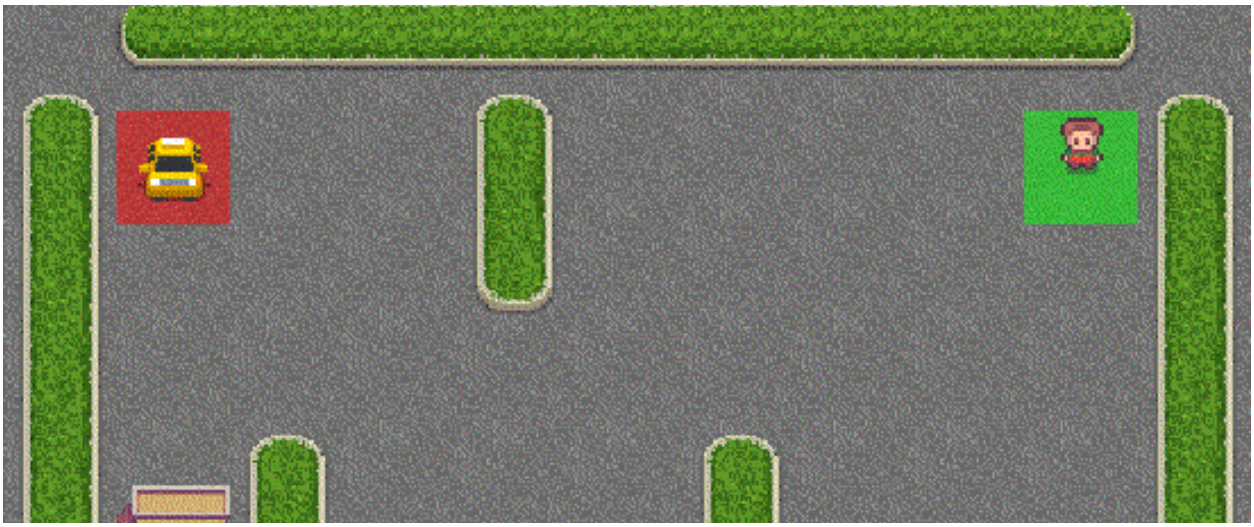


ДЗ1

Задание 1

Анализ задачи

Задача “Taxi-v3” содержит 6 действий и 500 состояний. Из этого можно вывести, что необходимо увеличить количество траекторий при оценке политики, чтобы протестировать агента на всех возможных начальных состояниях.

Пусть q = 0.9, тогда если всего возможно 500 начальных состояний (для простоты опустим, что достижимых начальных состояния всего 404, так как на решение задачи это практически не влияет), чтобы в элитные траектории попала хотя бы одна траектория для каждого начального состояния, минимальное количество траекторий должно составлять 5000.

Baseline

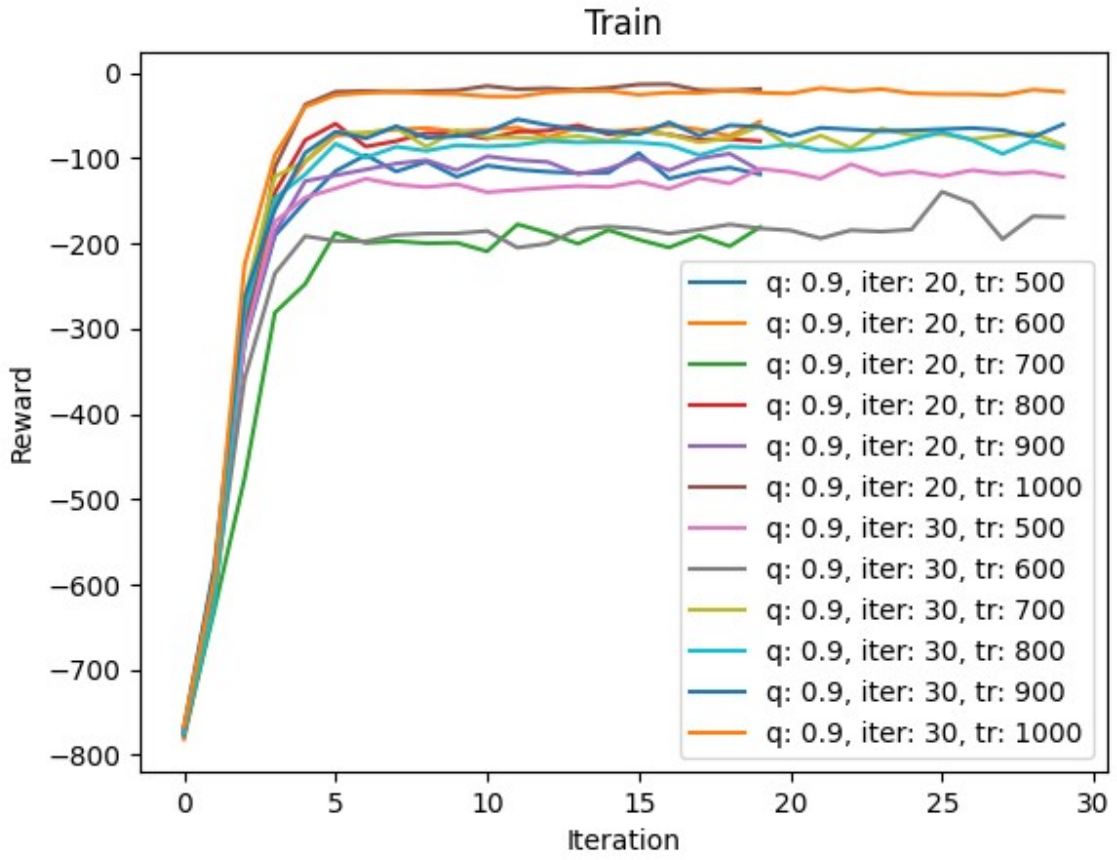
В качестве бейзлайна были использованы гиперпараметры с семинара:

q_param = 0.9

iterations_n =20

trajectory_n = 100

Для поиска оптимальных гиперпараметров я перебрал все возможные комбинации значений в определенном диапазоне, результаты экспериментов представлены на графике:



Для лучшей читаемости на график выведена только часть экспериментов.

