Taxi-v3 из Gym

Дано:

Всего имеется 500 дискретных состояний, поскольку имеется 25 позиций такси, 5 возможных местоположений пассажира (включая случай, когда пассажир находится в такси) и 4 места назначения.

Задание # 1 алгоритмом Кросс-Энтропии

Для воспроизведения зафиксируем seed = 42 и напишем класс для решения данной задачи.

Реализация – универсального метода:
При:
laplace_smoothing: float = 0.0,
policy_smoothing: float = 1.0,
не происходит сглаживаний -> базовый алгоритм

```
1 usage
def fit(   You, Yesterday · Uncommitted changes
    self,
    elite_trajectories,
    laplace_smoothing: float = 0.0,
    policy_smoothing: float = 1.0,
) -> None:
    new_model = np.ones((self.state_n, self.action_n)) * laplace_smoothing
    for trajectory in elite_trajectories:
        for state, action in zip(trajectory["states"], trajectory["actions"]):
            new_model[state][action] += 1

    for state in range(self.state_n):
        if np.sum(new_model[state]) > 0:
            new_policy = new_model[state] / np.sum(new_model[state])
            old_policy = self.model[state].copy()
            new_model[state] = (
                policy_smoothing * new_policy + (1 - policy_smoothing) * old_policy
            )
        else:
            new_model[state] = self.model[state].copy()

    self.model = new_model
    return None
```

Рассмотрим сводную таблицу результатов экспериментов, обратите внимание, что значения взяты на последней итерации mean_total_reward, max_total_reward:

	q_param	trajectory_n	iteration_n	max_length	mean_total_reward	max_total_reward
0	0.9	200	40	500	-461.8500	14
1	0.9	200	40	1000	-306.1400	14
2	0.9	400	40	500	-203.3300	15
3	0.9	400	40	1000	-163.8275	15
4	0.6	200	40	500	-132.7000	15
5	0.6	200	40	1000	-129.9400	15
6	0.6	400	40	500	2.0875	15
7	0.6	400	40	1000	-2.1450	15
8	0.4	200	40	500	-42.6450	13
9	0.4	200	40	1000	-122.6150	9
10	0.4	400	40	500	-12.5075	15
11	0.4	400	40	1000	6.6275	15

Обратим внимание, что сходимость достигнуто еще на 20 итерациях

```
{'iteration:': 11, 'mean_total_reward': -85.67, 'max_total_reward': 8}
{'iteration:': 12, 'mean_total_reward': -58.5775, 'max_total_reward': 10}
{'iteration:': 13, 'mean_total_reward': -36.505, 'max_total_reward': 11}
{'iteration:': 14, 'mean_total_reward': -21.175, 'max_total_reward': 14}
{'iteration:': 15, 'mean_total_reward': -9.6325, 'max_total_reward': 11}
{'iteration:': 16, 'mean_total_reward': -2.4175, 'max_total_reward': 13}
{'iteration:': 17, 'mean_total_reward': 1.145, 'max_total_reward': 15}
{'iteration:': 18, 'mean_total_reward': 3.29, 'max_total_reward': 14}
{'iteration:': 19, 'mean_total_reward': 4.23, 'max_total_reward': 14}
{'iteration:': 20, 'mean_total_reward': 5.0275, 'max_total_reward': 15}
{'iteration:': 21, 'mean_total_reward': 5.93, 'max_total_reward': 15}
{'iteration:': 22, 'mean_total_reward': 6.095, 'max_total_reward': 14}
{'iteration:': 23, 'mean_total_reward': 5.8425, 'max_total_reward': 15}
{'iteration:': 24, 'mean_total_reward': 6.435, 'max_total_reward': 15}
{'iteration:': 25, 'mean_total_reward': 6.52, 'max_total_reward': 15}
{'iteration:': 26, 'mean_total_reward': 6.79, 'max_total_reward': 15}
{'iteration:': 27, 'mean_total_reward': 6.53, 'max_total_reward': 14}
{'iteration:': 28, 'mean_total_reward': 6.7225, 'max_total_reward': 15}
```

В отличие от примера рассмотренного на практике для сходимости алгоритма нужно выбирать q – значение не слишком большим, так как множество траекторий не достигли хороших результатов из – за **СТОХАСТИЧЕСКОЙ СРЕДЫ**, поэтому в данном случае убрать неудачные траектории.

