Отчёт по домашнему заданию №1

DRL Course 2024

Cross-Entropy Method

Задание 1.

Пользуясь алгоритмом Кросс-Энтропии обучить агента решать задачу Taxi-v3 из Gym. Исследовать гиперпараметры алгоритма и выбрать лучшие.

Алгоритм 5: Cross Entropy Method

Гиперпараметры: $\pi(a \mid s, \theta)$ — стратегия с параметрами θ, N — число сэмплов, M — порог отбора

Инициализируем θ_0 произвольно.
На k-ом шаге:

1. сэмплируем N траекторий $\mathcal{T}_1 \dots \mathcal{T}_N$ игр при помощи стратегии $\pi(a \mid s, \theta_k)$
2. считаем кумулятивные награды $R(\mathcal{T}_i)$
3. сортируем значения: $R_{(1)} \leq R_{(2)} \leq \dots \leq R_{(N)}$
4. полагаем $\gamma_k := R_{(M)}$
5. решаем задачу оптимизации:

$$\theta_{k+1} \leftarrow \operatorname*{argmax}_{\theta} \frac{1}{N} \sum_{j=1}^{N} \mathbb{I}[R(\mathcal{T}_j) \geq \gamma_k] \sum_{s,a \in \mathcal{T}_j} \log \pi(a \mid s, \theta)$$

1. Состояния (States)

- Всего состояний: 500.
- Каждое состояние состоит из:
  - Позиция такси: такси может находиться в одной из 25 возможных позиций (5x5 сетка).
  - Местоположение пассажира: пассажир может находиться в одном из 5 мест (четыре фиксированные точки и внутри такси).
  - Местоположение пункта назначения: 4 возможных пункта назначения.

2. Действия (Actions)

Всего действий: 6. Это дискретное пространство действий, включающее следующие команды:

- 0: перемещение на юг.
- 1: перемещение на север.
- 2: перемещение на восток.
- 3: перемещение на запад.
- 4: забрать пассажира.
- 5: высадить пассажира.

3. Награды (Rewards)

- +20: если пассажир был успешно доставлен в пункт назначения.
- -1: за каждое перемещение такси (действие).
- -10: за нелегальные действия (например, попытка забрать пассажира, когда его нет, или высадка в неверном месте).

4. Цель

Цель агента — максимально быстро забрать пассажира и доставить его в нужное место, минимизируя штрафы за ненужные действия.

5. Пространство действий и состояний

Пространство действий (Action space): Discrete(6) (всего 6 возможных действий).

Пространство состояний (Observation space): Discrete(500) (состояние кодируется как одно число из 500 возможных комбинаций позиций такси, пассажира и пункта назначения).

6. Правила переходов (Transitions)

Среда является детерминированной, то есть любое действие всегда приводит к ожидаемому результату (например, команда "переместиться на север" переместит такси на клетку севернее, если она доступна).

7. Особенности

Агент должен управлять последовательностью действий так, чтобы не допускать неправильной высадки или бесцельных перемещений.

Такси может двигаться только по ячейкам в пределах 5x5 сетки и не может выходить за её пределы.

Первое обучение проводилось с гиперпараметрами:
q_param = 0.9
iteration_n = 20
trajectory_n = 50

В начале обучения:

```
iteration: 0 mean total reward: -767.76
new_model[state][action]: 1.0, action: 1, state: 227
new_model[state][action]: 1.0, action: 0, state: 127
new_model[state][action]: 1.0, action: 5, state: 227
```

В конце:

```
new_model[state][action]: 1.0, action: 5, state: 479
deprecation(
total reward: -370
model:
[[0.16666667 0.16666667 0.16666667 0.16666667 0.16666667 0.16666667]
[0.16666667 0.16666667 0.16666667 0.16666667 0.16666667 0.16666667]
[0.16666667 0.16666667 0.16666667 0.16666667 0.16666667 0.16666667]
...
[0.16666667 0.16666667 0.16666667 0.16666667 0.16666667 0.16666667]
[0.11111111 0.11111111 0.18518519 0.18518519 0.18518519 0.22222222]
[0.16666667 0.16666667 0. 0. 0.33333333 0.33333333]]
```

Таким образом, модели не хватило итераций для обучения (либо траекторий). Сначала попробуем увеличить кол-во итераций до 40 и увеличим квантиль до 0.95 для сокращения отбора “элитных траекторий”

