НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

Факультет компьютерных наук

Обучение с подкреплением

Москва, 2021

Scaling up algorithms

- Value function approximations
- Stochastic gradient descent
- Experience replay
- DQN

Обозначения:

- S, Statement - Состояние
- Environment - Среда
- R, reward - награда
- S’ - новое состояние

Value function approximation

- Слишком большое количество состояний(не влезает в память)
- Уйдет слишком много времени чтобы обучиться на каждом состоянии

Пример: шахматы: ~очень очень много вариантов

Value function approximation

Linear VFA

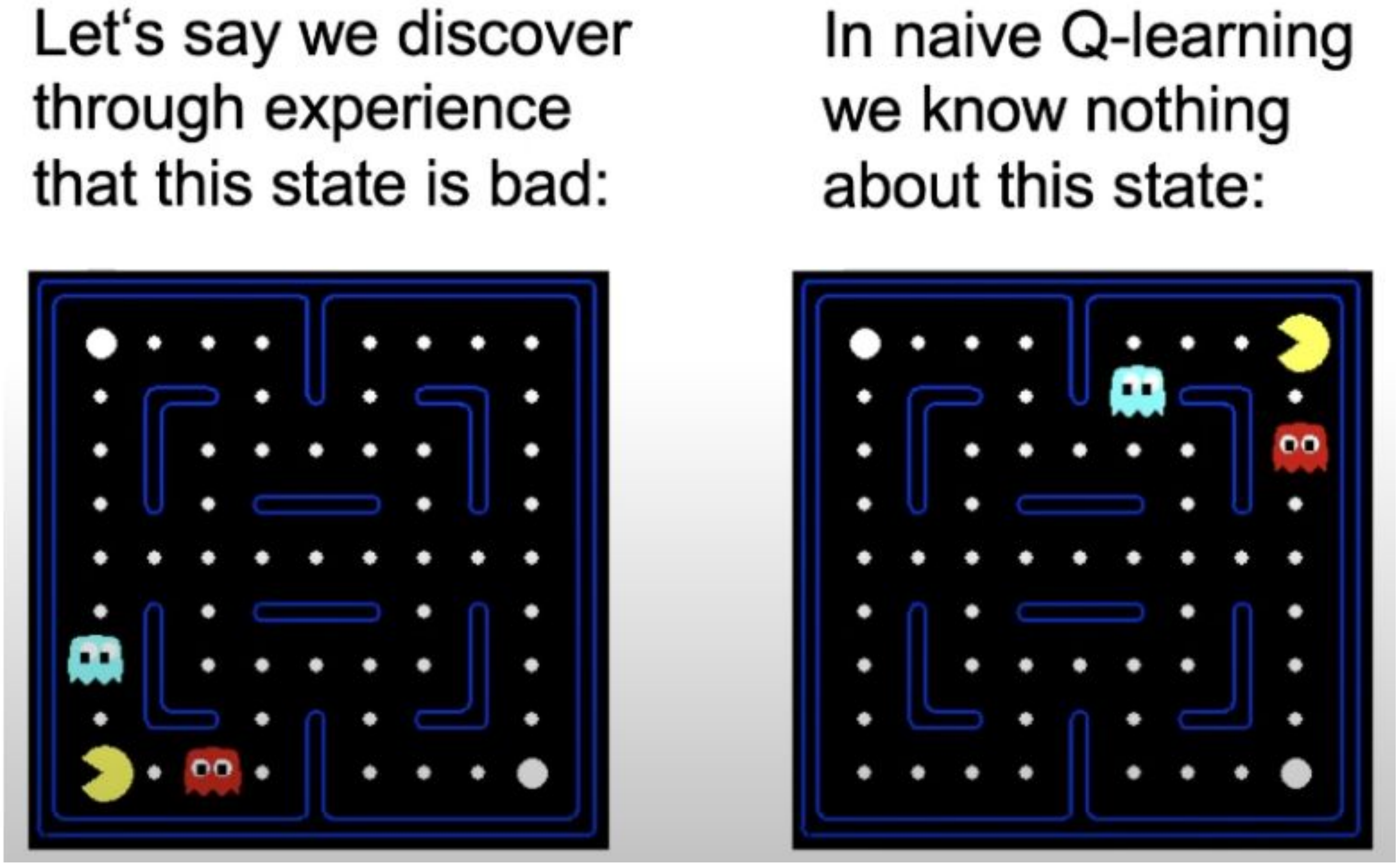
- Linear regression

Nonlinear VFA

- Neural networks
- Tree-based algorithms
- KNN

Постановка задачи

- Обобщение схожих состояний



VFA:

$$\hat{v}(s, \mathbf{w}) \approx v_\pi(s)$$

$$\text{or } \hat{q}(s, a, \mathbf{w}) \approx q_\pi(s, a)$$

Решение:
Представлять текущее состояние среды в виде вектора признаков.

В случае игры pacman признаками могут быть:

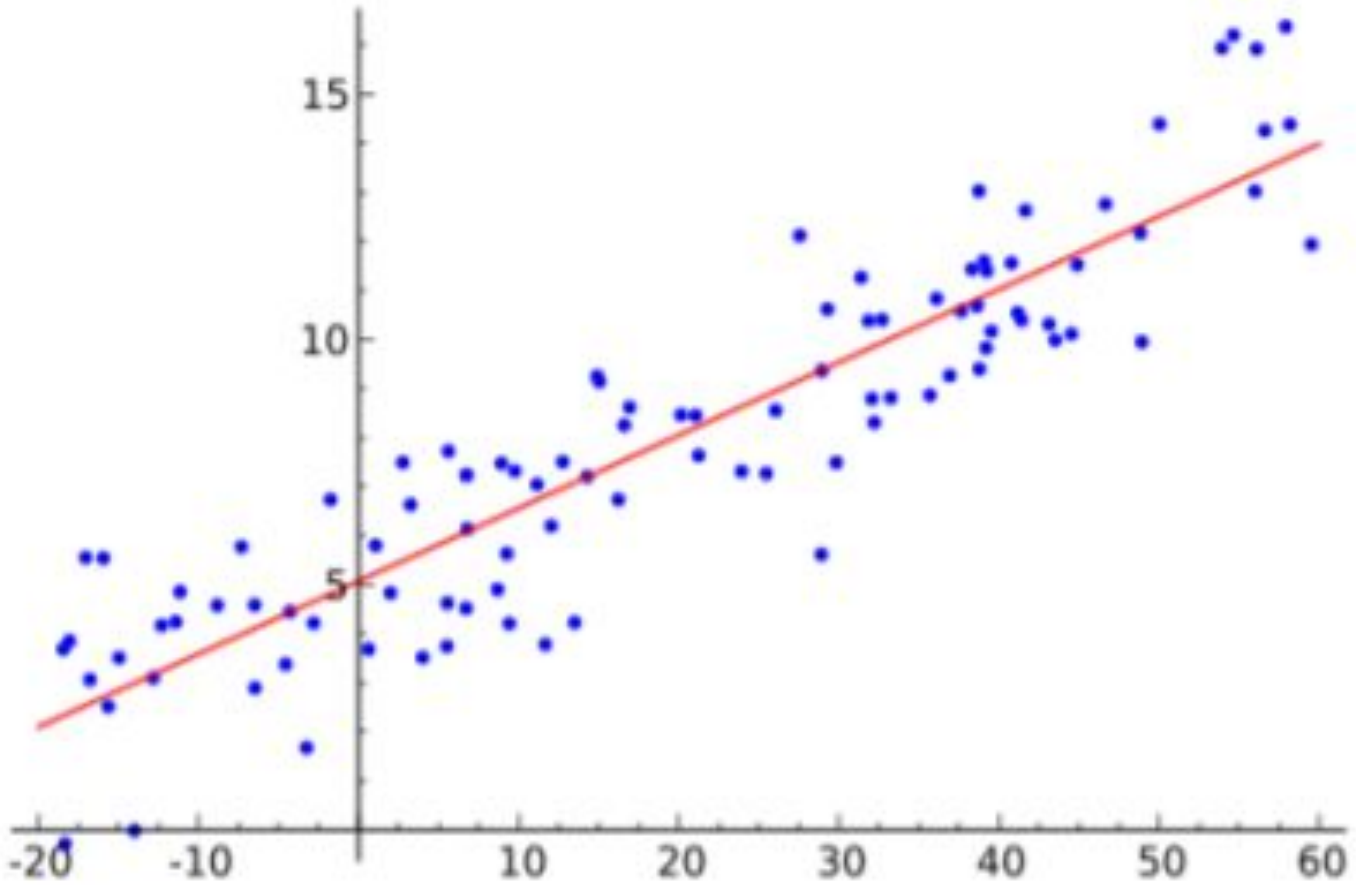
- Расстояние до ближайшего призрака
- Количество призраков
- Расстояние до ближайшей еды