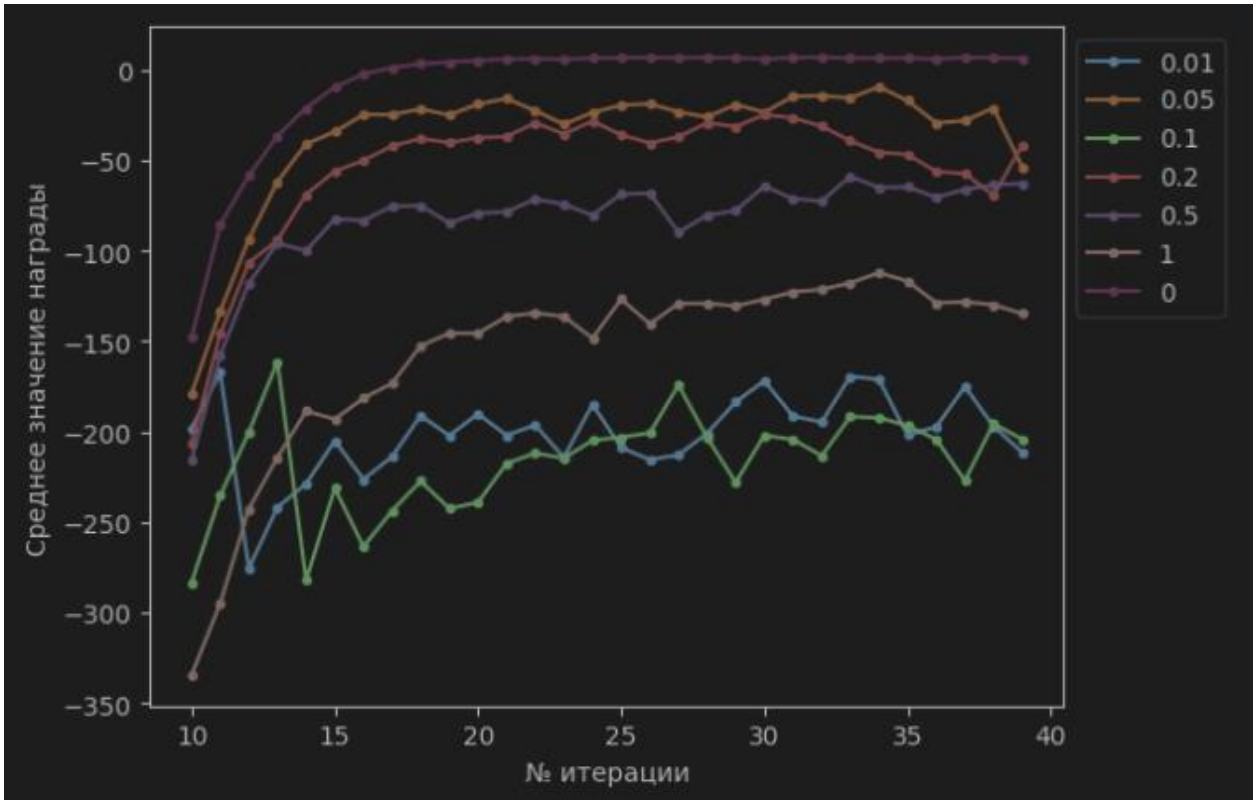
Задание # 2.1 Сглаживание Лапласиана

Смысл – добавление вероятностей для каждого действия -> Уход от жестких траекторий.
Реализация в 54 строчке:

```
54        new_model = np.ones((self.state_n, self.action_n)) * laplace_smoothing
55        for trajectory in elite_trajectories:
56            for state, action in zip(trajectory["states"], trajectory["actions"]):
57                new_model[state][action] += 1    You, Yesterday • Uncommitted changes
```

Результаты после обучения алгоритмов на лучших параметрах изначального решения:

laplace_smoothing	mean_total_reward	max_total_reward
0.00	6.6275	15
0.01	-211.4775	14
0.05	-53.7075	15
0.10	-204.1525	14
0.20	-41.9375	14
0.50	-62.7975	13
1.00	-134.8225	8



Вывод:

Решение также сошлось при lambda = 0.05, однако из-за сглаживания метрика среднего значения награды меньше ввиду наличия вероятности совершить неоптимиальные действия из-за сглаживания. Отсюда сходимость медленнее, но модель не стабильнее – на

графиках видны скачки метрики. Также стоит отметить слишком большое lambda не даст модели сойтись и метрики базовой модели не были пробиты.