Из графика можно сделать следующие выводы:

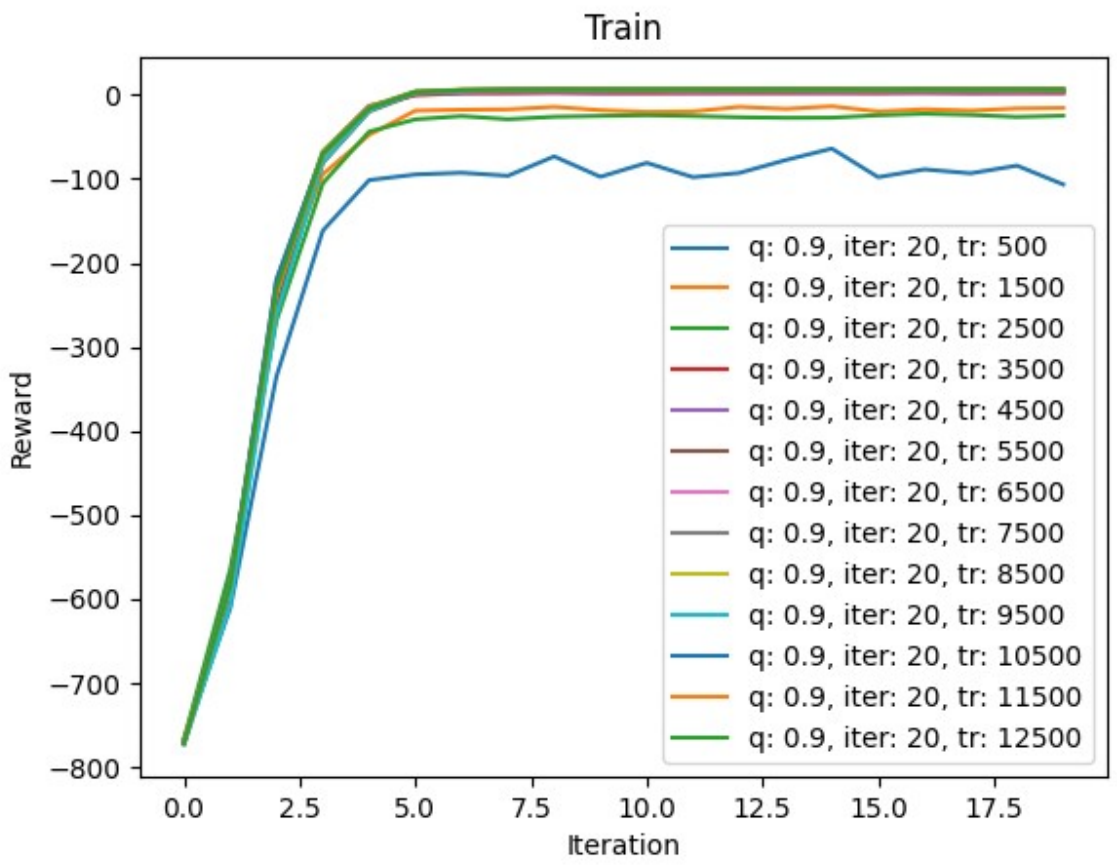
- агент достигает верхнего порога качество приблизительно за 5-8 итераций
- после достижения потолка качество меняется не стабильно при малом количестве изученных траекторий, и стабилизируется при увеличении их количества

Из второго утверждения можно сделать вывод, что агент выбирает действие случайно. Можно предположить, что так происходит из-за того, что при обучении он не изучил все возможные варианты траекторий. Данное предположение может

