Аверченко Марк, КН402, ДЗ 1 по спецкурсу RL

Отчет по домашнему заданию №1

Задание 1

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

Решение

Решение задачи находится в файле task1.py, за основу взят код из лекции. Гиперпараметры алгоритма:

- state_n - кол-во состояний в окружении
- action_n - кол-во действий доступных для агента
- episode_n - кол-во эпох обучения
- trajectory_n - кол-во траекторий
- trajectory_len - длина траектории, то есть кол-во действий, которое делает алгоритм в траектории
- q_param - отсечка по квантилю

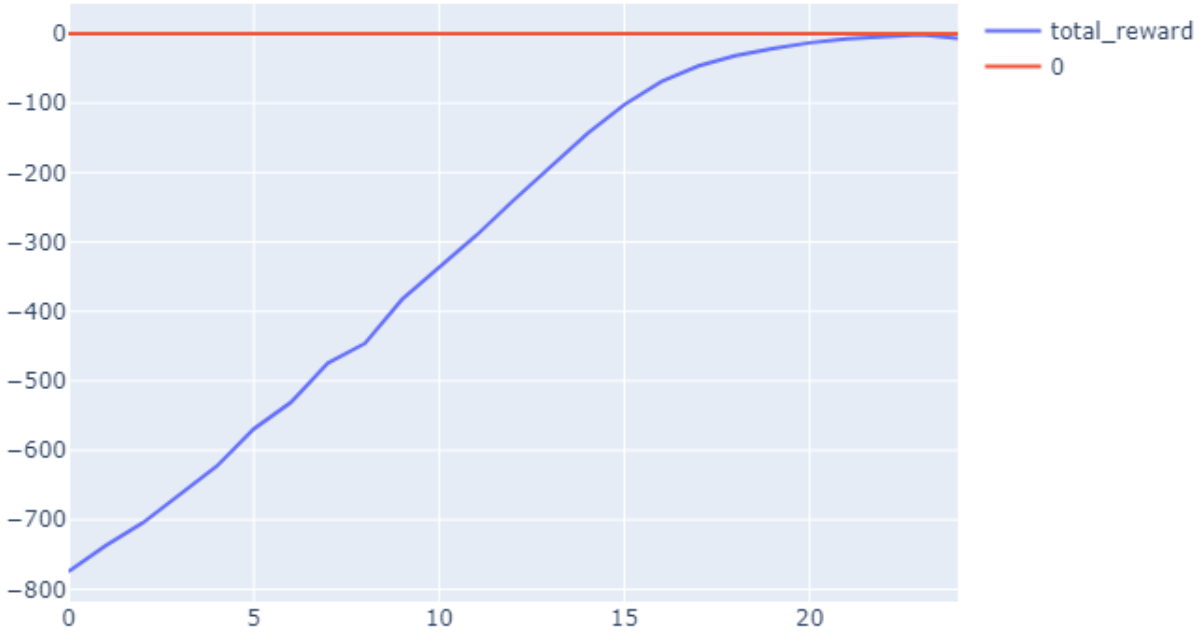
Менять параметры state_n, action_n не имеет смысла, так как они нужны для того, чтобы алгоритм корректно работал на текущем окружении, все остальные параметры имеет смысл менять. Для перебора параметров я использовал RandomSearch со следующими значениями для гиперпараметров и провел 30 обучений CEM алгоритма

```
"q_param": [0.3, 0.4, 0.5, 0.6, 0.7],
"episode_n": [12, 25, 50],
"trajectory_n": [250, 500],
"trajectory_len": [500, 1000, 2000, 10 ** 4]
```

