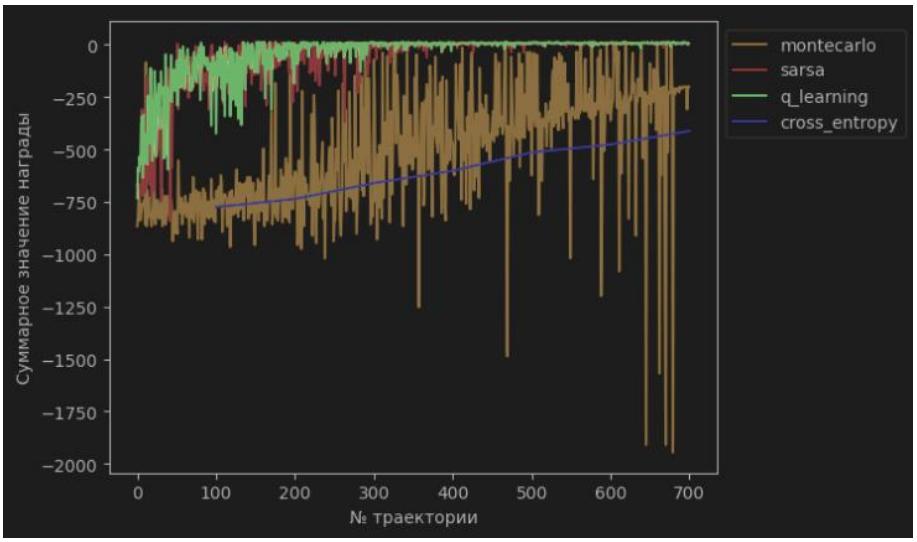
Задание # 1 **Реализовать Q-Learning и сравнить его результаты с реализованными ранее алгоритмами: Cross-Entropy, Monte Carlo, SARSA в задаче Taxi-v3**

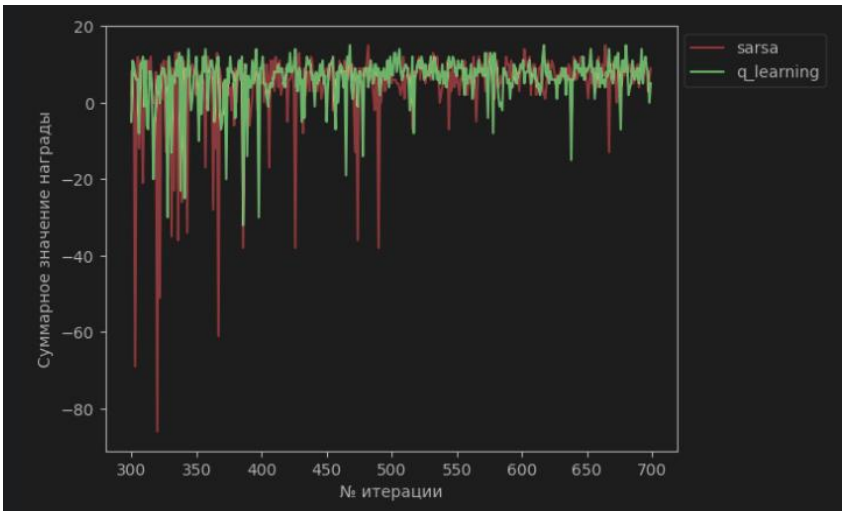
Отличие, Q-learning от SARSA в обновлении в qfunction:

```
qfunction[state][action] += alpha * (
    reward + gamma * max(qfunction[next_state]) - qfunction[state][action]
)
```

Реализовав Q-learning сравним алгоритмы: Cross-Entropy, Monte Carlo, SARSA, Q-Learning:



Видим, что алгоритм montecarlo нестабилен и сходится хуже с другими 3 алгоритмами. Что касается cross_entropy, то можно утверждать, что он стабильнее, так как монотонно возрастает саммарная награда, но медленнее с точки зрения сходимости. Чтобы понять какой из Q-learning и SARSA.



Sarsa по сравнению с Q-learning более нестабильная во время обучения, но итоговый результат близкий к одинаковому. Однако оба алгоритма сошлись, поэтому выделять лучшего не будем).

Задание # 2 Дискретизировать пространство состояний и обучить Агента решать Acrobot-v1

Рассмотрим решение зачачи Acrobot-v1 с помощью Monte Carlo, SARSA, Q-Learning.

Проблема заключается в том, что Пространство состояний = 6 и все значения типа float. Следовательно нужно разбить по бинам каждое значение для того чтобы комбинация этих параметров была уникальна и к ней можно было обращаться как к хэшу.