Получен график для средней награды и (сомнительный график) для поиска номеров элитных траекторий:

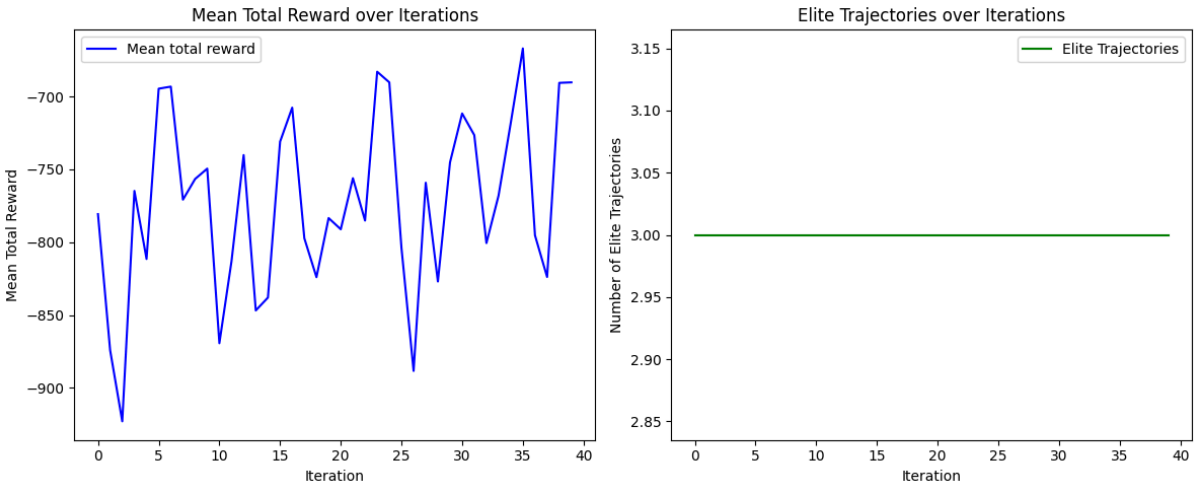
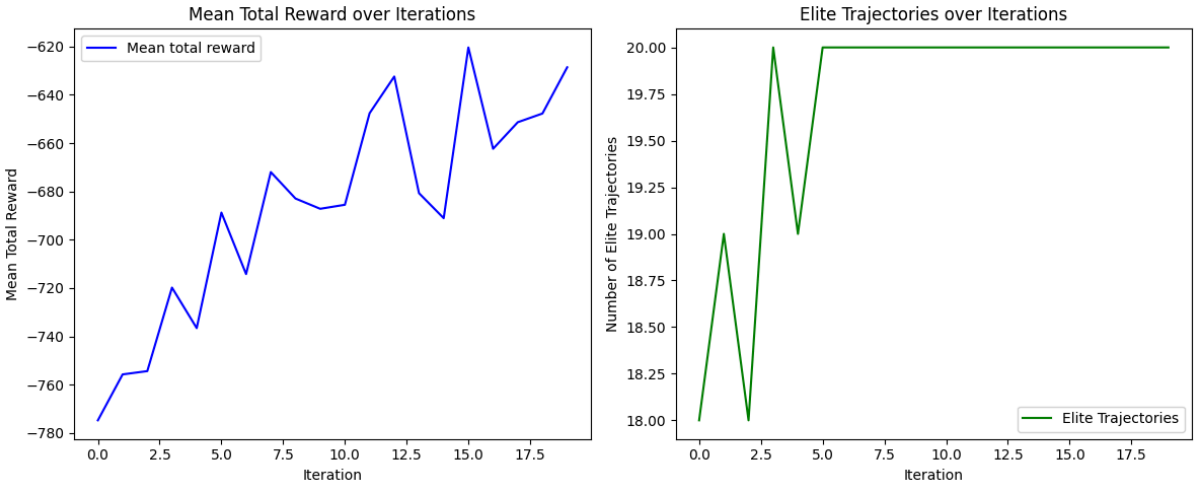
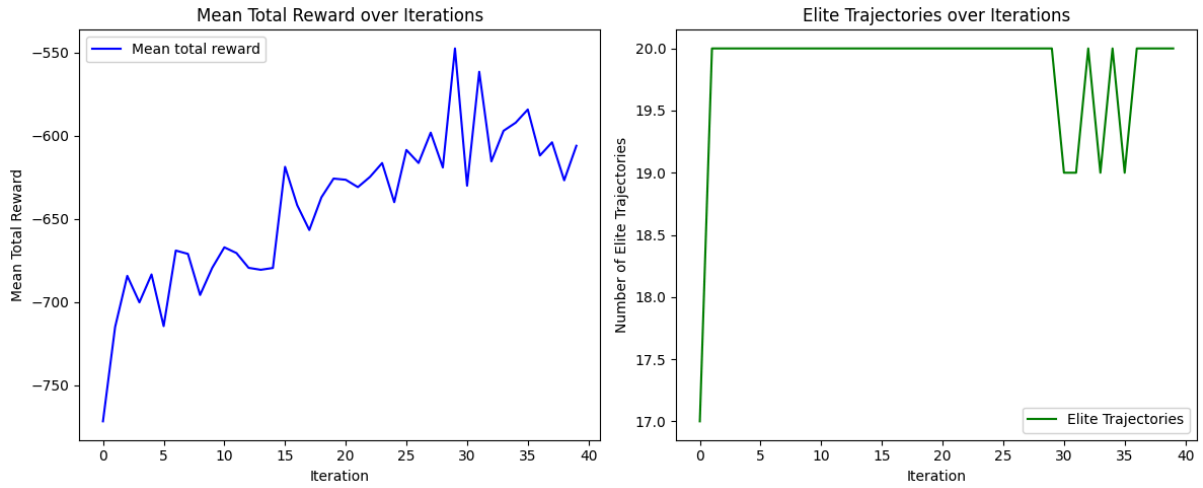