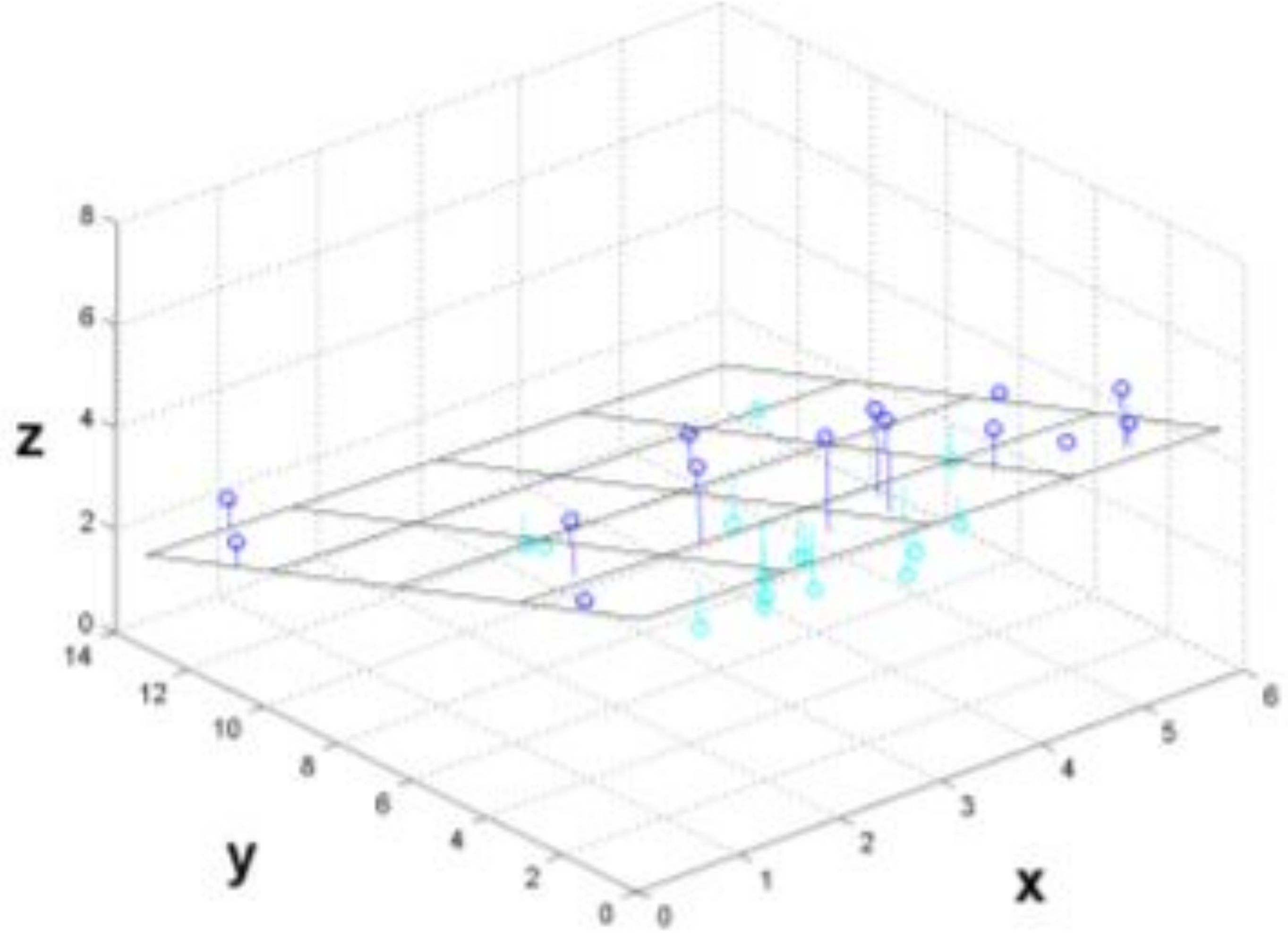
Linear Function Approximation

- The estimate value function:

$$V(s, \theta_t) = \theta_t^\top \phi(s)$$

where $\theta \in \Re^d$ is a vector of parameters, $\phi : \mathcal{S} \mapsto \Re^d$ is a mapping from states to d-dimensional spaces.

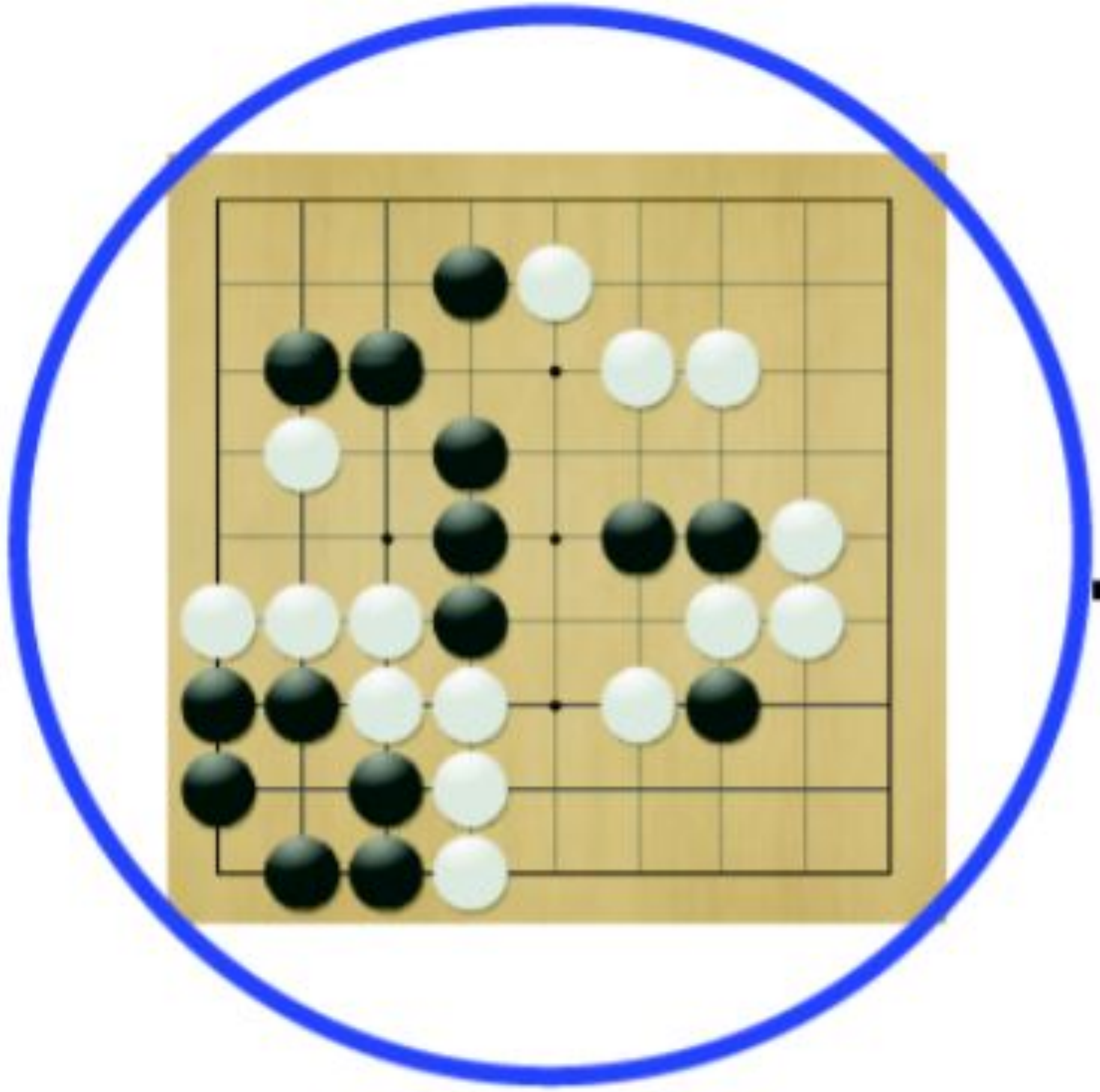




– Examples: polynomial, RBF, fourier, wavelet basis, tile-coding. (suffer from the curse of dimensionality)

Linear Function Approximation

state



s

$$\overset{\text{feature vector}}{\underbrace{\begin{bmatrix}0\\1\\0\\0\\0\\1\\1\\0\end{bmatrix}}_{\phi_s}} \times \overset{\text{parameter vector}}{\underbrace{\begin{bmatrix}0.1\\-2\\0\\0.5\\0\\0\\5\\-.4\end{bmatrix}}_{\theta}} = -2 + 0 + 5 = 3$$

estimated value

(10^{35} states, 10^{5} binary features and parameters.)

