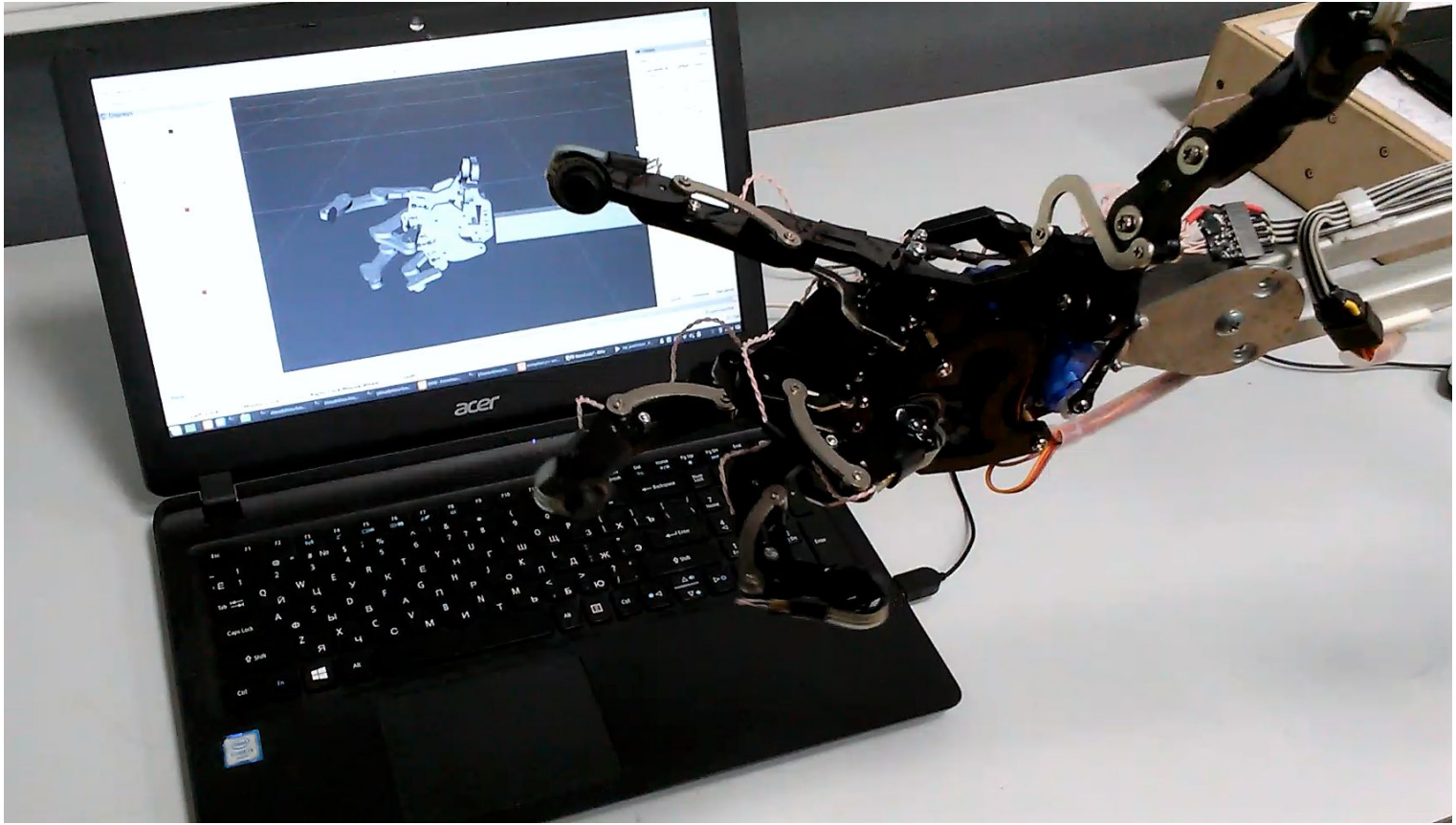
Машинное обучение антропоморфных робототехнических систем на примере модели роботизированной руки с тактильной обратной связью