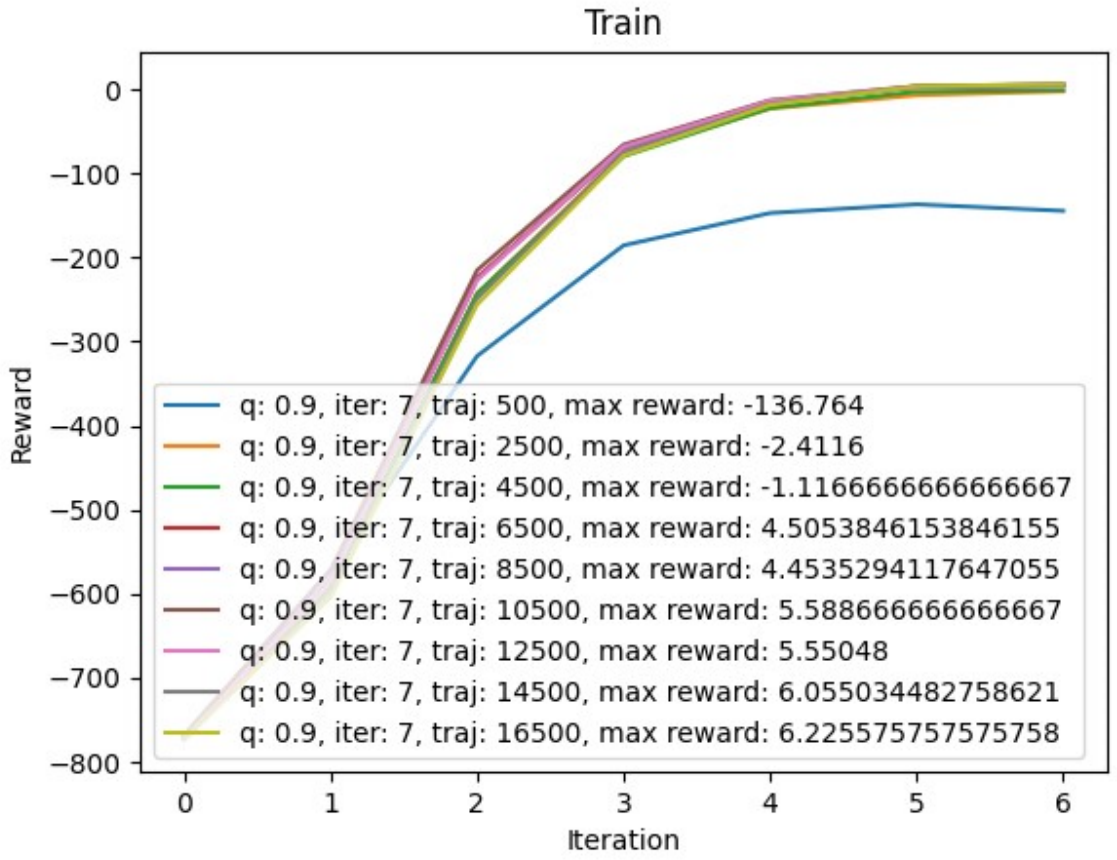
быть правдивым потому, что при q = 0.9 и trajectory_n = 500 в элитных траекториях окажутся траектории, использующие только приблизительно 10% начальных состояний.

Подбор гиперпараметров

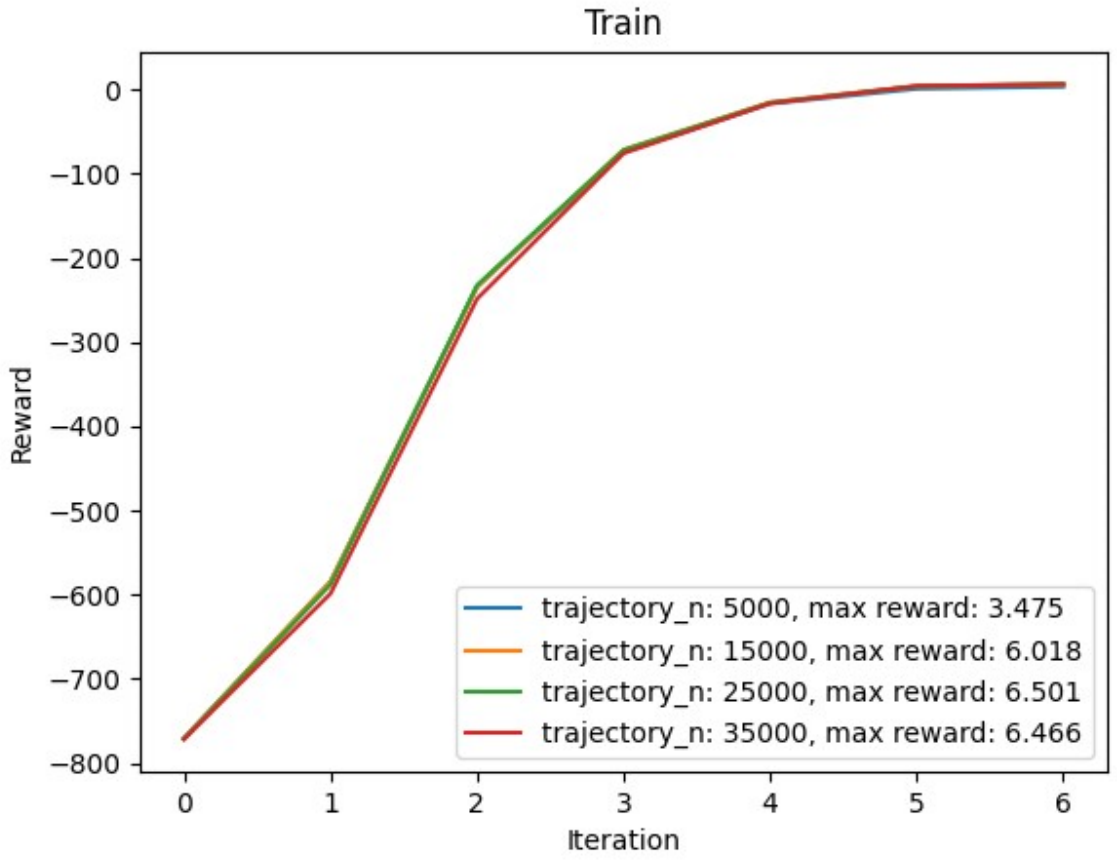
Попробуем увеличить количество анализируемых траекторий. Результаты эксперимента представлены на графике:



Видно, что при trajectory_n > 2500 обучение стабилизируется и качество улучшается. Примем iteration_n = 7 и обучим агента заново, отобразив максимальную награду:



Видно, что 7 итераций достаточно для стабилизации обучения и достижения определенного порога качества при заданных гиперпараметрах. Попробуем еще сильнее увеличить количество траекторий:



Видно, что графики практически совпадают, значит мы достигли определенного максимума стабильности и качества. Примем итоговые гиперпараметры следующими:

q_param = 0.9

iterations_n = 7

trajectory_n = 15000

Если вспомнить, что поле игры имеет размер 5x5, за каждый шаг до финиша начисляется награда -1, а за доставку пассажира +20, то оптимальная средняя награда будет около 8.5, так как минимальная награда будет равна 2 (когда такси находится в локации Y, пассажир в локации G, а точка назначения в локации R), а максимальная 15 (такси и пассажир находятся в Y, а пункт назначения в R). Улучшение качества до максимального может быть достигнуто с помощью сглаживания, что будет рассмотрено в следующем задании.