Самыми лучшими гиперпараметрами оказались гиперпараметры ниже, с ними получилось набрать вознаграждение равное -4.526

q_param	**0.3**
episode_n	25
trajectory_n	500
trajectory_len	2000

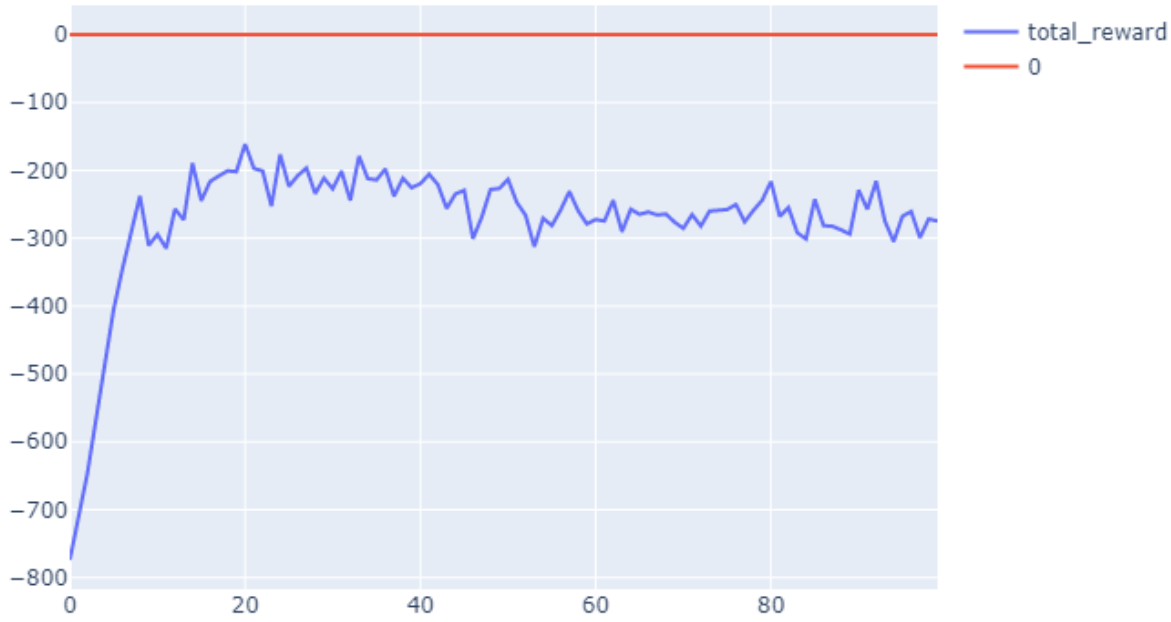
График обучения CEM с данными гиперпараметрами



Также были замечены следующие особенности:

- Лучше q_param делать не высоким, это связано с тем, что у нас получится больше элитных траекторий и следовательно будет меньше ошибка, если какая-то элитная траектория имеет случайно высокое вознаграждение

- Лучше episode_n делать не более 25, так как алгоритм CEM склонен к переобучению, как, например, на этом графике



- Лучше trajectory_n, trajectory_len ставить как можно больше, так как trajectory_n позволяет снизить ошибку в среднем вознаграждении, а trajectory_len расширяет вектор действий алгоритма

Задание 2

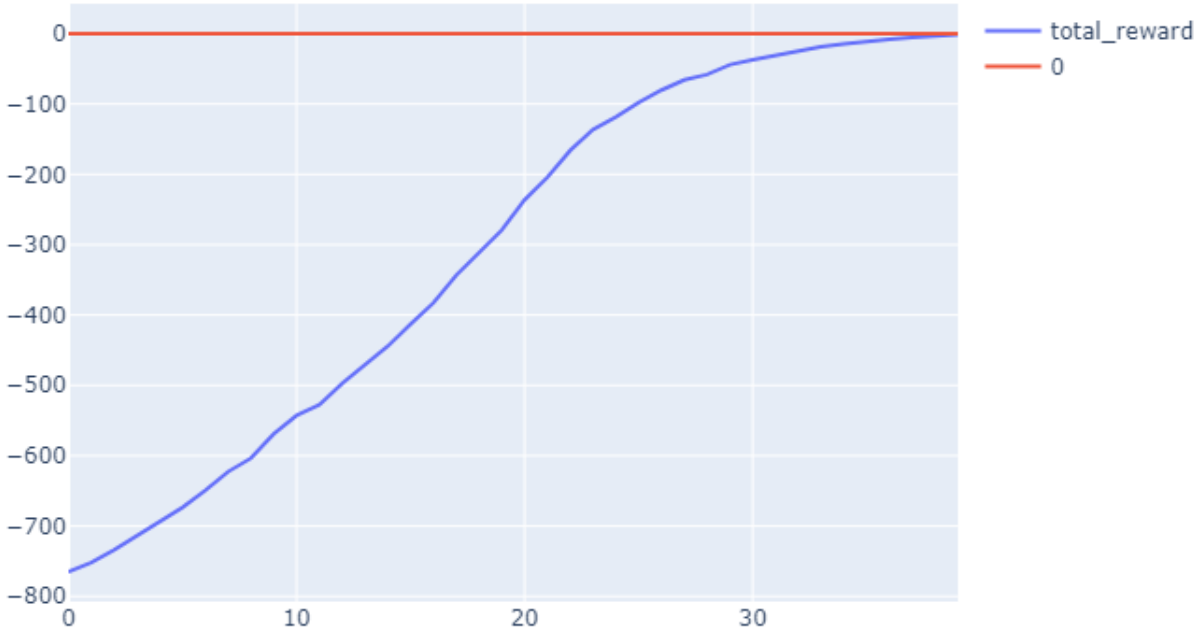
Реализовать алгоритм Кросс-Энтропии с двумя типами сглаживания, указанными в лекции 1. При выбранные в пункте 1 гиперпараметров сравнить их результаты с результатами алгоритма без сглаживания.