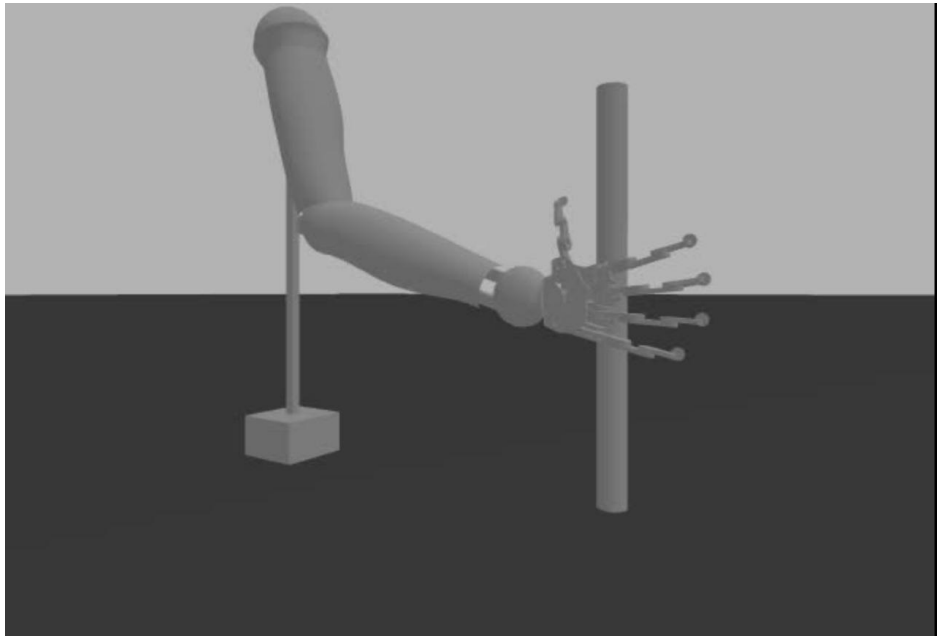
Выполнил: студент 202 гр. Прихно Мария Александровна
Научный руководитель: кандидат физ.-мат. наук Запуниди Сергей Александрович

Московский государственный университет имени М.В. Ломоносова,
Физический факультет
Москва — 2020 год



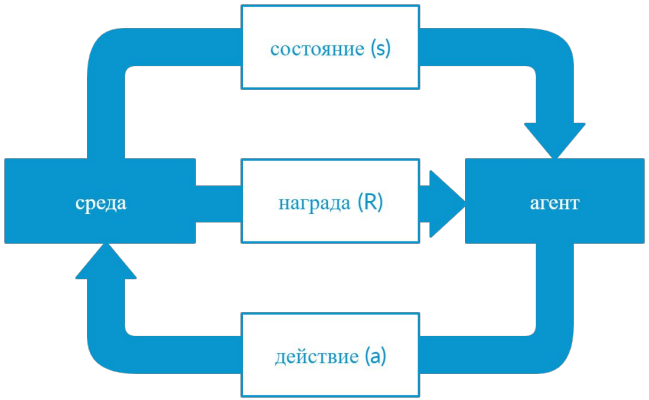
6 степеней подвижности: 5 сгибов пальцев и одна переносная в локте. 6 тензометрических датчиков на кончиках пальцев и посередине ладони.

Задача робота: опираясь только на тактильную обратную связь, захватить помещенный в рабочую зону предмет.



1. поиск предмета в рабочей зоне
2. устойчивый захват

Обучение с подкреплением



OpenAI

\+

`openai_ros`

Базовые понятия

Марковский процесс принятия решений:

- пространство состояний S
- пространство действий A
- вероятность перехода p(s’ | s, a)
- функция награды R(r’ | s, a)
- скидка 0 < ɣ < 1