Задание 2

Сглаживание по Лапласу

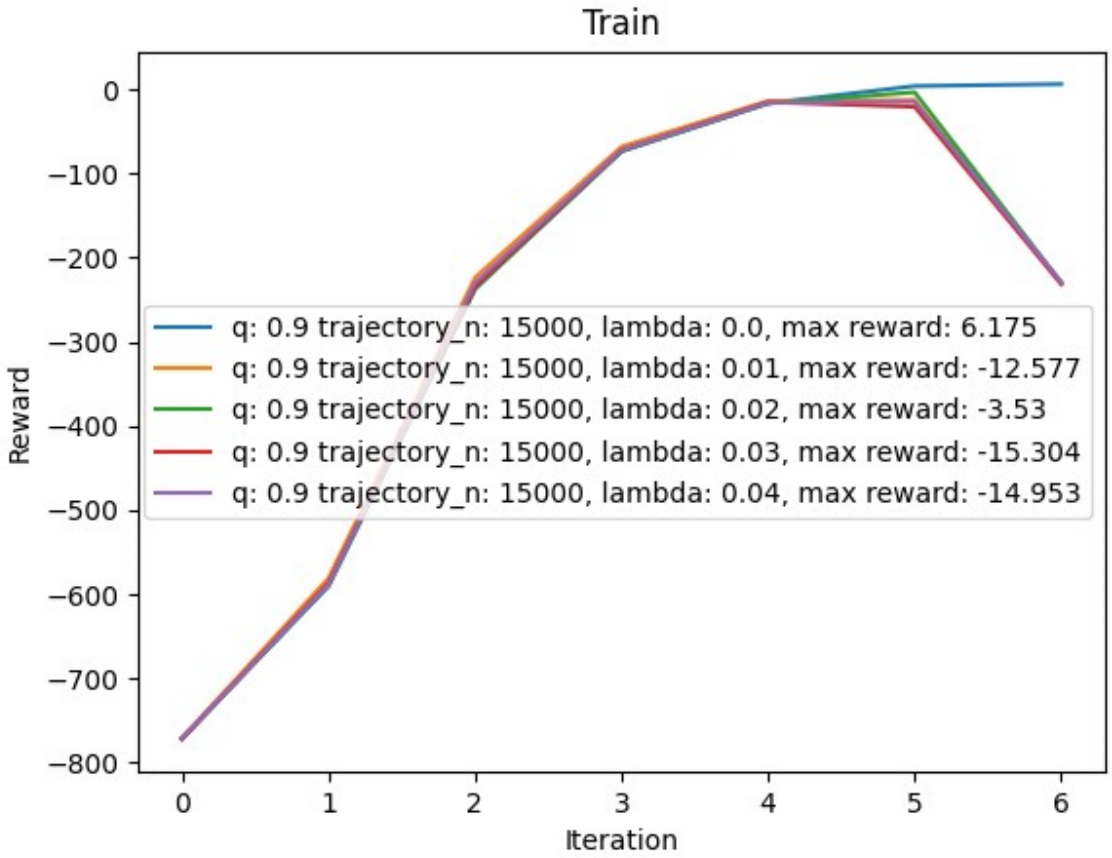
Вспомним формулу сглаживания по Лапласу:

$$\pi_{n+1}(a|s) = \frac{|(a|s) \in \mathcal{T}_n| + \lambda}{|s \in \mathcal{T}_n| + \lambda|\mathcal{A}|}, \quad \lambda > 0$$

В коде оно реализовано следующим образом в методе train класса Trainer:

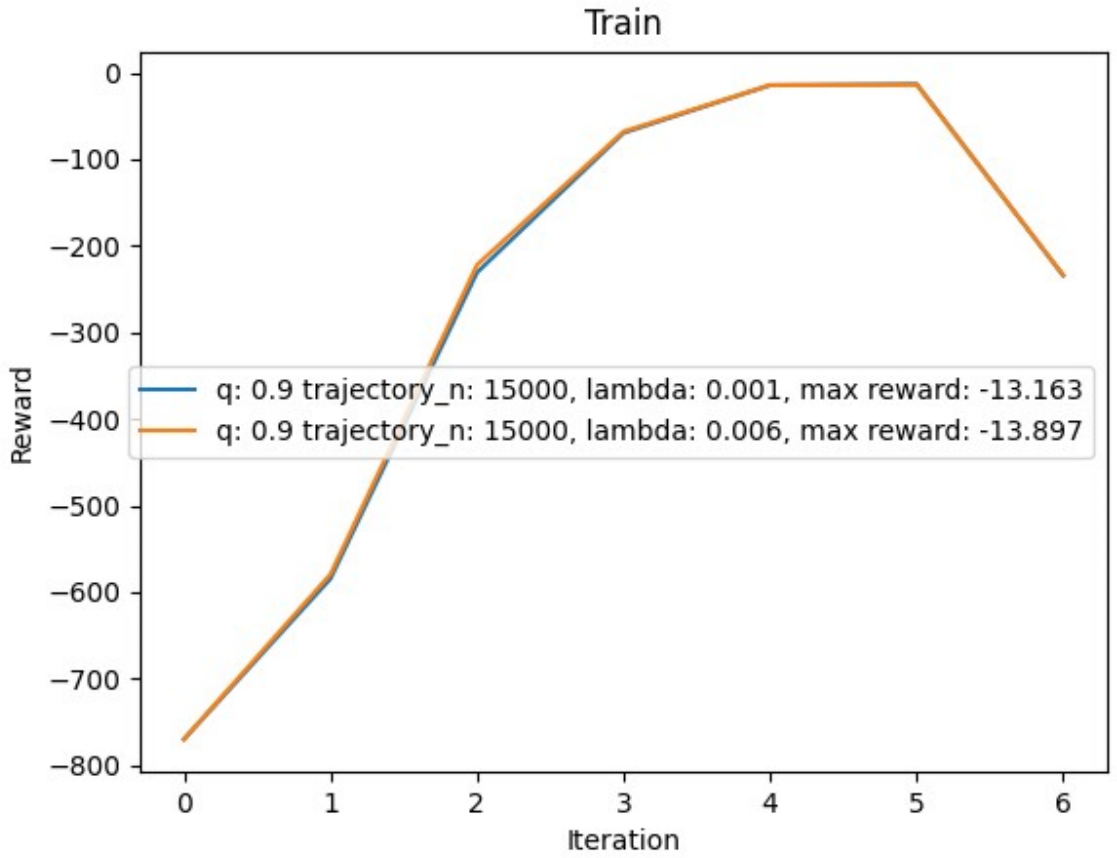
```
elif smoothing == "laplace":
    for state in range(self.state_n):
        if np.sum(new_model[state]) > 0:
            s = np.sum(new_model[state])
            new_model[state] += lmbd
            new_model[state] /= (s + self.action_n * lmbd)
        else:
            new_model[state] += lmbd
            new_model[state] /= (self.action_n * lmbd)
    agent.model = new_model
```

Сравним сходимость алгоритма со сглаживанием и без (при lambda = 0), используя гиперпараметры, подобранные в задаче 1:



Видно, что со сглаживанием алгоритм работает менее стабильно. Можно предположить, что причина в “размазывании” распределения вероятностей действий, из за чего агент начинает совершать действия в околослучайном порядке.

Попробуем уменьшить параметр сглаживания:



Уменьшение параметра не вызывает стабилизации обучения. Попробуем улучшить качество с помощью сглаживания по политике.

Сглаживание по политике

$$\pi_{n+1}(a|s) \leftarrow \lambda\pi_{n+1}(a|s) + (1-\lambda)\pi_n(a|s), \quad \lambda \in (0,1]$$

Формула сглаживания по политике

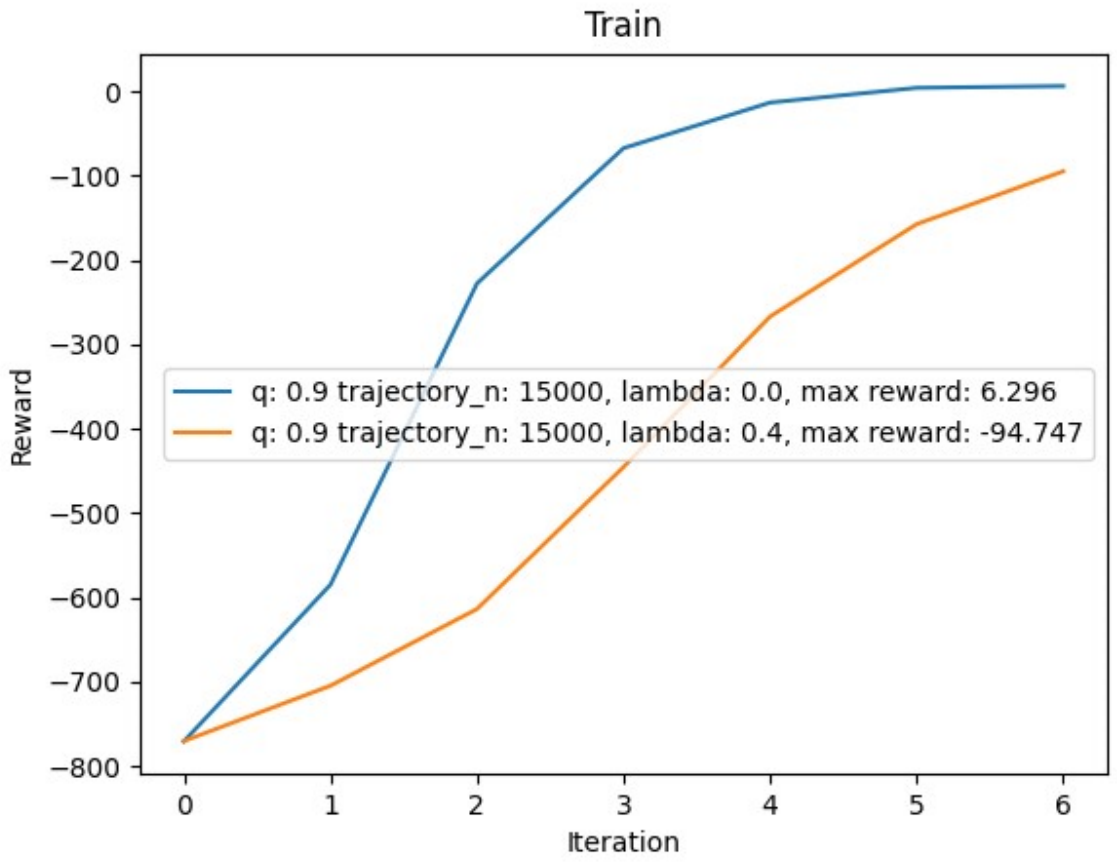
Реализация в коде:

```
elif smoothing == "policy":
    for state in range(self.state_n):
        if np.sum(new_model[state]) > 0:
            new_model[state] /= (np.sum(new_model[state]))
```

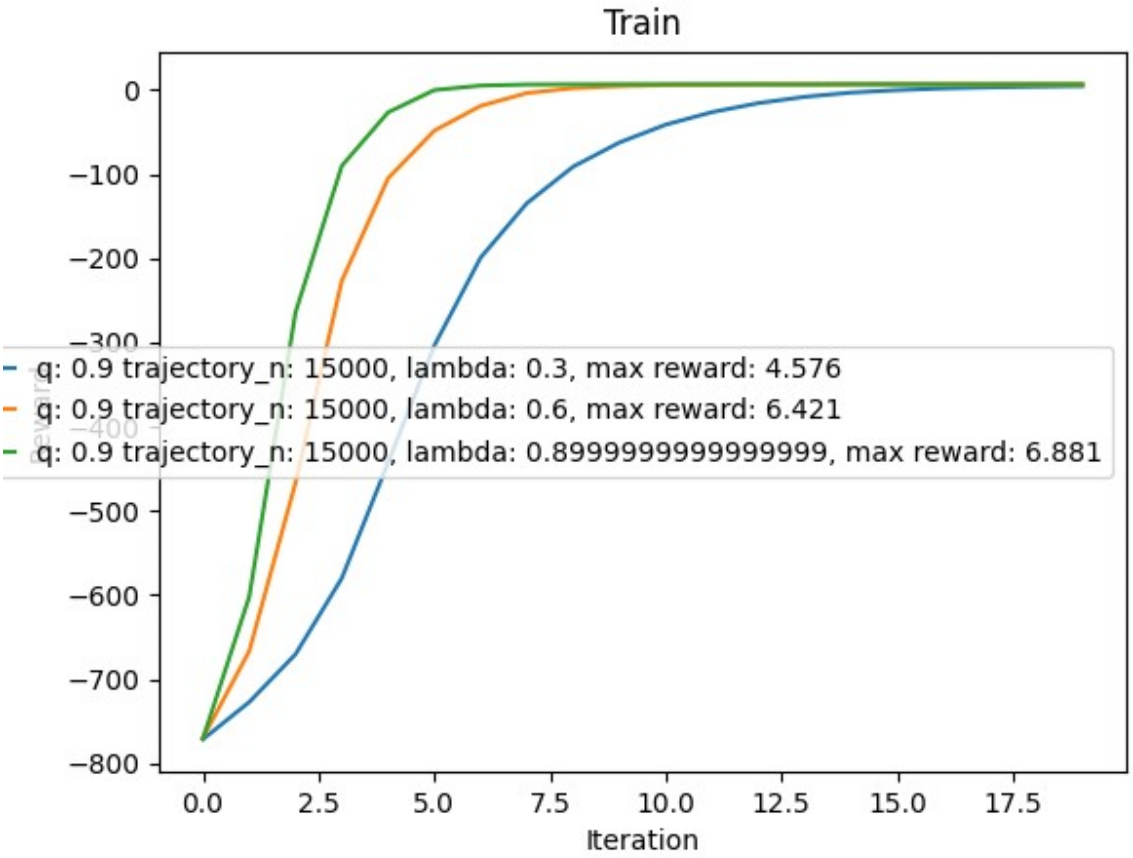
```
        else:
            new_model[state] = agent.model[state].copy()

    agent.model = new_model * lmbd + agent.model * (1 - lmbd)
```

Сравним сходимость алгоритма со сглаживанием по политике и без сглаживания:



Видно, что сходимость алгоритма со сглаживанием ощутимо замедляется, но при этом идет стабильно. Увеличим количество итераций обучения и попробуем разные значения разных параметров лямбда:



Видно, что сходимость ускоряется с увеличением лямбда. Это связано с тем, что параметр лямбда отвечает за “консервативность алгоритма”, т.е. чем меньше лямбда, тем меньше изменений в политику вносит каждая итерация.

Задание 3

Задача состояла в реализации следующего подхода к работе со стохастическими средами:

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 $V_{\pi_{n,m}}$, $m \in \overline{1,M}$).