График застрял на одном значении - постоянно показывает число "3" - это может означать, что агент "застрял" в обучении и не улучшает свою стратегию.

Средняя награда увеличивается со временем: признак того, что агент обучается, однако она растёт нестабильно, поэтому необходимо менять гиперпараметры. Увеличим количество траекторий кратно, остальные параметры вернём в исходное значение.
Такс, это уже значительно лучше:



Судя по второму графику, уже на 8ой итерации агент уверенно находит нужные траектории, которых не больше 20 выходит. Теперь оставим такое кол-во траекторий и увеличим кол-во итераций, квантиль оставим 0.9



Похоже, что модель переобучилась. Квантиль сильно не меняла, не проверяла, оставила тот, который валидно работает. Также интересно проявляет себя количество траекторий элитных, не могу объяснить такое поведение графика, единственное - заметна взаимосвязь, похоже на резонанс, когда начинают резко меняться награды, тогда же и возникает скачкообразное изменение кол-ва элитных траекторий.

Таким образом, одними из лучших гиперпараметров является сочетание: квантиль 0.9, кол-во итераций 20/25, кол-во траекторий 200.

Задание 2.

Реализовать алгоритм Кросс-Энтропии с двумя типами сглаживания, указанными в лекции 1. При выбранных в пункте 1 гиперпараметров сравнить их результаты с результатами алгоритма без сглаживания.

https://docs.blender.org/manual/ru/3.3/modeling/meshes/editing/vertex/laplacian_smooth.html

Laplace smoothing — это метод, который помогает избежать присвоения нулевых вероятностей действиям, которые еще не были встречены. Он добавляет небольшую константу α\alphaα к числу выборов каждого действия, гарантируя, что все действия будут иметь ненулевую вероятность, даже если они не были выбраны в наблюдаемых данных.

Laplace Smoothing

Laplace smoothing is a smoothing technique that handles the problem of zero probability in Naïve Bayes. Using Laplace smoothing, we can represent P(w'|positive) as

$$P(w'|positive) = \frac{\text{number of reviews with w' and y = positive } + \ \alpha}{\text{N } + \ \alpha * K}$$

Here,
alpha represents the smoothing parameter,
K represents the number of dimensions (features) in the data, and
N represents the number of reviews with y=positive

If we choose a value of alpha!=0 (not equal to 0), the probability will no longer be zero even if a word is not present in the training dataset.

В коде реализовано так:

```python
def fit_laplace(self, elite_trajectories):
    new_model = np.zeros((self.state_n, self.action_n))
    for trajectory in elite_trajectories:
        for state, action in zip(trajectory['states'], trajectory['actions']):
            new_model[state][action] += 1

    # Laplace smoothing
    alpha = 0.1
    for state in range(self.state_n):
        new_model[state] = (new_model[state] + alpha) / (np.sum(new_model[state]) + alpha * self.action_n)

    self.model = new_model
```

Policy smoothing — это метод, при котором обновленная политика является взвешенной комбинацией новой оценочной политики:

$$\pi_{n+1}(a|s) = \lambda \cdot \pi_{n+1}(a|s) + (1 - \lambda) \cdot \pi_n(a|s)$$

Помогает сглаживать резкие изменения в политике, предотвращая большие колебания и обеспечивая более стабильное обучение агента.