Решение

Начнем с **Policy smothing**. Также перебирал параметры для λ в policy smothing. Выбрал такие значения [0.1, 0.2, 0.5, 0.6, 0.7] и добавил их в параметры для перебора. Перебирал 40 наборов гиперпараметров, получил наилучшее среднее вознаграждение равное -12.8. Потом попробовал модель с параметрами из задания 1 и λ = 0.5, получил среднее вознагражение равное 1.048

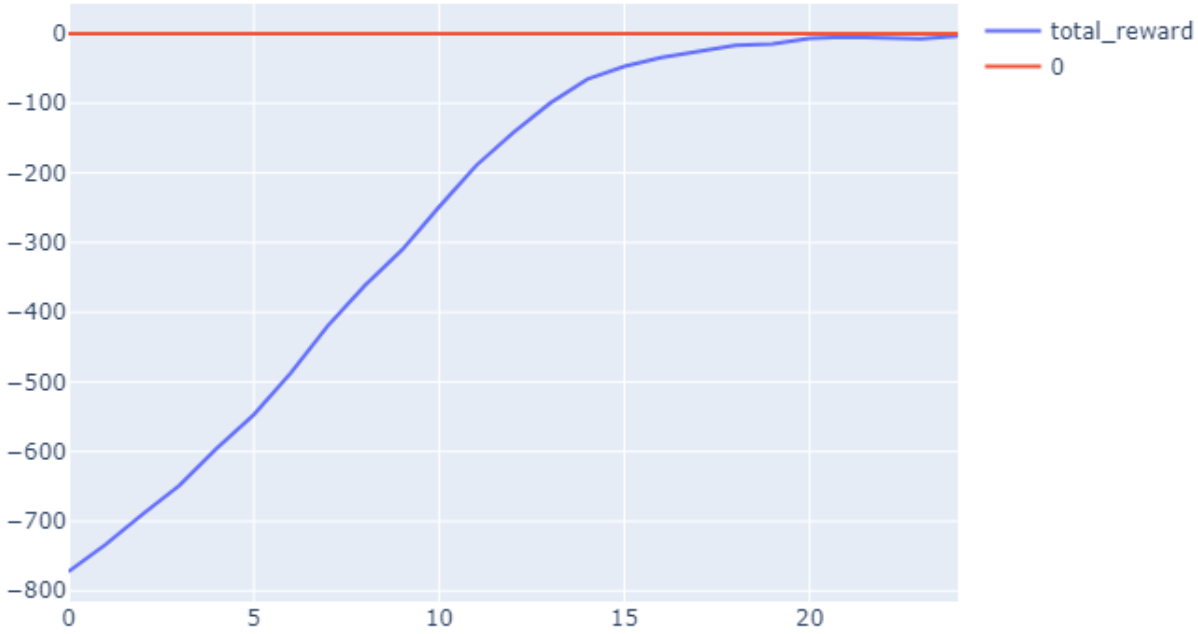
Итого параметры, с которыми получил наилучшую модель