Задание # 2.2 Сглаживание Политик

Смысл – частично оставляем вероятности переходов от политики прошлой итерации для каждого действия -> Уход от жестких траекторий.
Реализация 62 – 65 строчки:

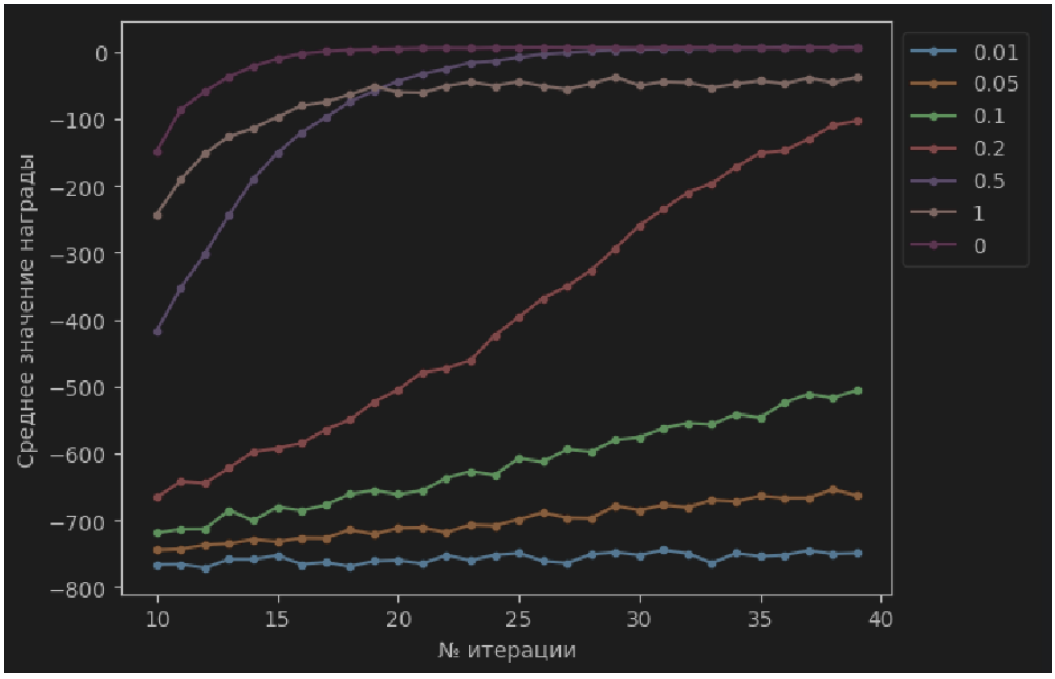
```
        for state in range(self.state_n):
            if np.sum(new_model[state]) > 0:
                new_policy = new_model[state] / np.sum(new_model[state])
                old_policy = self.model[state].copy()
                new_model[state] = (    You, Yesterday • Uncommitted changes
                    policy_smoothing * new_policy + (1 - policy_smoothing) * old_policy
                )
            else:
                new_model[state] = self.model[state].copy()

        self.model = new_model
```

Результаты экспериментов в таблице:

	policy_smoothing	mean_total_reward	max_total_reward
0	0.00	6.6275	15
1	0.01	-749.1025	-155
2	0.05	-662.7250	-58
3	0.10	-505.5125	-53
4	0.20	-102.9050	12
5	0.50	6.9575	15
6	0.90	-37.7625	15

График



Вывод:

Из таблицы видно, что модель policy_smoothing = 0.5 имеет лучше метрики чем базовая модель!!! На графике видно, что модель обучается дольше. Однако четко виден монотонный рост метрики с каждой итерацией. И видим, что модель сходится.

Задание # 3 модификацию алгоритм Кросс-Энтропии для стохастических сред

Реализация:

```
for iteration in range(iteration_n):
    policies = {}
    # policy evaluation
    for i in range(num_policy):
        trajectories = [
            self.get_trajectory(env=env, max_len=max_length)
            for _ in range(trajectory_n)
        ]

        total_rewards = [
            np.sum(trajectory["rewards"]) for trajectory in trajectories
        ]
        policies[i] = {"trajectories": trajectories, "total_rewards": total_rewards}
    reward_for_metric = flatten_list([p["total_rewards"] for p in policies.values()])
    all_policies_rewards = [np.sum(p["total_rewards"]) for p in policies.values()]
    info = {
        "iteration:": iteration,
        "mean_total_reward": np.mean(reward_for_metric),
        "max_total_reward": np.max(reward_for_metric),
    }
    if debug:
        print(info)
    mean_iteration_rewards.append(info)

    # policy improvement
    quantile = np.quantile(all_policies_rewards, q_param)
    elite_trajectories = []
    for policy, r in zip(policies.values(), all_policies_rewards):
        if r > quantile:   You, Moments ago • Uncommitted changes
            elite_trajectories.extend(policy["trajectories"])
```