- где $\lambda \in (0, 1]$

4.2 POLICY SMOOTHING

Building on these results, we develop *policy smoothing*, a simple model-agnostic randomized-smoothing based technique that can provide certified robustness without increasing the computational complexity of the agent's policy. Given a policy π, we define a smoothed policy $\bar{\pi}$ as:

$$\bar{\pi}\left(\cdot \mid o(s_t)\right) = \pi\left(\cdot \mid o(s_t) + \delta_t\right), \text{ where } \delta_t \sim \mathcal{N}(0, \sigma^2 I).$$

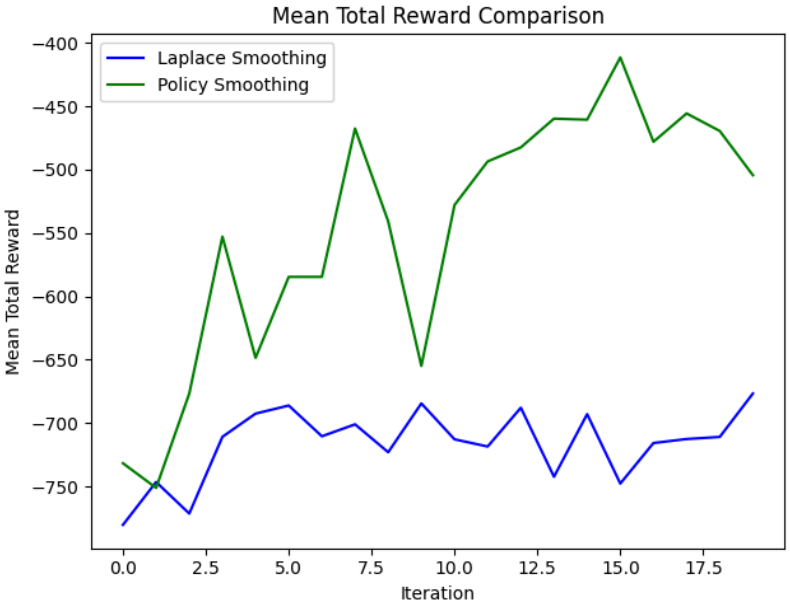
[POLICY SMOOTHING FOR PROVABLY ROBUST REINFORCEMENT LEARNING]

В коде реализуется таким методом:

```
def fit_policy_smoothing(self, elite_trajectories, lambda_):
    new_model = np.zeros((self.state_n, self.action_n))
    for trajectory in elite_trajectories:
        for state, action in zip(trajectory['states'], trajectory['actions']):
            new_model[state][action] += 1

    # Нормализация новой модели
    for state in range(self.state_n):
        if np.sum(new_model[state]) > 0:
            new_model[state] /= np.sum(new_model[state])
        else:
            new_model[state] = self.model[state].copy()

    # Применение сглаживания политики
    self.model = lambda_ * new_model + (1 - lambda_) * self.model
```

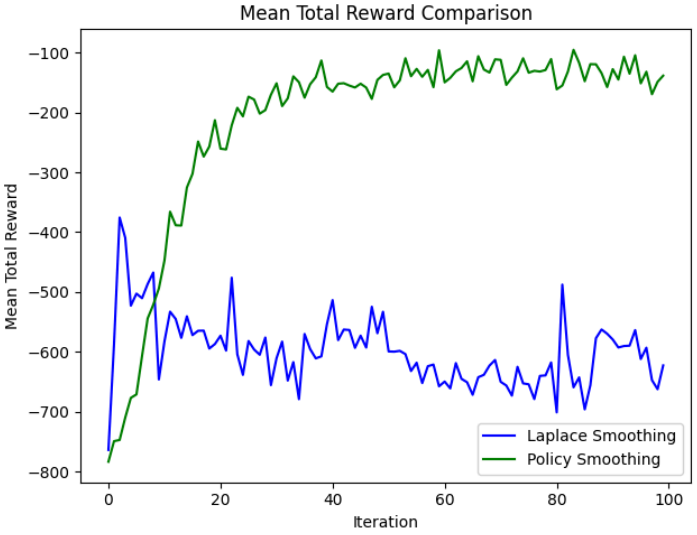
При 50 траекториях график сравнения двух сглаживаний вышел острым и ненаглядным, увеличила количество траекторий до 200
График представлен для $\lambda = 0.8$ (для policy) и $\alpha = 0.1$ (для smoothing):