Каждый параметр состояния побил на 20 бинов и получил общее количество стейтов и сами бины:

```
state_ranges = np.vstack((env.observation_space.low, env.observation_space.high)).T
state_bins = [np.linspace(low, high, 20) for low, high in state_ranges]
for bins in state_bins:
    num_states += len(bins)
```

Рассмотрим на примере Q-learning:

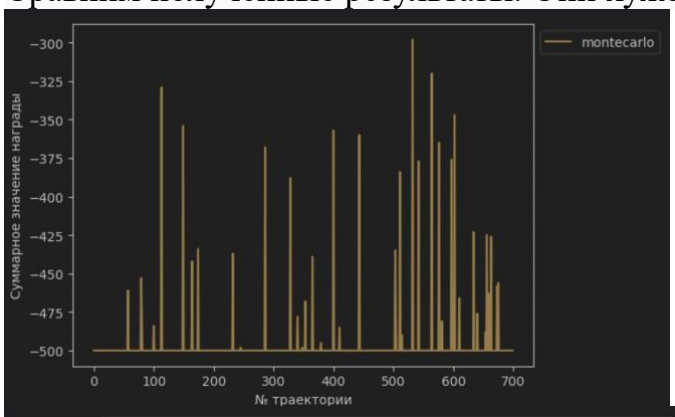
Qfunction – теперь словарь в котором мы кешируем состояния 2 введена функция discretize_state, для того чтобы определять к каким бинам относиться данные показатели state.

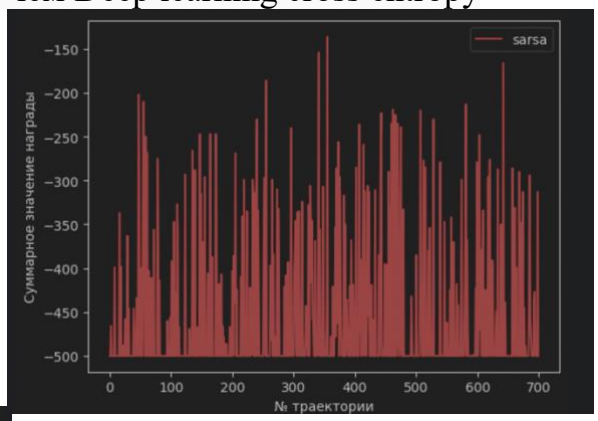
```
def q_learning(env, episode_n, gamma=0.99, trajectory_len=500, alpha=0.5):
    total_rewards = np.zeros(episode_n)
    action_n = 3
    num_states = 0
    n_finished = 0
    state_ranges = np.vstack((env.observation_space.low, env.observation_space.high)).T
    state_bins = [np.linspace(low, high, 20) for low, high in state_ranges]
    for bins in state_bins:
        num_states += len(bins)

    qfunction = dict()

    for episode in range(episode_n):
        epsilon = 1 / (episode + 1)

        state = env.reset()
        state = discretize_state(state, state_bins)
        if state not in qfunction:
            qfunction[state] = np.zeros(action_n)
        action = get_epsilon_greedy_action(qfunction[state], epsilon, action_n)
        for i in range(trajectory_len):
            next_state, reward, done, _ = env.step(action)
            next_state = discretize_state(next_state, state_bins)
            if next_state not in qfunction:
                qfunction[next_state] = np.zeros(action_n)

            qfunction[state][action] += alpha * (
                reward + gamma * max(qfunction[next_state]) - qfunction[state][action]
            )
```

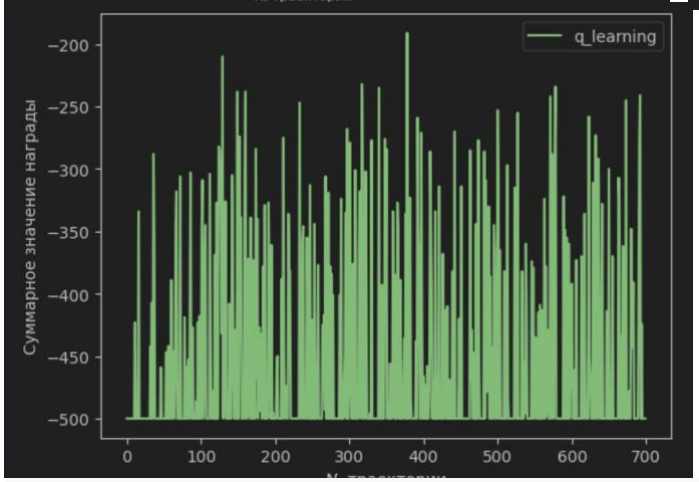
Код функции discretize_state

```python
def discretize_state(state, state_bins):
    p1, p2, p3, p4, p5, p6 = state
    theta1_bins, theta2_bins, sin_theta1_bins, sin_theta2_bins, theta1_dot_bins, theta2_dot_bins = state_bins
    theta1_bin = np.digitize(p1, theta1_bins)
    sin_theta1_bin = np.digitize(p2, sin_theta1_bins)   You, 17 minutes ago • Uncommitted changes
    theta2_bin = np.digitize(p3, theta2_bins)
    sin_theta2_bin = np.digitize(p4, sin_theta2_bins)
    theta1_dot_bin = np.digitize(p5, theta1_dot_bins)
    theta2_dot_bin = np.digitize(p6, theta2_dot_bins)
    return tuple([theta1_bin, sin_theta1_bin, theta2_bin, sin_theta2_bin, theta1_dot_bin, theta2_dot_bin])
```