q_param	**0.3**
episode_n	25
trajectory_n	500
trajectory_len	2000
λ	0.5



Laplace smoothing Перебирал параметры для λ из массива [0.01, 0.001, 0.0001, 0.00001] и с остальными параметрами лучшей модели из задачи 1, получил наилучшую модель с λ 0.001, которая получила среднее вознаграждение -0.914. Ее параметры

q_param	**0.3**
episode_n	25
trajectory_n	500
trajectory_len	2000
λ	0.001



Итого: модели со сглаживанием показали в среднем немного лучше результаты, чем CEM без сглаживания

Задача 3

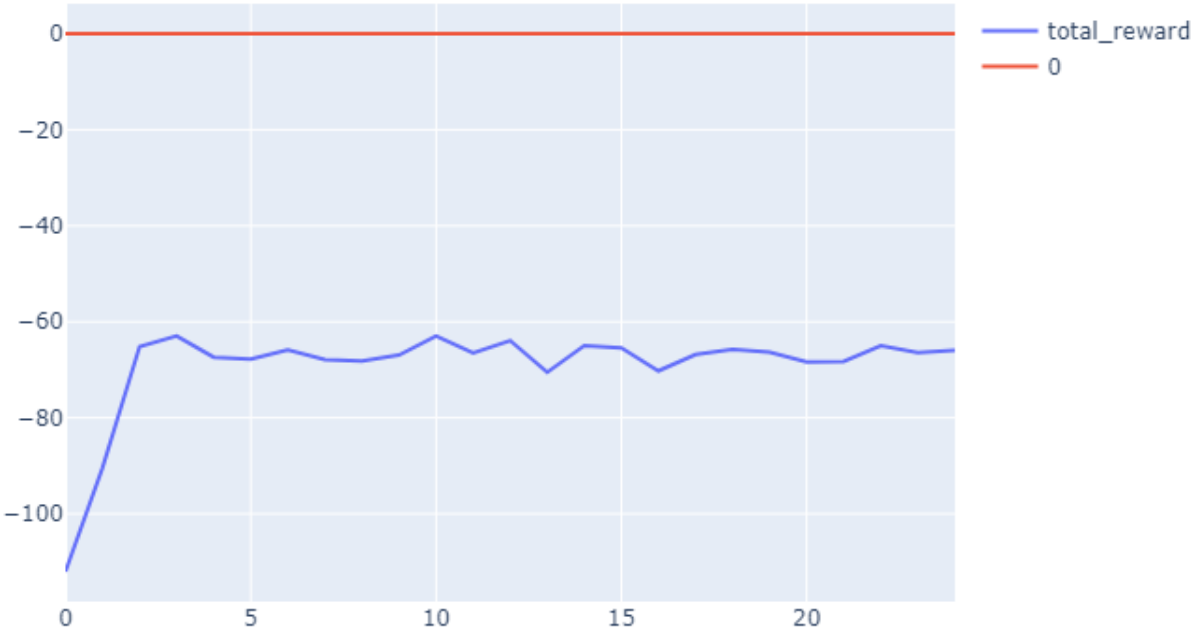
Реализовать модификацию алгоритм Кросс-Энтропии для стохастических сред, указанную в лекции 1. Сравнить ее результат с алгоритмами из пунктов 1 и 2.

Решение

Реализовал CEM с поправкой на стохастику, однако хороших результатов добиться не удалось. Пробовал следующие параметры. Подбирал алгоритм 10 раз

```
"m": [2, 4],
"q_param": [0.3, 0.4],
"episode_n": [12, 25],
"trajectory_n": [250, 500],
"trajectory_len": [20, 40, 80],
```

Лучший результат, который удалось получить -48.872



Также удалось отследить особенность, что алгоритм с поправкой на стохастику не переобучается

Итого

Наилучшим алгоритмом по результатам тестирования оказался CEM with policy smothing, который набрал положительный средний total reward, также хотелось бы отметить, что CEM with laplace smothing сходится к 0 быстрее всех алгоритмов, и CEM с поправкой на стохастическую среду не склонен к переобучению