Судя по графику, лучше агент обучается при policy smoothing. Уменьшу параметр $\lambda = 0.3$ и увеличу α до 0.6:

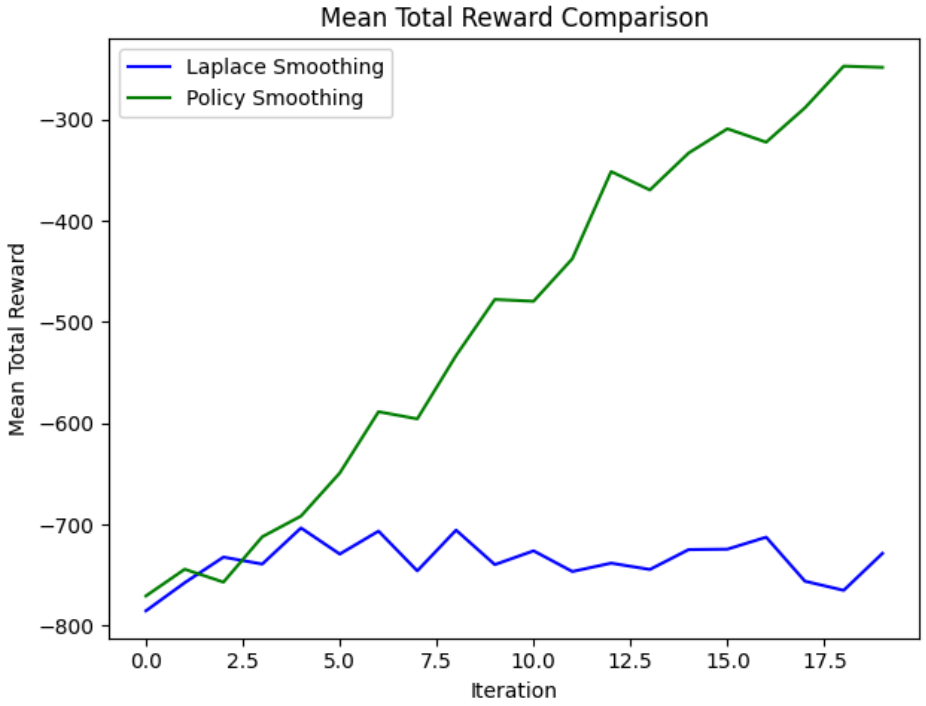


Увеличение коэффициента для лапласова сглаживания сказалось отрицательно на динамике обучения агента, поскольку эта добавка необходима для обработки неизведанных случаев и её лучше наоборот уменьшать, она может быть близка к нулю, главное - не равна ему.

Уменьшение коэффициента policy повлияло положительно. Судя по всему, в данной задаче наоборот не нужно сильно сохранять прошлое состояние, но и абстрагироваться также не имеет смысла. Попробуем ещё уменьшить оба коэффициента. Пусть $\alpha = 0.001$, а $\lambda = 0.1$. Для наглядности взято 100 итераций



Лапласово сглаживание не работает, зато обучение для policy smothing работает стабильно и агент обучается.
Если коэффициент для лапласа увеличить до 7, то также ничего более оптимального не получится:



Задание 3

Реализовать модификацию алгоритм Кросс-Энтропии для стохастических сред, указанную в лекции 1. Сравнить ее результат с алгоритмами из пунктов 1 и 2.