Логи обучения:

{'iteration:': 0, 'mean_total_reward': -772.16525, 'max_total_reward': -18}

{'iteration:': 1, 'mean_total_reward': -768.13025, 'max_total_reward': -70}

{'iteration:': 2, 'mean_total_reward': -765.96925, 'max_total_reward': -32}

{'iteration:': 3, 'mean_total_reward': -764.48825, 'max_total_reward': -9}

{'iteration:': 4, 'mean_total_reward': -763.0635, 'max_total_reward': -42}

{'iteration:': 5, 'mean_total_reward': -763.8945, 'max_total_reward': -59}

{'iteration:': 6, 'mean_total_reward': -757.075, 'max_total_reward': 2}

{'iteration:': 7, 'mean_total_reward': -751.12225, 'max_total_reward': -36}

{'iteration:': 8, 'mean_total_reward': -747.5925, 'max_total_reward': -27}
{'iteration:': 9, 'mean_total_reward': -745.01925, 'max_total_reward': -21}
{'iteration:': 10, 'mean_total_reward': -742.473, 'max_total_reward': -17}
{'iteration:': 11, 'mean_total_reward': -737.62325, 'max_total_reward': -38}
{'iteration:': 12, 'mean_total_reward': -735.2585, 'max_total_reward': -36}
{'iteration:': 13, 'mean_total_reward': -732.1625, 'max_total_reward': -14}
{'iteration:': 14, 'mean_total_reward': -721.969, 'max_total_reward': -33}
{'iteration:': 15, 'mean_total_reward': -711.97375, 'max_total_reward': 0}
{'iteration:': 16, 'mean_total_reward': -708.2305, 'max_total_reward': -32}
{'iteration:': 17, 'mean_total_reward': -701.6465, 'max_total_reward': -5}
{'iteration:': 18, 'mean_total_reward': -700.235, 'max_total_reward': -14}
{'iteration:': 19, 'mean_total_reward': -693.801, 'max_total_reward': 3}
{'iteration:': 20, 'mean_total_reward': -692.29175, 'max_total_reward': -9}
{'iteration:': 21, 'mean_total_reward': -687.1895, 'max_total_reward': 0}
{'iteration:': 22, 'mean_total_reward': -681.69625, 'max_total_reward': 10}
{'iteration:': 23, 'mean_total_reward': -679.30075, 'max_total_reward': 0}
{'iteration:': 24, 'mean_total_reward': -681.06675, 'max_total_reward': 1}
{'iteration:': 25, 'mean_total_reward': -676.71175, 'max_total_reward': -10}
{'iteration:': 26, 'mean_total_reward': -668.4255, 'max_total_reward': 3}
{'iteration:': 27, 'mean_total_reward': -659.7625, 'max_total_reward': 2}
{'iteration:': 28, 'mean_total_reward': -656.38525, 'max_total_reward': -9}
{'iteration:': 29, 'mean_total_reward': -650.4265, 'max_total_reward': -2}
{'iteration:': 30, 'mean_total_reward': -649.119, 'max_total_reward': -9}
{'iteration:': 31, 'mean_total_reward': -642.82425, 'max_total_reward': 11}
{'iteration:': 32, 'mean_total_reward': -626.805, 'max_total_reward': 0}
{'iteration:': 33, 'mean_total_reward': -631.45425, 'max_total_reward': -18}
{'iteration:': 34, 'mean_total_reward': -625.54975, 'max_total_reward': 5}
{'iteration:': 35, 'mean_total_reward': -613.763, 'max_total_reward': 10}
{'iteration:': 36, 'mean_total_reward': -606.99125, 'max_total_reward': -2}
{'iteration:': 37, 'mean_total_reward': -608.369, 'max_total_reward': 7}
{'iteration:': 38, 'mean_total_reward': -605.03675, 'max_total_reward': -3}
{'iteration:': 39, 'mean_total_reward': -596.1735, 'max_total_reward': 10}
Средняя награда на последней итерации: -596.1735
Максимальная награда на последней итерации: 10

Вывод более стабильный метод требуемый большего кол-ва итераций для сходимости, и соотвественно наиболее вычислительно сложный.

Общий вывод:

Рассмотрев 3 разных способа улучшения базовой модели, можно смело утвеждать, что policy_smoothing самый просто и действенный из всех. В этом методе получились лучшие метрики и это единственный метод, который смог пробить метрики базовой модели.