Gradient descent in VFA

Gradient descent

$J(w)$ - дифференцируемая функция от параметров w среды s

Обновляем параметры w шагом по антиградиенту

$$\Delta \mathbf{w} = -\frac{1}{2}\alpha \nabla_{\mathbf{w}} J(\mathbf{w})$$

Находим оптимальный вектор параметров w минимизируя MSE между $\hat{V}(s, w)$ и $V_\pi(s)$

$J(w) = E_\pi[(V_\pi(S) - \hat{V}(S, w))^2]$

- Gradient descent finds a local minimum

$$\begin{aligned}\Delta\mathbf{w} &= -\frac{1}{2}\alpha\nabla_{\mathbf{w}}J(\mathbf{w})\\ &= \alpha\mathbb{E}_{\pi}\left[(v_{\pi}(S)-\hat{v}(S,\mathbf{w}))\nabla_{\mathbf{w}}\hat{v}(S,\mathbf{w})\right]\end{aligned}$$

- Stochastic gradient descent *samples* the gradient

$$\Delta\mathbf{w} = \alpha(v_{\pi}(S)-\hat{v}(S,\mathbf{w}))\nabla_{\mathbf{w}}\hat{v}(S,\mathbf{w})$$

- Expected update is equal to full gradient update

Stochastic gradient descent in VFA

$$\mathcal{D} = \{\langle s_1, v_1^\pi \rangle, \langle s_2, v_2^\pi \rangle, ..., \langle s_T, v_T^\pi \rangle\}$$

Берем батч из множества, считаем приближение, обновляем веса по ошибке (MSE)

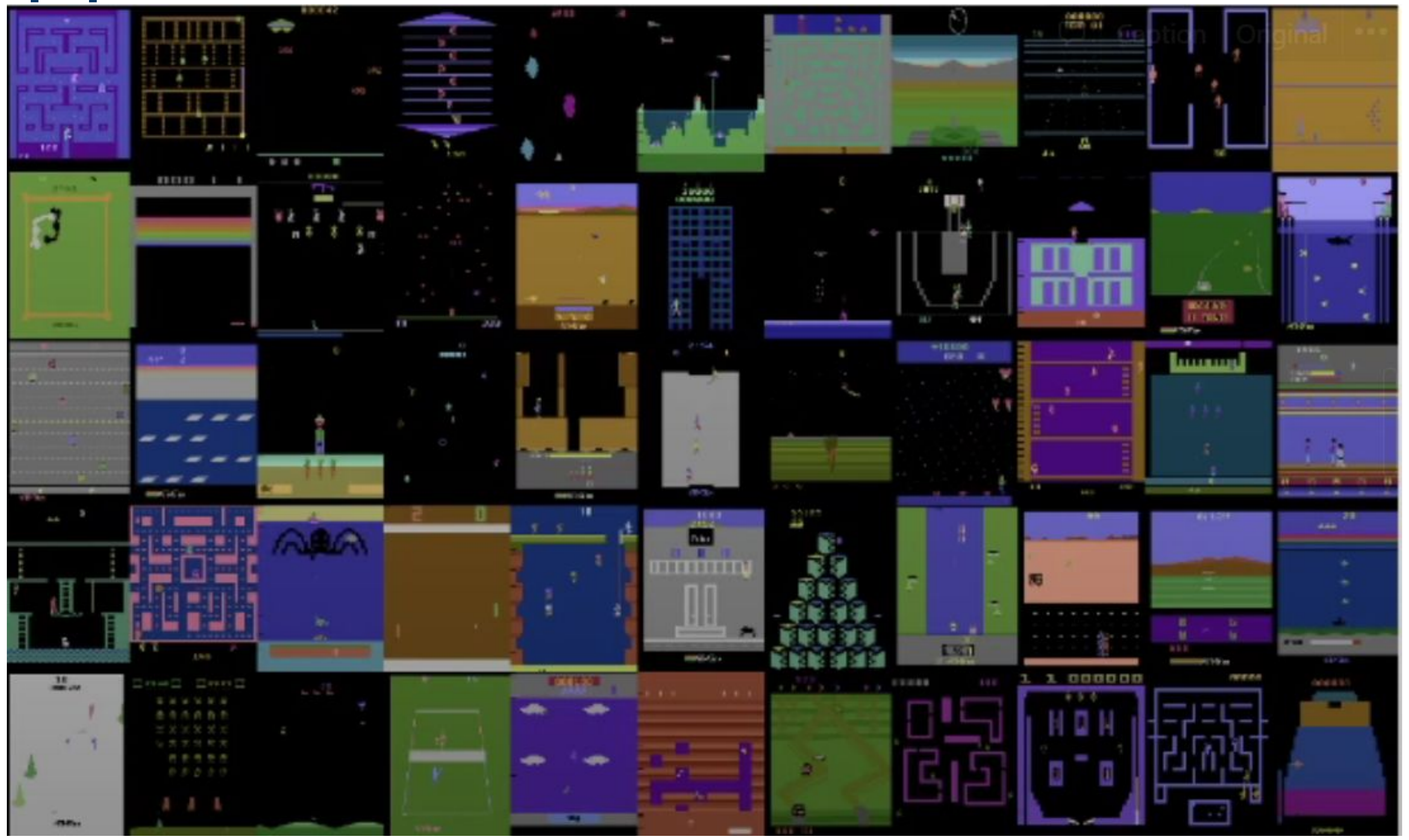
$$\Delta \mathbf{w} = \alpha (v^\pi - \hat{v}(s, \mathbf{w})) \nabla_{\mathbf{w}} \hat{v}(s, \mathbf{w})$$

SGD намного эффективнее обычного метода градиентного спуска. (В RL особенно)

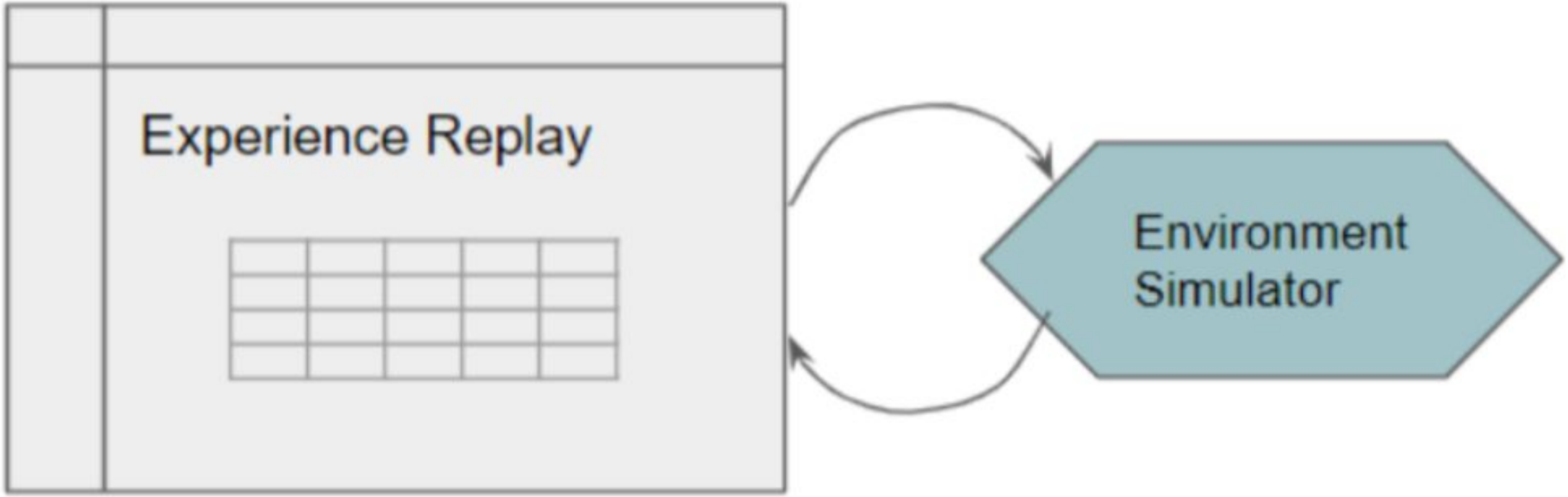
Deep Q-networks (DQN)
Авторы: DeepMind

DQN

- Превзошли человеческий результат в 49 играх Atari
- Никаких признаков. На вход подавалось только изображение с экрана
- Один RL алгоритм на все игры
- Породили новую эру в RL