Сравним полученные результаты. Они хуже чем Deep learning cross entropy

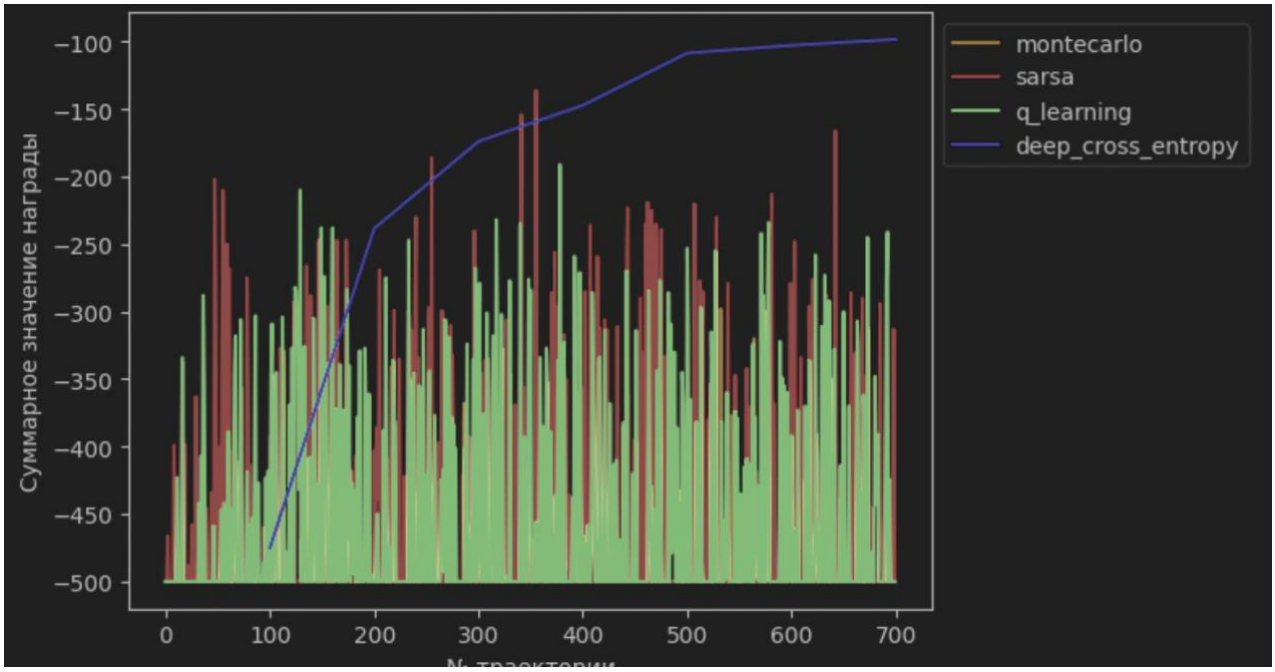






По графика видно, что хуже всего обучен montecarlo. Однако для выбора лучшего из новых алгоритмов я ввел метрику, показывающую в скольких траекториях, было найдено решение
montecarlo: 41
sarsa: 264
q-learning: 265

Снова незначительно лучше q_learning.
Однако посмотрев на график награды deep_cross_entropy становится понятно, что данный алгоритм намного лучше q- learning для данной задачи.


montecarlo
sarsa
q_learning
deep_cross_entropy
Суммарное значение награды
−100
−150
−200
−250
−300
−350
−400
−450
−500
0
100
200
300
400
500
600
700

Задание # 3 Придумать стратегию для выбора epsilon позволяющую агенту наилучшим образом решать Taxi-v3 алгоритмом Monte Carlo.

Я пробовал и возводить в степень в зависимости от итерации и пробовал брать -exp от степени. И тюнил модель, но побить дефолтную метрику montecarlo не удалось. Остановился на данном методе:

```
for episode in range(episode_n):
    epsilon = np.exp(-decay_rate * episode)
```

Это лучший затюненный результат с decay_rate = 0.009. Но как видимо на графике значительных улучшений это не принесло: