, непонятно как найти ту самую

Будем аппроксимировать: выделим параметризованное семейство функций, которым можно хорошо приближать искомую Q-функцию

$Q(s, a) = Q(s, a \mid \theta)$

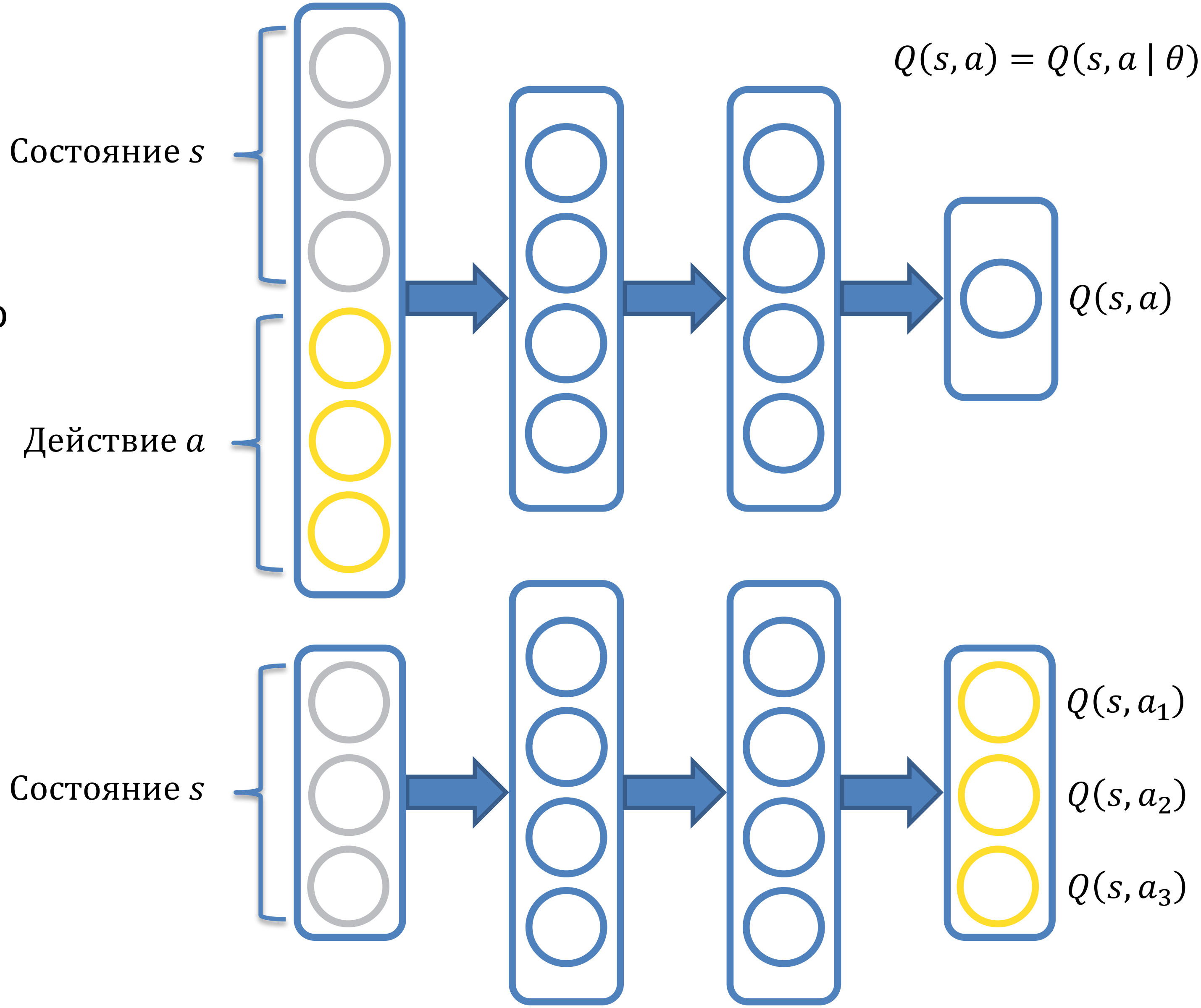
Как аппроксимировать:

- Линейная регрессия
  $y = \theta_1 x + \theta_0$
- Полиномы $y = \sum_0^n \theta_k x^k$
- Нейронная сеть



Варианты

Используем эффективную аппроксимацию



Посмотрим еще раз на Q-обучение

Табличный метод

$Q^*(s,a) = r(s,a\,) + \ \gamma \mathbb{E}_{p(s'|\,s,a)} \max_{a'} Q^*(s',a')$ – уравнение Беллмана для Q^*

$$F(Q) = \frac{1}{|S||A|} \sum_{s,a} \left(Q(s,a) - r(s,a\,) - \ \gamma \mathbb{E}_{p(s'|\,s,a)} \max_{a'} Q\,(s',a') \right)^2 \to \min_{Q}$$

$$Q_{new}(s,a) = Q_{old}(s,a) - \ \alpha \nabla_{Q(s,a)} F(Q) =$$

$$= Q_{old}(s,a) - 2\alpha (Q_{old}(s,a) - r(s,a\,) - \ \gamma \max_{a'} Q_{old}(s',a'))$$

Параметризация

$$Q(s,a) = Q(s,a \mid \theta)$$

$$\theta_{new} = \theta_{old} - \ \alpha \nabla_{\theta} F(Q) =$$

$$= \theta_{old} - 2\alpha (Q(s,a,\theta_{old}) - r(s,a) - \ \gamma \max_{a'} Q(s',a',\theta_{old}))\, \nabla_{\theta} Q(s,a,\theta)|_{\theta_{old}}$$

Q-обучение

1. Сделаем некоторое действие и получим (s_t, a_t, r_t, s_{t+1})

2. $y_t = \begin{cases} r_t, & \text{если } s_t - \text{терминальное} \\ r_t + \gamma \max\limits_{a} Q(s_{t+1}, a \quad, \theta) \end{cases}$

3. $\theta \leftarrow \theta - \alpha \sum (Q(s_t, a_t, \theta) - y_t) \nabla_\theta Q(s_t, a_t, \theta)$

Проблемы:

1. Мы решаем задачу регрессии, но данные коррелированы

2. Целевая переменная изменяется с обновлением θ

Использование памяти для Q-обучения

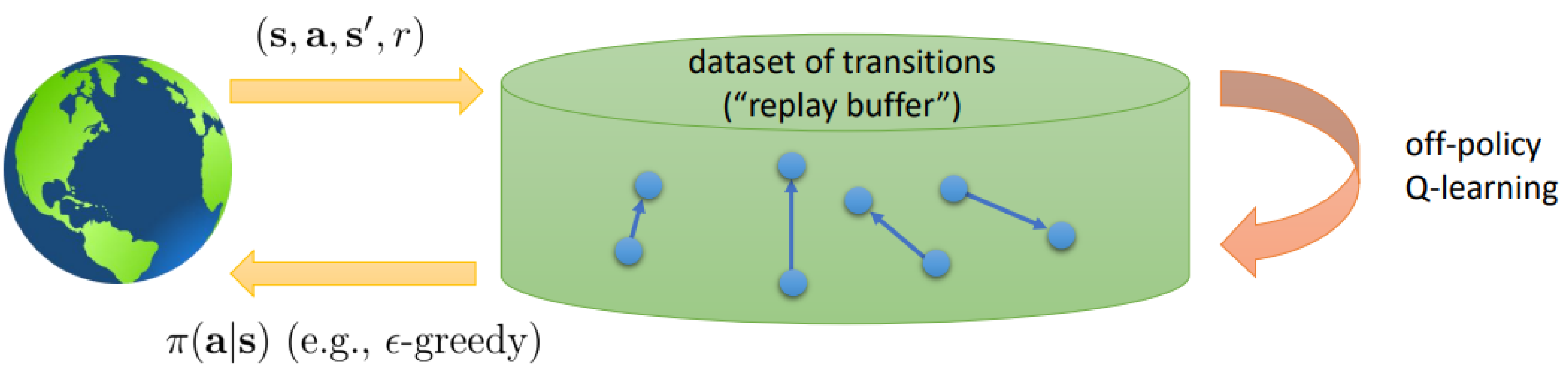
Есть какая – то $Q(s, a \mid \theta)$.

Действуем ε – жадно и получаем примеры $s_t, a_t \rightarrow r_t \rightarrow s_{t+1}$

Experience replay:

Сохраняем опыт $e_t = (s_t, a_t, r_t, s_{t+1})$ в память M

Делаем шаги обучения не только по новому опыту, но и по сохраненному в памяти



Q-обучение’

Инициализируем память M *двусторонней очередью*

Инициализируем функцию Q со случайными весами

for $episode_i$ in $[1 \ldots N]$:

Инициализируем s_0

for t in $[0 \ldots T]$:

$$a_t = \begin{cases} \underset{a}{\operatorname{argmax}} Q(s_t, a) & \text{с вероятностью } 1 - \varepsilon \\ \forall a \neq \underset{a}{\operatorname{argmax}} Q(s_t, a) & \text{с вероятностью } \frac{\varepsilon}{|A| - 1} \end{cases}$$

Получаем s_{t+1}, r_t

Сохраняем $e_t = (s_t, a_t, r_t, s_{t+1})$ в память M

Сэмплируем минибатч (s_j, a_j, r_j, s'_j) из M

$$y_j = \begin{cases} r_j, & \text{если } s_j - \text{терминальное} \\ r_j + \gamma \max_{a'} Q(s'_j, a', \theta) \end{cases}$$

$$\theta \leftarrow \theta - \alpha \sum \left(Q(s_j, a_j, \theta) - y_j\right) \nabla_\theta Q(s_j, a_j, \theta)$$

end for

end for

Target network

$$y_j = \begin{cases} r_j, & \text{если } s_j - \text{терминальное} \\ r_j + \gamma \max_{a'} Q(s'_j, a', \theta) \end{cases}$$

Таргет всё еще двигается

$$\theta \leftarrow \theta - \alpha \sum \left(Q(s_j, a_j, \theta) - y_j\right) \nabla_\theta Q(s_j, a_j, \theta)$$

Решение: Зафиксируем веса θ' для таргета

$$y_j = \begin{cases} r_j, & \text{если } s_j - \text{терминальное} \\ r_j + \gamma \max_{a'} Q(s'_j, a', \theta') & ! \end{cases}$$

$$\theta \leftarrow \theta - \alpha \sum \left(Q(s_j, a_j, \theta) - y_j\right) \nabla_\theta Q(s_j, a_j, \theta)$$

И будем обновлять: $\theta' \rightarrow (1 - \tau)\theta' + \tau\theta$

Например, $\tau = 0.001$

DQN (Deep Q-Network)

$\theta' \coloneqq \theta$

for $episode_i$ in $[1 \ldots N]$:

Инициализируем s_0

for t in $[0 \ldots T]$:

$$a_t = \begin{cases} \underset{a}{\operatorname{argmax}}\, Q(s_t, a) & \text{с вероятностью } 1 - \varepsilon \\ \forall a \neq \underset{a}{\operatorname{argmax}}\, Q(s_t, a) & \text{с вероятностью } \frac{\varepsilon}{|A|-1} \end{cases}$$

Получаем s_{t+1}, r_t

Сохраняем $e_t = (s_t, a_t, r_t, s_{t+1})$ в память M

Сэмплируем минибатч (s_j, a_j, r_j, s'_j) из M

$$y_j = \begin{cases} r_j, & \text{если } s_j - \text{терминальное} \\ r_j + \gamma \max_{a'} Q\left(s'_j, a', \theta\right) \end{cases}$$

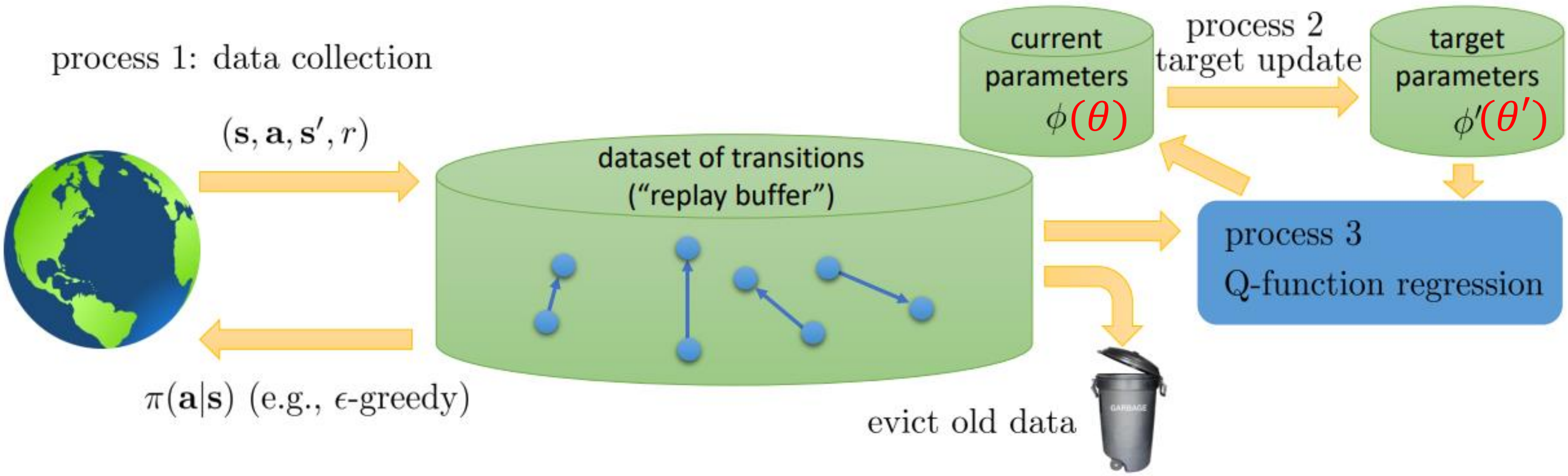
$$\theta \leftarrow \theta - \alpha \sum \left(Q\left(s_j, a_j, \theta\right) - y_j\right) \nabla_\theta Q\left(s_j, a_j, \theta\right)$$

$\theta' \rightarrow (1 - \tau)\theta' + \tau\theta$

end for

end for

DQN. Схема обучения



Можно ли совместить Double Q-Learning и Target network?

Target Network

$$y_j = \begin{cases} r_j, & \text{если } s_j - \text{терминальное} \\ r_j + \gamma \max_{a'} Q(s'_j, a', \theta') \end{cases}$$

$$\theta \leftarrow \theta - \alpha \sum (Q(s_j, a_j, \theta) - y_j) \nabla_\theta Q(s_j, a_j, \theta)$$

$$\theta' \rightarrow (1-\tau)\theta' + \tau\theta$$

Double Q-Learning

$$y_j = \begin{cases} r_j, \quad \text{если } s_j - \text{терминальное} \\ r_j + \gamma Q(s'_j, argmax\, Q(s'_j, a_j, \theta_1), \theta_2) \end{cases}$$

$$\theta_2 \leftarrow \theta_2 - \alpha \sum (Q(s_j, a_j, \theta_2) - y_j) \nabla_{\theta_2} Q(s_j, a_j, \theta_2)$$

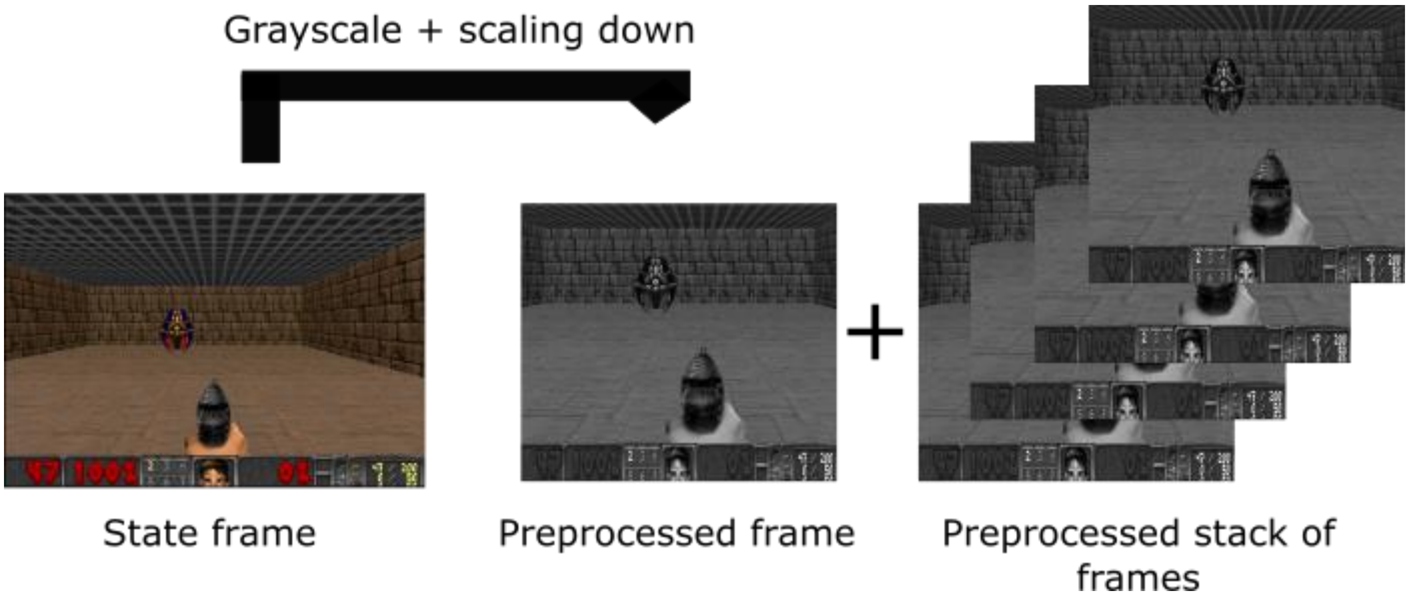
Double Q-Learning with target network

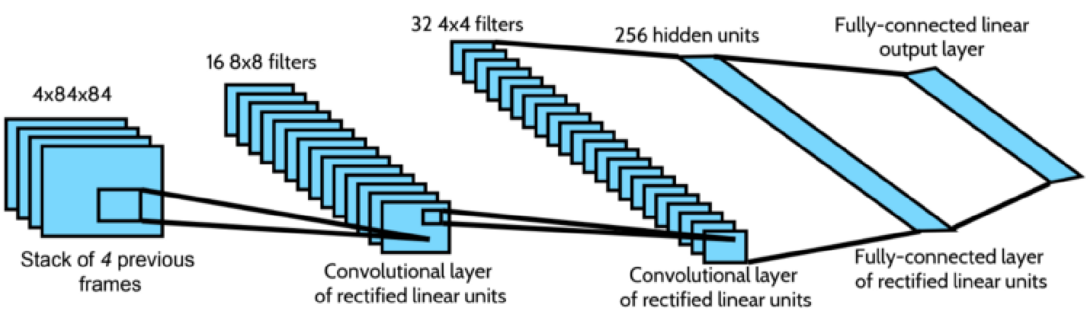
$$y_j = \begin{cases} r_j, \quad \text{если } s_j - \text{терминальное} \\ r_j + \gamma Q(s'_j, argmax\, Q(s'_j, a_j, \theta), \theta') \end{cases}$$

$$\theta \leftarrow \theta - \alpha \sum (Q(s_j, a_j, \theta) - y_j) \nabla_\theta Q(s_j, a_j, \theta)$$

$$\theta' \rightarrow (1-\tau)\theta' + \tau\theta$$

State


Grayscale + scaling down
State frame
Preprocessed frame
Preprocessed stack of frames


4x84x84
16 8x8 filters
32 4x4 filters
256 hidden units
Fully-connected linear output layer
Stack of 4 previous frames
Convolutional layer of rectified linear units
Convolutional layer of rectified linear units
Fully-connected layer of rectified linear units

Непрерывные действия

Почему это проблема для рассмотренных алгоритмов?

1. Нейронные сети?
   - Как изменится архитектура?
   - А как искать действие с максимальным Q-значением?
   - Численная оптимизация - медленно

2. Дискретизация
   - Часто непрерывные действия уже дискретизированы в среде:
     Например, в гонках скорость машины регулируется 2-3мя действиями
   - Теряем информацию
   - Не делаем оптимальные действия

3. Случайно сэмплируем n действий;
   Затем выбираем действие с максимальным Q
   - Можно параллелить

DDPG

Сделаем еще одну нейронную сеть $\mu(s|\phi)$ для выбора действия - actor

Как?

Мы хотим добиться $\mu(s|\phi) \cong \mathrm{argmax}\, \mathrm{Q}(\mathrm{s}, \mathrm{a}|\theta)$

Значит задача:

$$\phi \to \underset{\phi}{\mathrm{argmax}}\ \mathrm{Q}(\mathrm{s}, \mu(s|\phi)|\theta)$$

$$\mathrm{Q}(\mathrm{s}, \mu(s|\phi)|\theta) \to \max_{\phi}$$

Градиентный подъём

$$\frac{dQ}{d\phi} = \frac{d\mathrm{Q}}{d\mu}\frac{d\mu}{d\phi}$$

DDPG:

1. take some action $\mathbf{a}_i$ and observe $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$, add it to $\mathcal{B}$
2. sample mini-batch $\{\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j, r_j\}$ from $\mathcal{B}$ uniformly
3. compute $y_j = r_j + \gamma \max_{\mathbf{a}'_j} Q_{\phi'}(\mathbf{s}'_j, \mu_{\theta'}(\mathbf{s}'_j))$ using *target* nets $Q_{\phi'}$ and $\mu_{\theta'}$
4. $\phi \leftarrow \phi - \alpha \sum_j \frac{dQ_\phi}{d\phi}(\mathbf{s}_j, \mathbf{a}_j)(Q_\phi(\mathbf{s}_j, \mathbf{a}_j) - y_j)$
5. $\theta \leftarrow \theta + \beta \sum_j \frac{d\mu}{d\theta}(\mathbf{s}_j)\frac{dQ_\phi}{d\mathbf{a}}(\mathbf{s}_j, \mathbf{a})$
6. update ϕ' and θ' (e.g., Polyak averaging)

Summary

- Обычный Q-learning может переоценивать истинные значения
  - Double Q-learning
- Нейронные сети отлично подходят для RL
  - Хорошо выделяют сложные нелинейные зависимости
  - Позволяют дообучаться по новым примерам
- DQN – естественное расширение табличного Q-обучения
  - Вместо таблиц используем параметризованную нейросетью Q-функцию
  - Стабилизируем обучение использованием памяти
  - Target network, чтобы целевая переменная была менее чувствительна к изменениям весов сети
- DDPG для работы с непрерывными действиями