https://storage.googleapis.com/deepmind-media/dqn/DQNNaturePaper.pdf



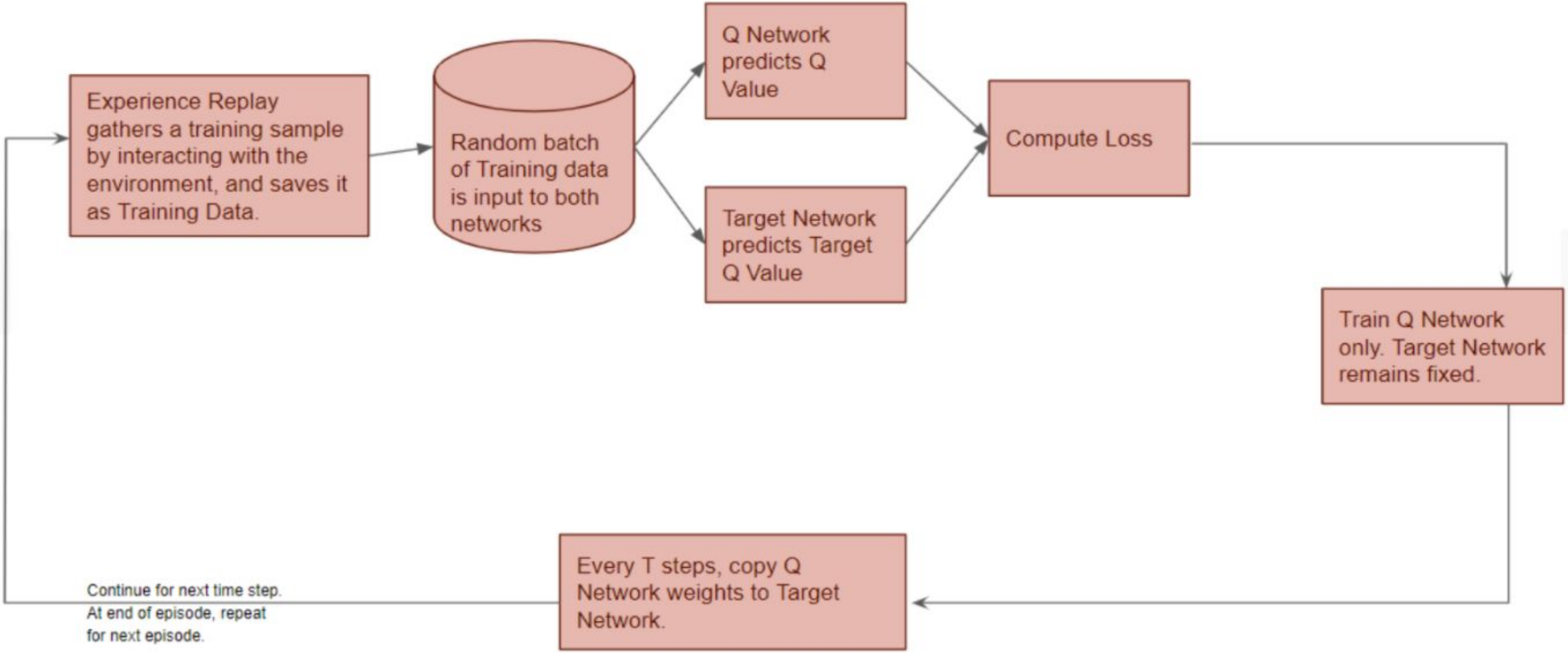
Архитектура DQN

Experience replay

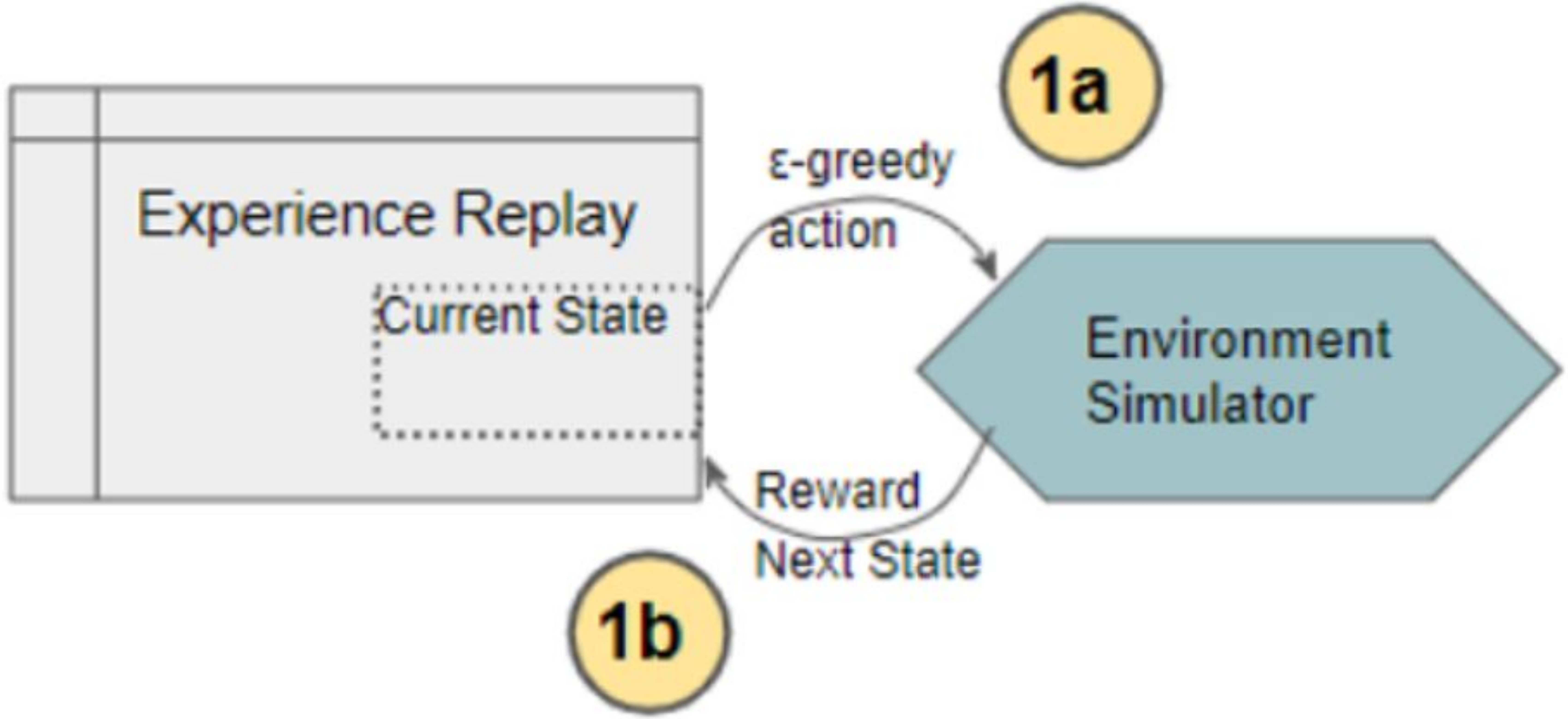
Храним ячейки “опыта” в виде таблицы , где строка это $e_t = (s_t, a_t, r_t, s_{t+1})$

Преимущества expierence replay

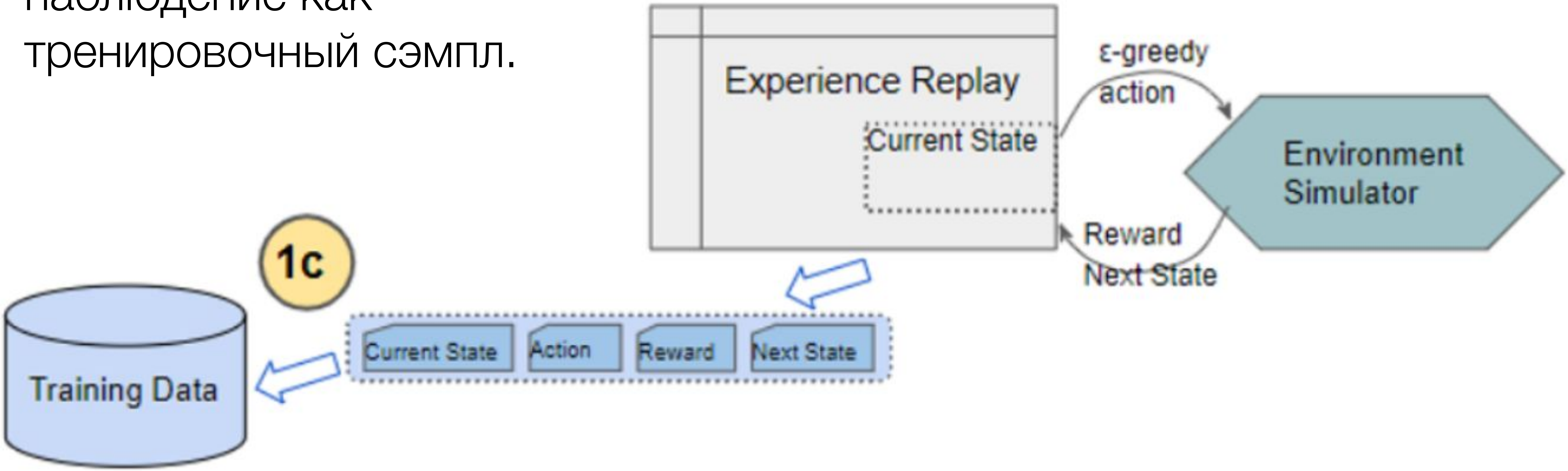
- Повышается эффективность и скорость обучения
- Улучшается сходимость аппроксиматора.