- стратегия π:

$$\pi(s) : S \rightarrow A$$

- цель агента — максимизация суммарной награды:

$$\sum_{t=0}^{T} \gamma^t r_t$$

Базовые понятия

- функция ценности состояния V(s):

$$V^{\pi}(s) = \mathbb{E}[\sum_{t=0}^{T} \gamma^t r_t] \quad \forall s \in S$$

- существует оптимальная функция ценности и оптимальная стратегия

$$V^*(s) = \max_{\pi} V^{\pi}(s) \quad \forall s \in S$$

$$\pi^* = arg\max_{\pi} V^{\pi}(s) \quad \forall s \in S$$

- функция ценности действия Q(s, a):

$$Q : S \times A \rightarrow R$$

- оптимальная Q определяет, насколько выгодно агенту выбрать действие a из состояния s

$$V^*(s) = \max_{a} Q^*(s, a)$$

$$\pi^*(s) = arg\max_{a} Q^*(s, a) \quad \forall s \in S$$

Уравнение Беллмана

$$Q^*(s,a) = R(s,a) + \gamma \mathbb{E}_{s'}[V^*(s')]$$

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s' \in S} p(s' \mid s,a) V^*(s')$$

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

$$V^*(s) = \max_a [R(s,a) + \gamma \sum_{s' \in S} p(s' \mid s,a) V^*(s')]$$

- value iteration

Initialize $V(s)$ to arbitrary values
Repeat
  For all $s \in S$
    For all $a \in \mathcal{A}$
      $Q(s,a) \leftarrow E[r|s,a] + \gamma \sum_{s' \in S} P(s'|s,a) V(s')$
    $V(s) \leftarrow \max_a Q(s,a)$
Until $V(s)$ converge

- policy iteration

Initialize a policy π' arbitrarily
Repeat
  $\pi \leftarrow \pi'$
  Compute the values using π by
    solving the linear equations
      $V^{\pi}(s) = E[r|s,\pi(s)] + \gamma \sum_{s' \in S} P(s'|s,\pi(s)) V^{\pi}(s')$
  Improve the policy at each state
    $\pi'(s) \leftarrow \arg\max_a (E[r|s,a] + \gamma \sum_{s' \in S} P(s'|s,a) V^{\pi}(s'))$
Until $\pi = \pi'$

Алгоритмы

- Q-learning
- DQN (Deep Q-Network)
- DDPG (Deep Deterministic Policy Gradient)

Q-learning

$$Q_{i+1}(s,a) = (1-\alpha)Q_i(s,a) + \alpha Q_{obs}(s,a)$$

$$Q_{obs}(s,a) = R_i(s,a) + \gamma \max_{a'} Q_i(s',a')$$

DQN (Deep Q-Network)

$Q(s, a; \theta) \approx Q^*(s, a)$ — нейронная сеть Q-сеть с весами θ

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

$y_i = \mathbb{E}_{s,a;s'}[(R + \gamma \max_{a'} Q(s', a'; \theta_i)) \mid s, a]$ Дифференцируем по весам θ:

$$\nabla_\theta L_i(\theta_i) = \mathbb{E}_{s,a;s'}[(R+\gamma \max_{a'} Q(s', a'; \theta_i) - Q(s', a'; \theta_i)) \nabla_{\theta_i} Q(s, a; \theta_i)]$$

- \+ experience replay

DQN (Deep Q-Network)

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

Mnih V. et al. Playing atari with deep reinforcement learning //arXiv preprint arXiv:1312.5602. – 2013.

DDPG (Deep Deterministic Policy Gradient)

$\max_a Q^*(s,a) \approx Q(s,\mu(s))$ DDPG — Q-learning для непрерывных пространств действий

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

$Q(s,a \mid \theta^q) \approx Q^*(s,a)$ — функция-аппроксиматор нейронная сеть-критик

В среде берется набор D переходов (s, a, r, s', d), d=1, если состояние конечное, и d=0, если нет.

Среднеквадратичная ошибка Беллмана:

$$L(\theta^q, D) = \mathbb{E}_{(s,a,r,s',d) \sim D}[(Q(s,a \mid \theta^q) - (r + \gamma(1-d) \max_{a'} Q(s',a' \mid \theta^q)))^2]$$

DDPG (Deep Deterministic Policy Gradient)

$r + \gamma(1 - d) \max_{a'} Q(s', a' \mid \theta^q)$ — цель, зависит от обучаемых параметров.