Реализация в коде:

1. Сэмплирование детерминированных политик

```
determ_policy = np.zeros((self.state_n, self.action_n))
for state in range(self.state_n):
    action = np.random.choice(np.arange(self.action_n), p=stochastic_policy[state])
    determ_policy[state][action] = 1
```

2. Подсчет V

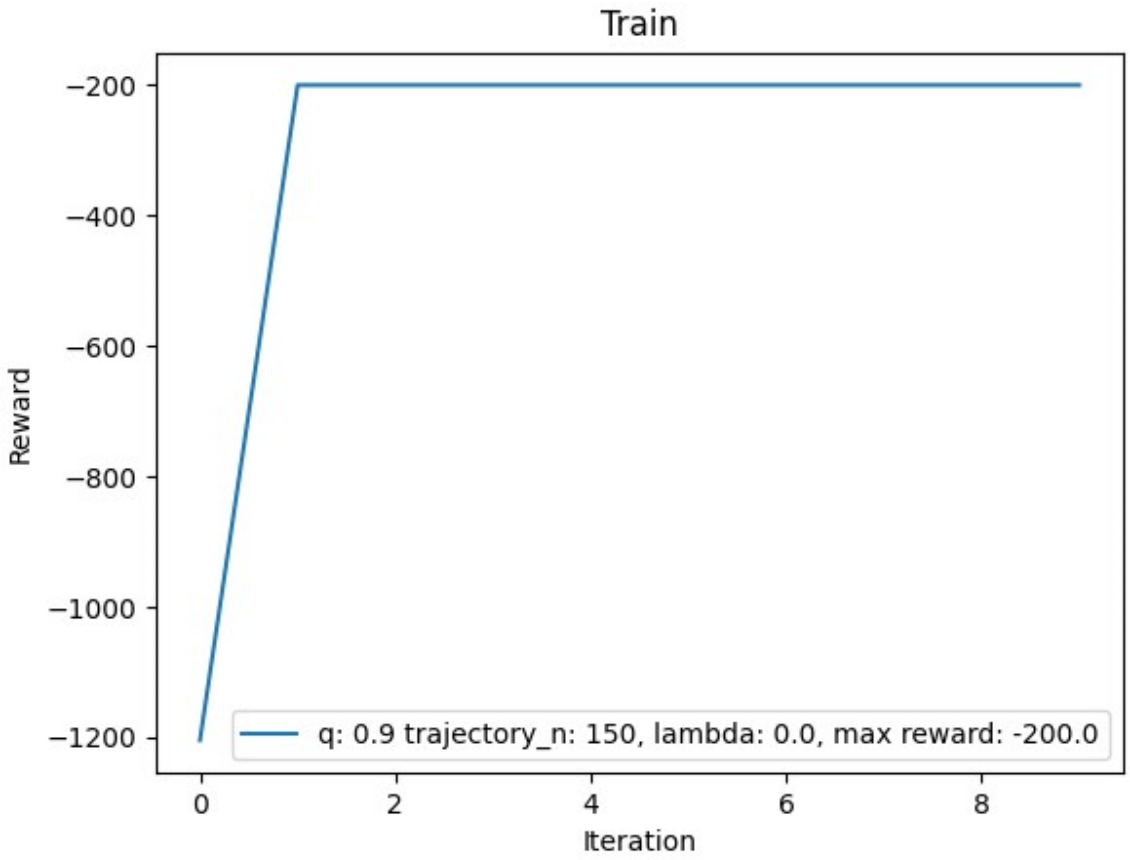
```
agent.model = determ_policy
trs = [agent.get_trajectory(self.env) for _ in range(trajectory_n)]
trajectories = trajectories + trs
local_reward = np.mean([np.sum(trajectory['rewards']) for trajectory in trs])
total_rewards += [local_reward for _ in trs]
```

Код целиком:

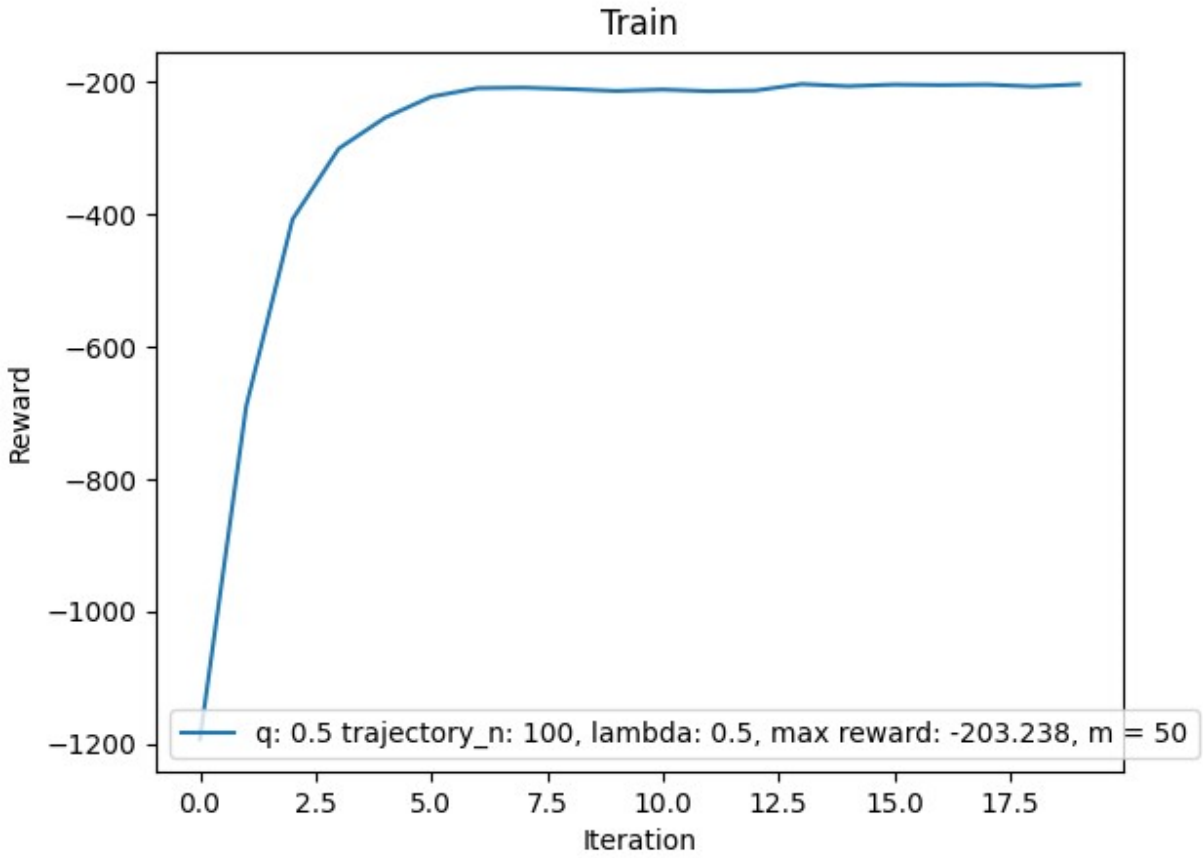
```
total_rewards = []
if self.stochastic_env:
    stochastic_policy = agent.model.copy()
    trajectories = []
    for i in tqdm(range(self.m), desc="Iteration {0}".format(iteration)):
        determ_policy = np.zeros((self.state_n, self.action_n))
        for state in range(self.state_n):
            action = np.random.choice(np.arange(self.action_n), p=stochastic_policy[state])
            determ_policy[state][action] = 1
        agent.model = determ_policy
        trs = [agent.get_trajectory(self.env) for _ in range(trajectory_n)]
```

```
        trajectories = trajectories + trs
        local_reward = np.mean([np.sum(trajectory['rewards']) for trajectory in trs])
        total_rewards += [local_reward for _ in trs]
    rewards.append(np.mean(total_rewards))
    agent.model = stochastic_policy
```

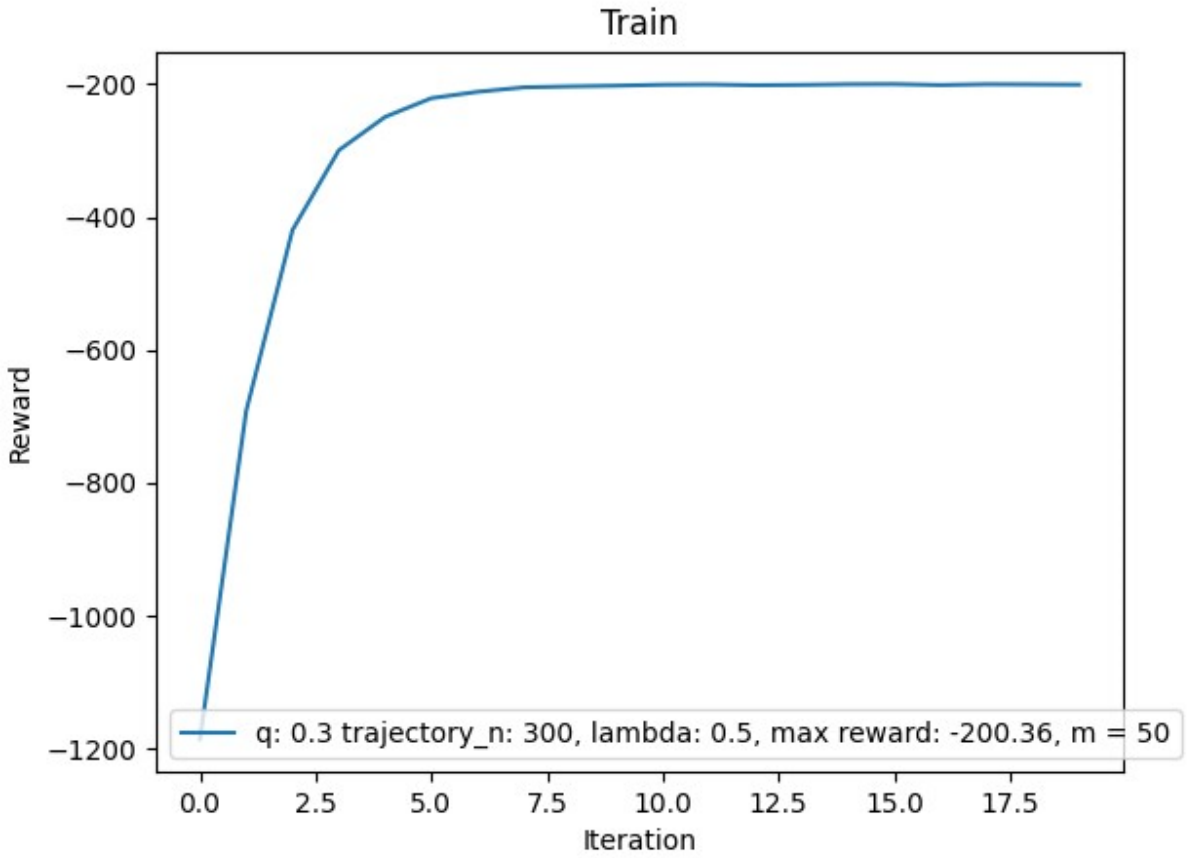
Без сглаживания алгоритм сходится с ошибке -200:



Применим сглаживание:



Увеличим количество траекторий на каждую детерминированную политику и уменьшим q:



Таким образом, при добавлении сглаживания алгоритм все еще сходится к -200, однако при увеличении числа траекторий сходимость происходит более плавно.