Experience Replay gathers a training sample by interacting with the environment, and saves it as Training Data.
Random batch of Training data is input to both networks
Q Network predicts Q Value
Target Network predicts Target Q Value
Compute Loss
Train Q Network only. Target Network remains fixed.
Every T steps, copy Q Network weights to Target Network.
Continue for next time step. At end of episode, repeat for next episode.

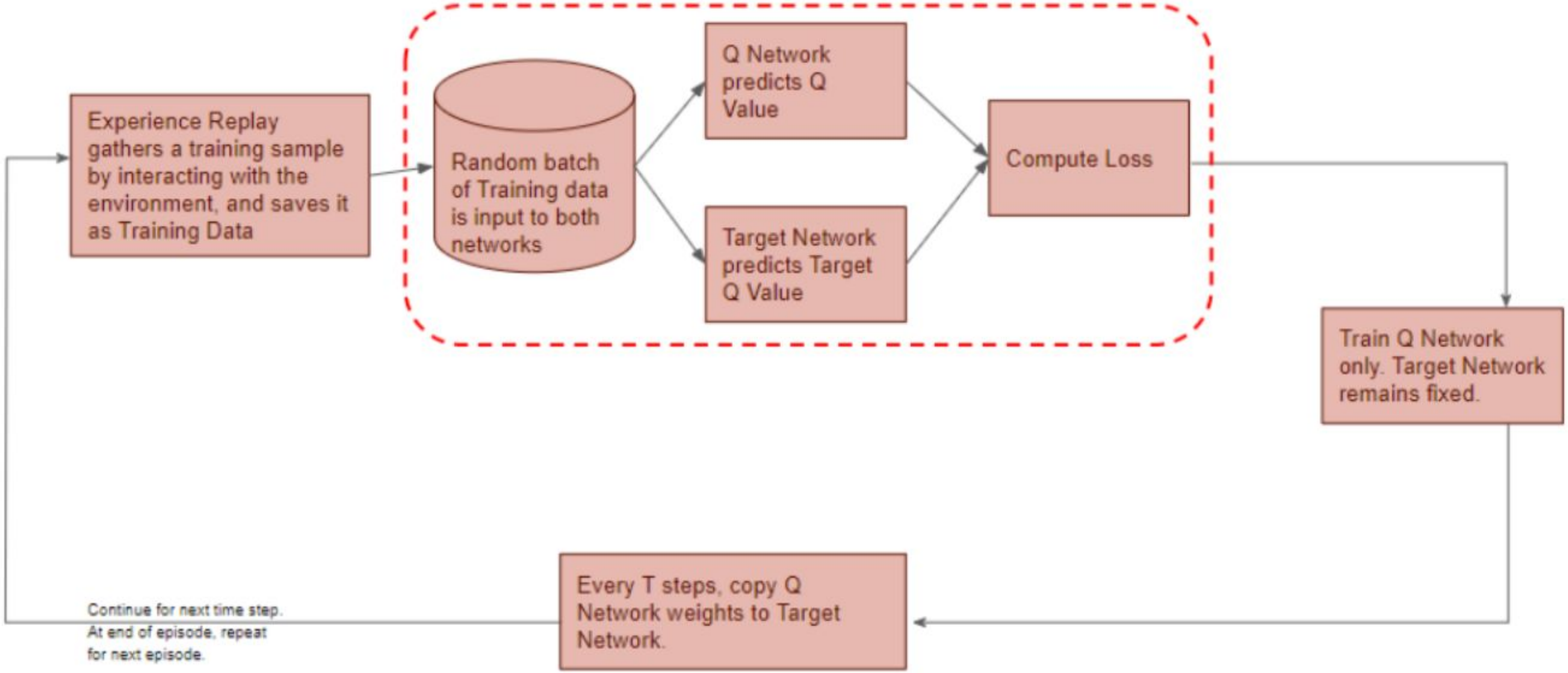
Experience replay для текущего состояния (S) выбирает действие с помощью e-greedy алгоримта (с вероятностью $1-\epsilon$ берем $max(Q(s,a))$), с вероятностью ϵ выбираем действие случайным образом.), выполняет это действие, получает награду r и переходит к следующему состоянию s′



ER сохраняет текущее наблюдение как тренировочный сэмпл.



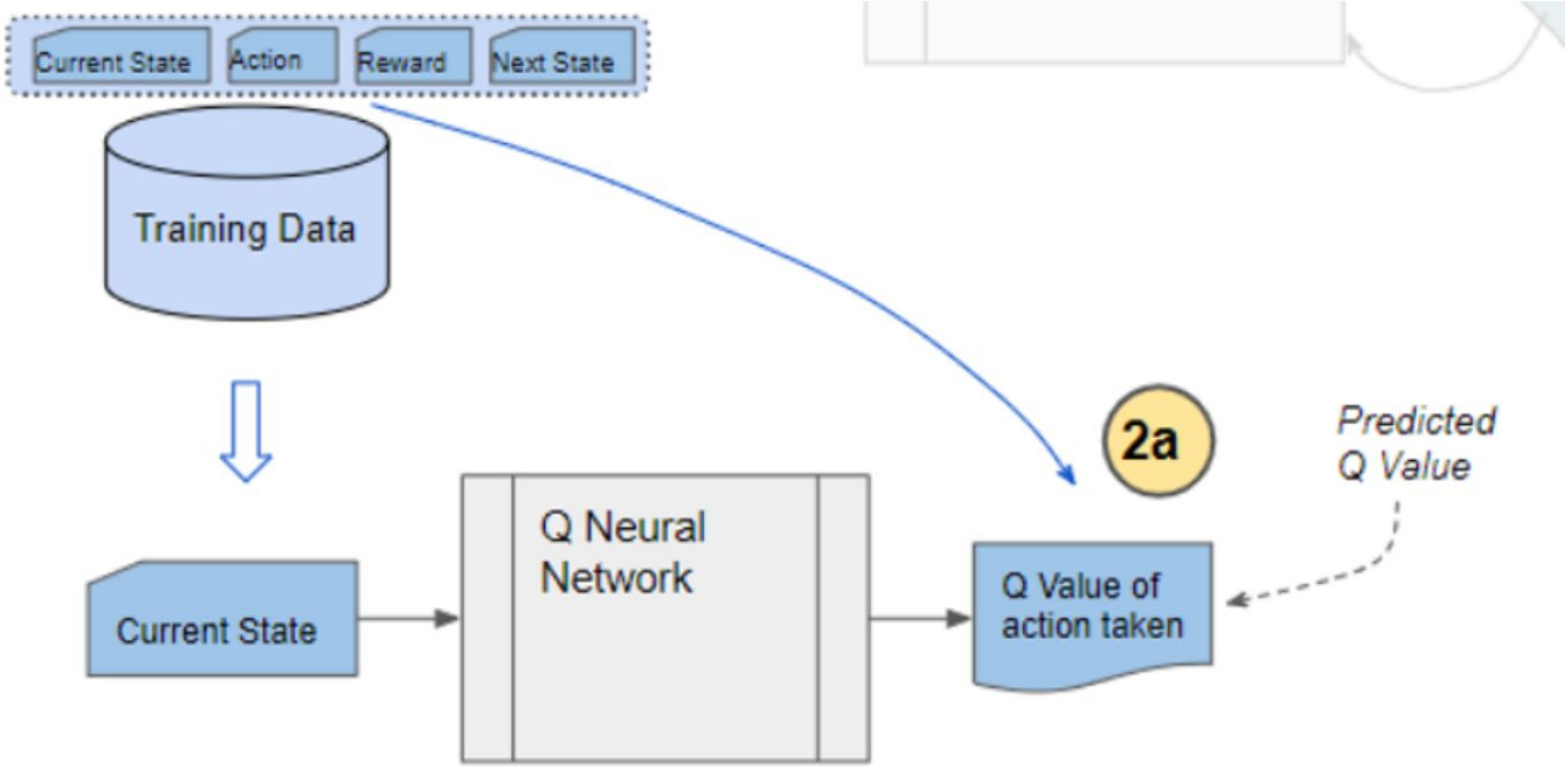
Рассмотрим подробнее выделенную часть модели.



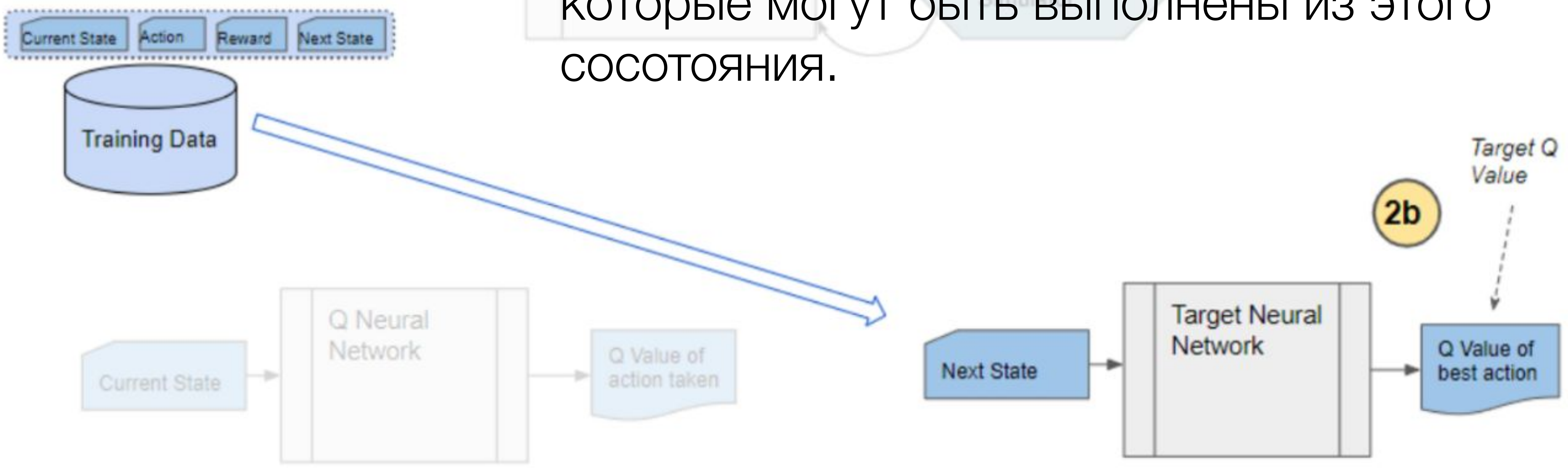
1. Берем случайным образом батч данных из тренировочной выборки

2. Этот батч подаем на вход обеим моделям

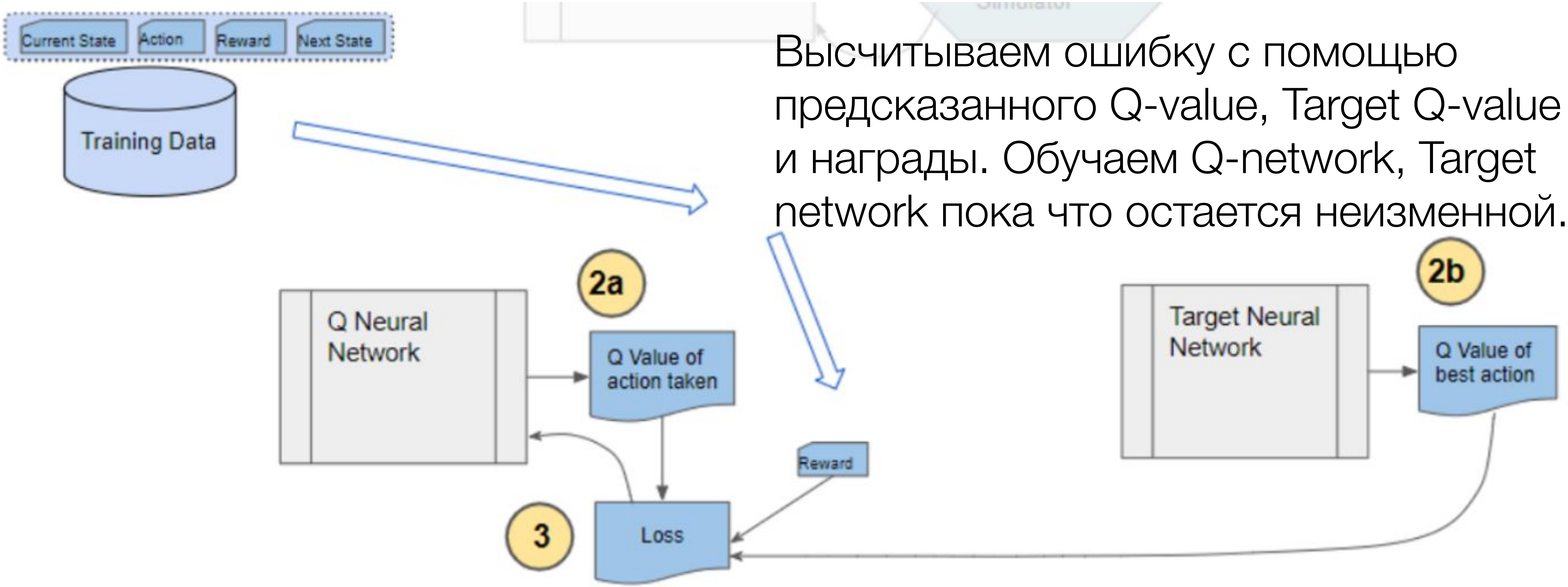
3. Q-network получает состояние среды и действие от каждого объекта и предсказывает Q-value для этого действия.



Target network берет для каждого сэмпла следующее состояние s’ и предсказывает наилучшее Q-value для всех действия которые могут быть выполнены из этого сосотояния.



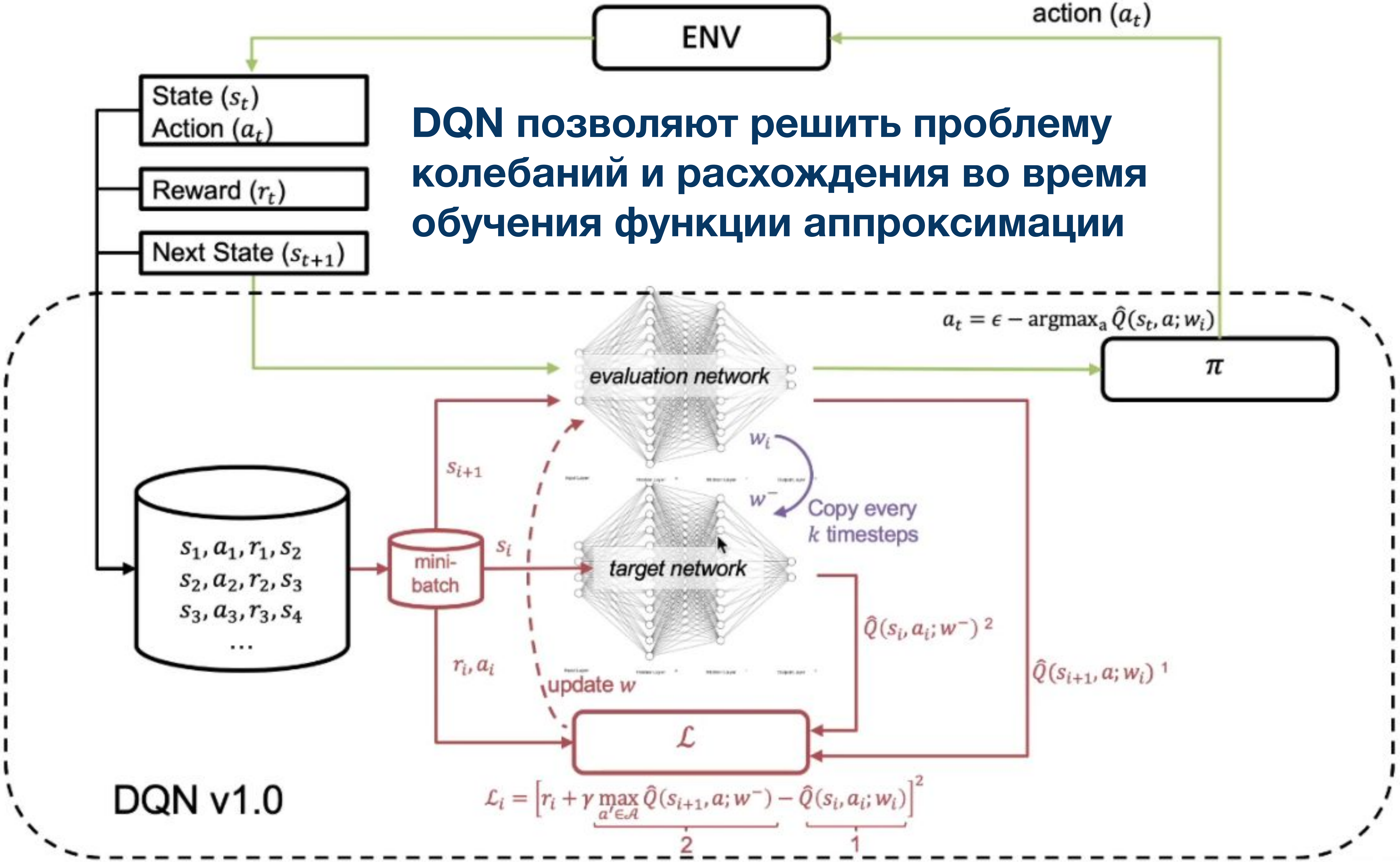
Высчитываем ошибку с помощью предсказанного Q-value, Target Q-value и награды. Обучаем Q-network, Target network пока что остается неизменной.



Через T шагов копируем веса Q-network для Target-network

Это позволяет Target network более точно предсказывать target values, так как мы берем обновленные веса.

Deep Q-Networks (DQNs)



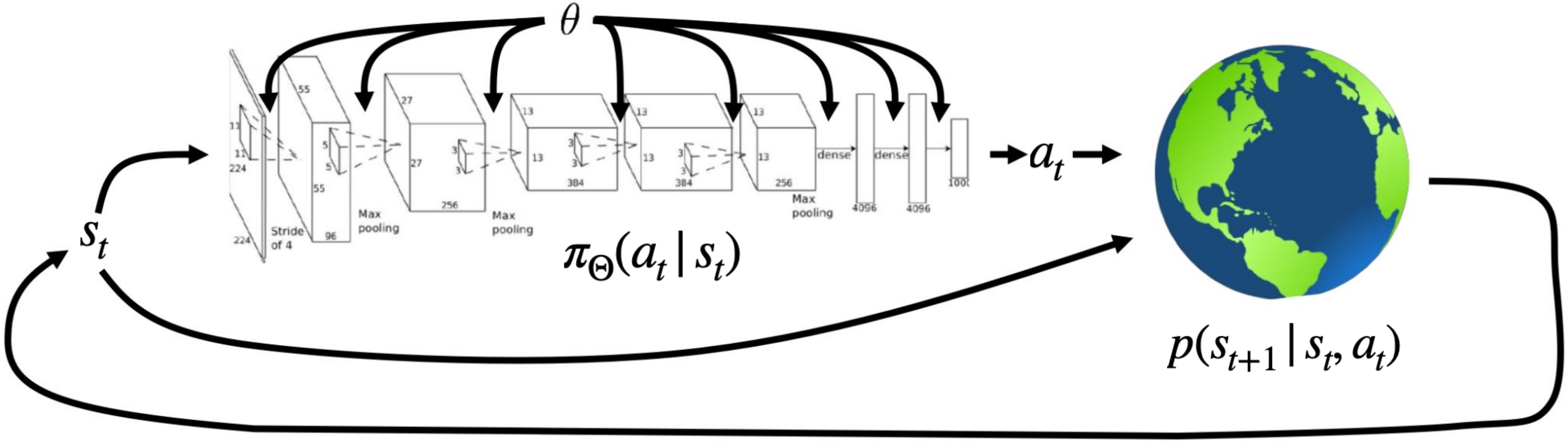
DQN позволяют решить проблему колебаний и расхождения во время обучения функции аппроксимации

Внимание. Спасибо за внимание

Policy Gradient Methods

Вводная информация

Policy Gradient Methods – это методы обучения с подкреплением, которые основываются на оптимизации стратегии (policy) с помощью градиентных методов.



$r(s_t, a_t)$ – награда полученная в результате действия a_t из состояния s_t.

$R_\tau = \sum_t r(s_t, a_t)$ – сумма всех выигрышей

$\tau = [s_1, a_1, .., s_T, a_T]$ – сценарий (последовательность состояний и действий)

Распределение над сценариями

$$p_\Theta(\tau) = p(s_1) \cdot \prod_{t=1}^{T} \pi_\Theta(a_t \mid s_t) \cdot p(s_{t+1} \mid s_t, a_t)$$

Распределение
начальных
состояний

Распределение над сценариями

$$p_\Theta(\tau) = p(s_1) \cdot \prod_{t=1}^{T} \pi_\Theta(a_t \mid s_t) \cdot p(s_{t+1} \mid s_t, a_t)$$

Стратегия

Распределение над сценариями

$$p_\Theta(\tau) = p(s_1) \cdot \prod_{t=1}^{T} \pi_\Theta(a_t \mid s_t) \cdot p(s_{t+1} \mid s_t, a_t)$$

Вероятность
перехода между
состояниями

Основная задача

Мат. ожидание суммы выигрышей по всем сценариям

$$\boxed{\Theta^* = \underset{\Theta}{argmax}\left[\mathbb{E}_{\tau\sim p_\Theta(\tau)}R_\tau\right]}$$

Введем обозначение:

$$J(\Theta) = \mathbb{E}_{\tau\sim p_\Theta(\tau)}R_\tau = \int p_\Theta(\tau)\cdot R_\tau \, d\tau$$

$$\nabla_\Theta J(\Theta) = \int \nabla_\Theta p_\Theta(\tau)\cdot R_\tau d\tau$$

Основная задача

Мат. ожидание суммы выигрышей по всем сценариям

$$\Theta^* = \underset{\Theta}{argmax}\left[\mathbb{E}_{\tau \sim p_\Theta(\tau)} R_\tau\right]$$

Введем обозначение:

$$\boxed{J(\Theta) = \mathbb{E}_{\tau \sim p_\Theta(\tau)} R_\tau = \int p_\Theta(\tau) \cdot R_\tau \, d\tau}$$

$$\nabla_\Theta J(\Theta) = \int \nabla_\Theta p_\Theta(\tau) \cdot R_\tau d\tau$$

Основная задача

Мат. ожидание суммы выигрышей по всем сценариям

$$\Theta^* = \underset{\Theta}{argmax}\left[\mathbb{E}_{\tau\sim p_\Theta(\tau)} R_\tau\right]$$

Введем обозначение:

$$J(\Theta) = \mathbb{E}_{\tau\sim p_\Theta(\tau)} R_\tau = \int p_\Theta(\tau)\cdot R_\tau \, d\tau$$

$$\boxed{\nabla_\Theta J(\Theta) = \int \nabla_\Theta p_\Theta(\tau)\cdot R_\tau d\tau}$$

Log-derivative trick

$$p_\Theta(\tau)\,\nabla_\Theta \log p_\Theta(\tau) = p_\Theta(\tau)\frac{\nabla_\Theta p_\Theta(\tau)}{p_\Theta(\tau)} = \nabla_\Theta p_\Theta(\tau)$$

Закон больших чисел

$$\forall\ \varepsilon > 0 : \ \ P(\,|\,\overline{X} - \mathbb{E}[X_1]\,|\, > \varepsilon) \to 0$$

Вывод оценки

1) Применим log-derivative trick

R_τ – сумма выигрышей

τ – сценарий

$$\nabla_\Theta J(\Theta) = \int \nabla_\Theta p_\Theta(\tau) \cdot R_\tau \, d\tau$$

$$= \int p_\Theta(\tau) \nabla_\Theta \log p_\Theta(\tau) \cdot R_\tau d\tau$$

$$= \mathbb{E}_{\tau \sim p_\Theta(\tau)} \left[\nabla_\Theta \log p_\Theta(\tau) \cdot R_\tau \right]$$

Вывод оценки

2) Найдем $\log p_{\Theta}(\tau)$

Напоминание: $p_{\Theta}(\tau) = p(s_1) \cdot \prod_{t=1}^{T} \pi_{\Theta}(a_t \,|\, s_t) \cdot p(s_{t+1} \,|\, s_t, a_t)$

$$log\ p_{\Theta}(\tau) = log\ p(s_1) + \sum_{t=1}^{T} log\ \pi_{\Theta}(a_t \,|\, s_t) + log\ p(s_{t+1} \,|\, s_t, a_t)$$

Вывод оценки

3) Найдем $\nabla_{\Theta} \log p_{\Theta}(\tau)$

$$\nabla_{\Theta} log\ p_{\Theta}(\tau) = \underbrace{\nabla_{\Theta} log\ p(s_1)}_{=0} + \sum_{t=1}^{T} \nabla_{\Theta} log\ \pi_{\Theta}(a_t \,|\, s_t) + \underbrace{\nabla_{\Theta} log\ p(s_{t+1} \,|\, s_t, a_t)}_{=0}$$

Вывод оценки

4) Применим закон больших чисел

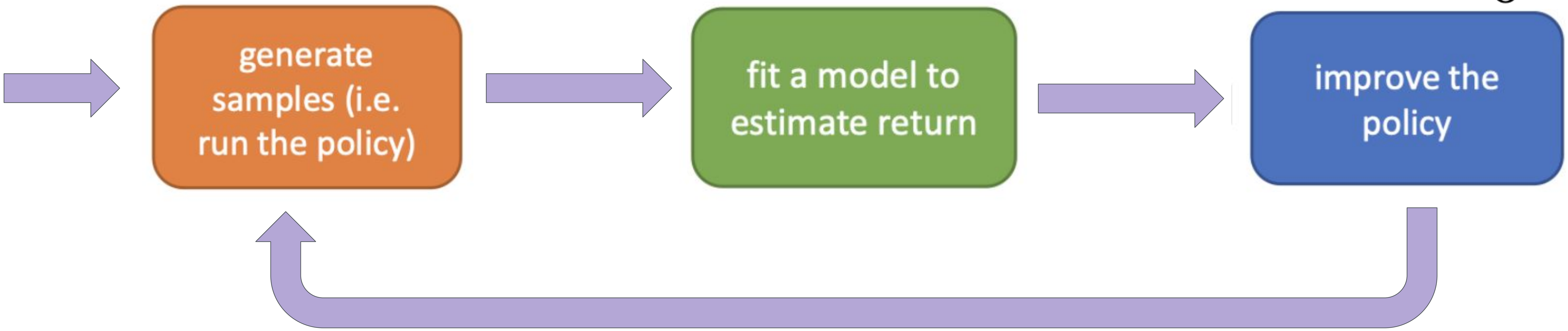
$$\nabla_{\Theta} J(\Theta) = \mathbb{E}_{\tau \sim p_{\Theta}(\tau)} \left[\left(\sum_{t=1}^{T} \nabla_{\Theta} log\ \pi_{\Theta}(a_t \mid s_t) \right) \cdot \underbrace{\left(\sum_{t=1}^{T} r(s_t, a_t) \right)}_{R_\tau} \right]$$

$$\approx \frac{1}{N} \sum_{i=1}^{N} \left[\left(\sum_{t=1}^{T} \nabla_{\Theta} log\ \pi_{\Theta}(a_{i,t} \mid s_{i,t}) \right) \cdot \left(\sum_{t=1}^{T} r(s_{i,t}, a_{i,t}) \right) \right]$$

Схема алгоритма

$$\nabla_\Theta J(\Theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left[\left(\sum_{t=1}^{T}\nabla_\Theta \log\, \pi_\Theta(a_{i,t}\,|\,s_{i,t})\right)\cdot\left(\sum_{t=1}^{T} r(s_{i,t}, a_{i,t})\right)\right]$$

$$\Theta = \Theta + \alpha \nabla_\Theta J(\Theta)$$



Преимущества и недостатки

Преимущества:

1. Легко обобщается на задачи с большим множеством действий, в том числе с непрерывным множеством;
2. Гарантированно сходится хотя бы к локальному максимуму.

Недостатки:

1. Низкая скорость работы
2. Случайная величина $\nabla_\Theta \log p_\Theta(\tau) R_\tau$ имеет большую дисперсию
3. Выборки, собранные для предыдущих значений, никак не переиспользуются.

Off-policy policy gradient method and importance sampling

1) Предположим, что у нас есть выборка из $p_{\Theta}(\tau)$ вместо $p_{\Theta'}(\tau)$

$$J(\Theta') = \int p_{\Theta'}(\tau) \cdot R_\tau \, d\tau$$

$$= \int p_{\Theta}(\tau) \cdot \frac{p_{\Theta'}(\tau)}{p_{\Theta}(\tau)} \cdot R_\tau \, d\tau = \mathbb{E}_{\tau \sim p_{\Theta}(\tau)} \left[\frac{p_{\Theta'}(\tau)}{p_{\Theta}(\tau)} \cdot R_\tau \right]$$

Off-policy policy gradient method and importance sampling

2) Найдем отношение $p_{\Theta'}(\tau)$ и $p_{\Theta}(\tau)$

s_t, s_{t+1} — состояние в момент t и t+1

a_t — действие, выбранное в момент t

π_θ — стратегия (policy)

$$\frac{p_{\Theta'}(\tau)}{p_{\Theta}(\tau)} = \frac{p\left(s_1\right) \prod_{t=1}^{T} \pi_{\Theta'}\left(\mathbf{a}_t \mid \mathbf{s}_t\right) p\left(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t\right)}{p\left(s_1\right) \prod_{t=1}^{T} \pi_{\Theta}\left(\mathbf{a}_t \mid \mathbf{s}_t\right) p\left(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t\right)} = \frac{\prod_{t=1}^{T} \pi_{\Theta'}\left(\mathbf{a}_t \mid \mathbf{s}_t\right)}{\prod_{t=1}^{T} \pi_{\Theta}\left(\mathbf{a}_t \mid \mathbf{s}_t\right)} = \prod_{t=1}^{T} \frac{\pi_{\Theta'}\left(\mathbf{a}_t \mid \mathbf{s}_t\right)}{\pi_{\Theta}\left(\mathbf{a}_t \mid \mathbf{s}_t\right)}$$

Off-policy policy gradient method and importance sampling

3) Вывод оценки

$$J(\Theta') = \mathbb{E}_{\tau \sim p_{\Theta}(\tau)} \left[\frac{p_{\Theta'}(\tau)}{p_{\Theta}(\tau)} \cdot R_{\tau} \right]$$

Off-policy policy gradient method and importance sampling

3) Вывод оценки

$$\nabla_{\Theta'} J(\Theta') = \mathbb{E}_{\tau \sim p_{\Theta}(\tau)} \left[\frac{\nabla_{\Theta'} p_{\Theta'}(\tau)}{p_{\Theta}(\tau)} \cdot R_\tau \right] = \mathbb{E}_{\tau \sim p_{\Theta}(\tau)} \left[\frac{p_{\Theta'}(\tau) \nabla_{\Theta'} log\ p_{\Theta'}(\tau)}{p_{\Theta}(\tau)} \cdot R_\tau \right]$$

$$= \mathbb{E}_{\tau \sim p_{\Theta}(\tau)} \left[\left(\prod_{t=1}^{T} \frac{\pi_{\Theta'}(a_t | s_t)}{\pi_{\Theta}(a_t | s_t)} \right) \cdot \left(\sum_{t=1}^{T} \nabla_{\Theta'} log\ \pi_{\Theta'}(a_t | s_t) \right) \cdot \left(\sum_{t=1}^{T} r(s_t, a_t) \right) \right]$$

Off-policy policy gradient method and importance sampling

3) Вывод оценки

$$\nabla_{\Theta'} J(\Theta') = \mathbb{E}_{\tau \sim p_\Theta(\tau)} \left[\frac{\nabla_{\Theta'} p_{\Theta'}(\tau)}{p_\Theta(\tau)} \cdot R_\tau \right] = \mathbb{E}_{\tau \sim p_\Theta(\tau)} \left[\boxed{\frac{p_{\Theta'}(\tau)}{p_\Theta(\tau)}} \cdot \nabla_{\Theta'} log\ p_{\Theta'}(\tau) \cdot R_\tau \right]$$

$$= \mathbb{E}_{\tau \sim p_\Theta(\tau)} \left[\left(\prod_{t=1}^{T} \frac{\pi_{\Theta'}(a_t | s_t)}{\pi_\Theta(a_t | s_t)} \right) \cdot \left(\sum_{t=1}^{T} \nabla_{\Theta'} log\ \pi_{\Theta'}(a_t | s_t) \right) \cdot \left(\sum_{t=1}^{T} r(s_t, a_t) \right) \right]$$

Список источников

1. http://rail.eecs.berkeley.edu/deeprlcourse/static/slides/lec-5.pdf
2. https://neerc.ifmo.ru/wiki/index.php?title=Методы_policy_gradient_и_алгоритм_асинхронного_актора-критика
3. http://www.scholarpedia.org/article/Policy_gradient_methods
4. https://jonathan-hui.medium.com/rl-policy-gradients-explained-advanced-topic-20c2b81a9a8b
5. https://www.youtube.com/watch?v=5P7I-xPq8u8&t=933s