По логике, для стохастической среды политику сформировать как матрицу из распределённых равномерно величин уже нельзя. И изменить алгоритм кросс-энтропии необходимо как раз через модификацию определения политики:

Stochastic policy

$$\pi(a|s) \in [0,1], \quad a \in \mathcal{A}, \quad s \in \mathcal{S}$$

- Set π
- Agent starts from the initial state $S_0 \sim \mathcal{P}_0$
- acts $A_0 \sim \pi(\cdot|S_0)$
- gets the reward $R_0 = \mathcal{R}(S_0, A_0)$ and goes to the next state $S_1 \sim \mathcal{P}(\cdot|S_0, A_0)$
- acts $A_1 \sim \pi(\cdot|S_1)$
- gets the reward $R_1 = \mathcal{R}(S_1, A_1)$ and goes to the next state $S_2 \sim \mathcal{P}(\cdot|S_1, A_1)$
- $\ldots$
- $\tau = \{S_0, A_0, S_1, A_1, S_2, A_2, \ldots\}, \quad G(\tau) = \sum_{t=0}^{\infty} \gamma^t \mathcal{R}(S_t, A_t)$

The Reinforcement Learning problem

$$\mathbb{E}_\pi[G] \quad \longrightarrow \quad \max_\pi$$

Weakness: Problems with the stochastic environments

Solution:

By stochastic policy π_n, sample deterministic policies $\pi_{n,m}$, $m \in \overline{1, M}$. According to them, get trajectories $\tau_{m,k}$, $m \in \overline{1, M}$, $k \in \overline{1, K}$. Define

$$V_{\pi_{n,m}} = \frac{1}{K} \sum_{k=1}^{K} G(\tau_{m,k})$$

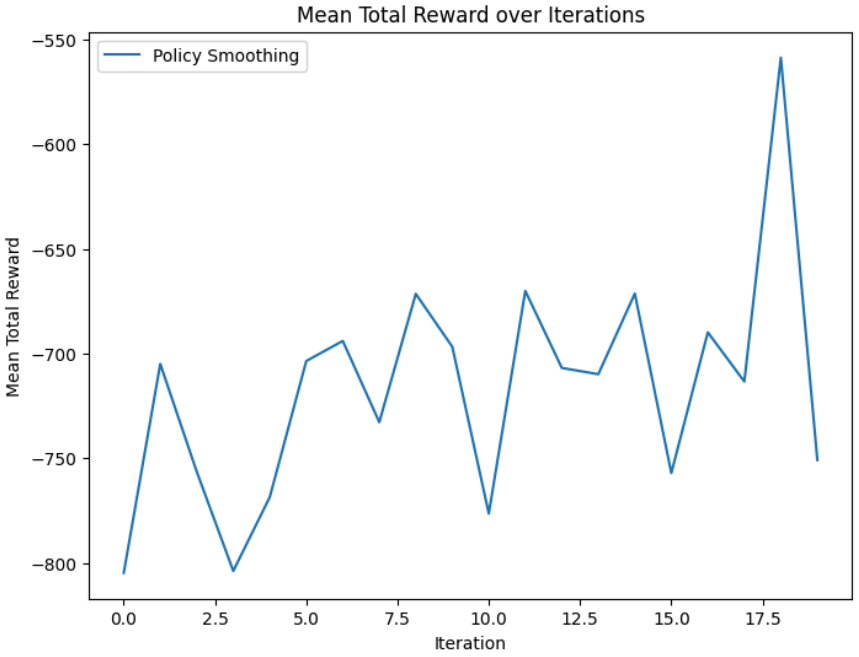
Select «elite» trajectories $\mathcal{T}_n = \{\tau_{m,k}, m \in \overline{1, M}, k \in \overline{1, K} : V_{\pi_{n,m}} > \gamma_q\}$ (γ_q — q-quantile of the numbers $V_{\pi_{n,m}}$, $m \in \overline{1, M}$).

Для решения проблемы стохастических сред, используя policy smoothing можно использовать подход, при котором стохастическая политика порождает детерминированные политики, каждая из которых генерирует множество траекторий. Затем можно выбрать элитные траектории на основе средней награды для каждой детерминированной политики.

Вот как изменится тогда код:

```python
def evaluate_deterministic_policy(env, agent, M, K):
    """
    Генерация детерминированных политик и их оценка.
    """
    deterministic_policies = []
    V_policies = []

    for m in range(M):
        policy = np.zeros((agent.state_n, agent.action_n))
        for state in range(agent.state_n):
            action = np.random.choice(np.arange(agent.action_n), p=agent.model[state])
            policy[state, action] = 1
        deterministic_policies.append(policy)

        total_rewards = []
        for _ in range(K):
            trajectory = get_trajectory(env, agent)
            total_rewards.append(np.sum(trajectory['rewards']))

        V_policies.append(np.mean(total_rewards))

    return deterministic_policies, V_policies
```

Взяла малое количество итераций и траекторий для ускорения проверки данных:



Обучение агента отследить четко не получается, робастности поведения/обучения агента не наблюдается.