Используется сеть-цель $Q'(s, a \mid \theta^{q'})$, отстающая от $Q(s, a \mid \theta^q)$. Обновляется единожды за цикл обучения:

$$\theta^{q'}_{i+1} = \tau\theta^q_i + (1 - \tau)\theta^{q'}_i$$

Для расчета максимума Q используется сеть-цель актера $\mu'(s \mid \theta^{\mu'})$, тогда ошибка:

$$L(\theta^q, D) = \mathbb{E}_{(s,a,r,s',d)\sim D}[(Q(s, a \mid \theta^q) - (r + \gamma(1 - d)Q'(s, \mu'(s \mid \theta^{\mu'}) \mid \theta^{q'})))^2]$$

Для поиска стратегии $\mu(s \mid \theta^\mu)$, максимизирующей $Q(s, a \mid \theta^q)$, достаточно совершить восхождение по градиенту:

$$\max_\theta \mathbb{E}_{s\sim D}[Q(s, \mu(s \mid \theta^\mu) \mid \theta^q)]$$

DDPG (Deep Deterministic Policy Gradient)

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

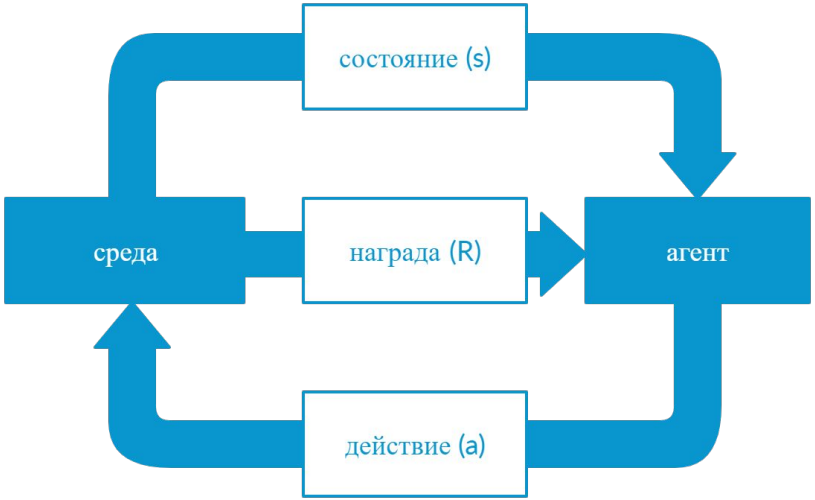
$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end for
end for

Lillicrap T. P. et al. Continuous control with deep reinforcement learning //arXiv preprint arXiv:1509.02971. – 2015.



DDPG

Deep Deterministic Policy Gradient

- состояние s — вектор значений с датчиков давления, 0 < s < 1 [Н]
- действие a — вектор скоростей пальцев, -10 < a < 10 [рад/с]
- награда R
  - симметричная — сумма значений с датчиков давления

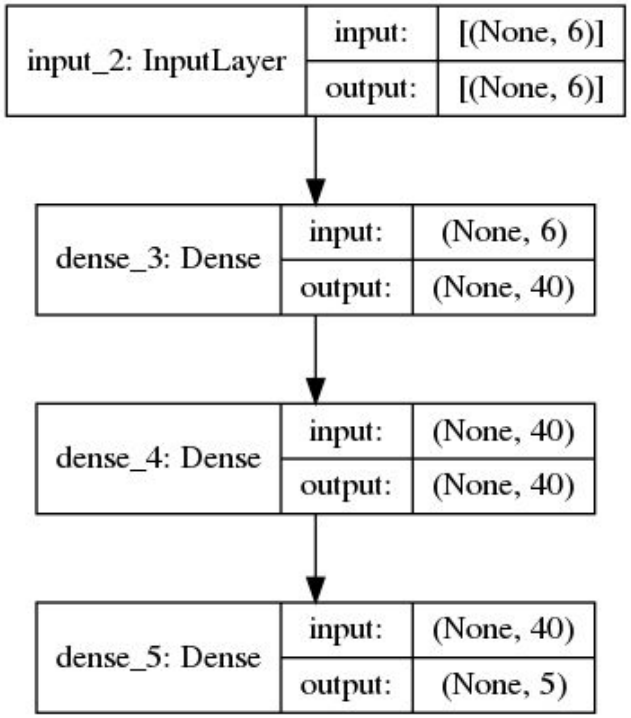
$$R = \sum_{i=1}^{6} s_i$$

  - асимметричная — сумма значений с датчиков давления с поправкой на расстояние до цилиндра

$$R = \sum_{i=1}^{6} s_i - \theta \sum_{i=1}^{5} (\pi - x_i)$$

актер + копия

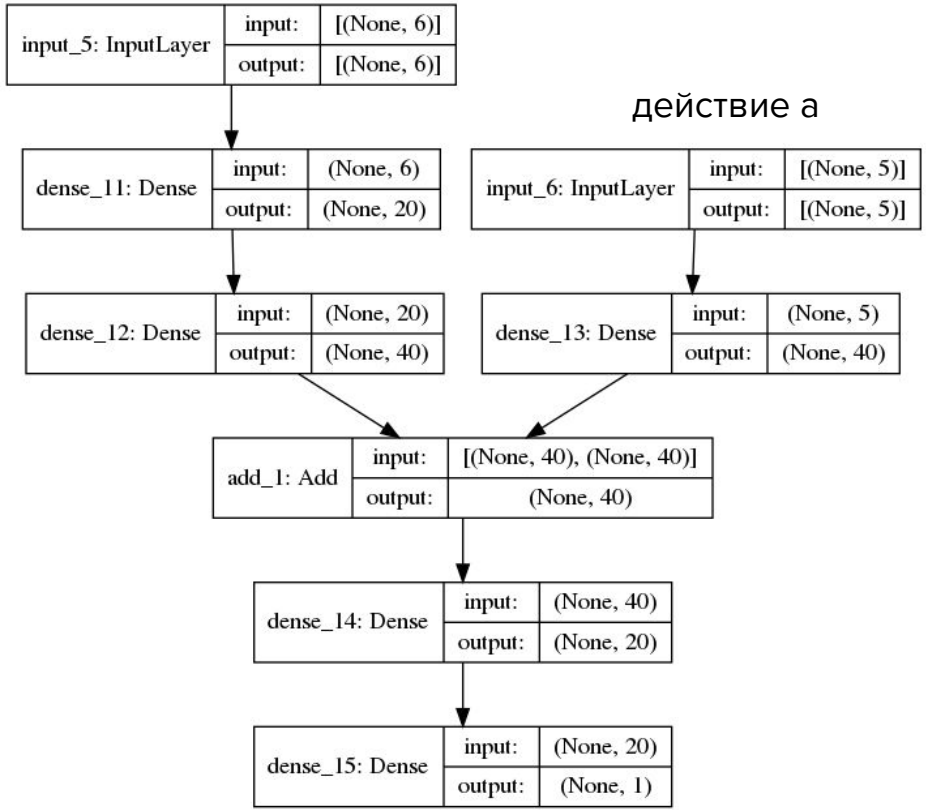
состояние s



действие a

критик + копия

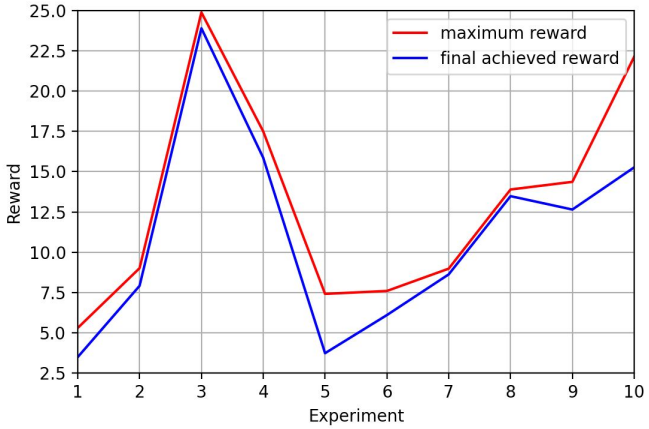
состояние s



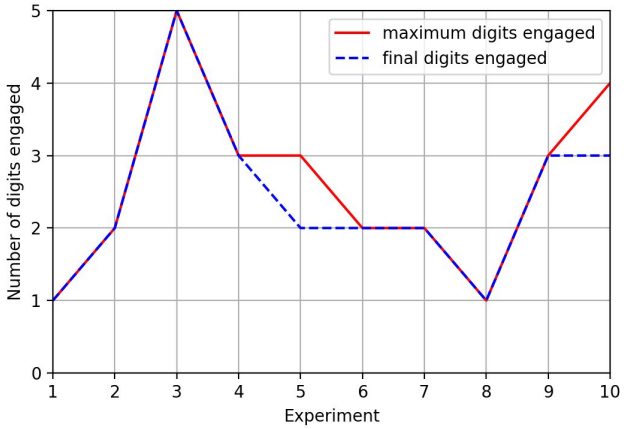
награда r

Результаты

А.



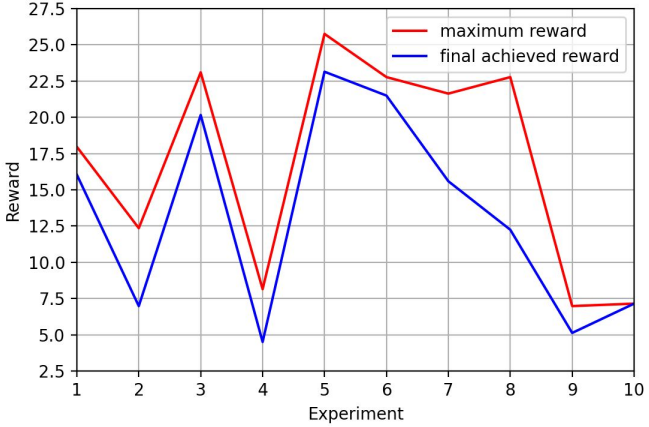
Б.



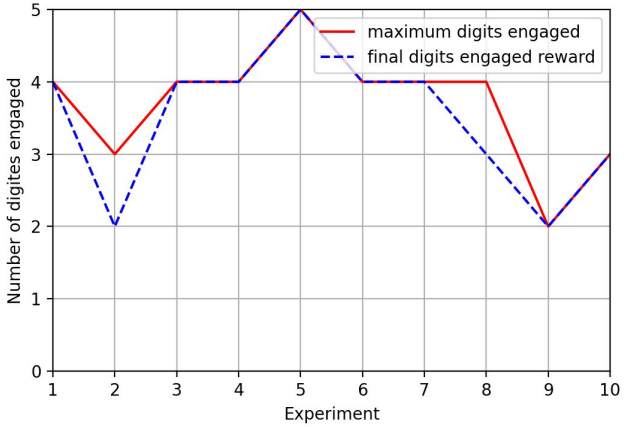
симметричная награда

$$R = \sum_{i=1}^{6} s_i$$

А.



Б.



асимметричная награда, θ=0.005

$$R = \sum_{i=1}^{6} s_i - \theta \sum_{i=1}^{5} (\pi - x_i)$$

Результаты:

- построена среда обучения с подкреплением
- проверена возможность обучения на стартовой задаче; использование асимметричной награды привело к лу́чшим результатам

- по данной работе уже существует публикация:

Прихно М.А., Бегишев Р.Р., Злобин Д.В. Технология цифрового двойника как инструмент обучения роборуки: программно-аппаратная архитектура, задачи и проблемы. // Ломоносов 2020: секция вычислительной математики и кибернетики (Москва) - Москва, 2020. - с.117-120

https://youtu.be/qabHxDTHEYw