1. QLearning

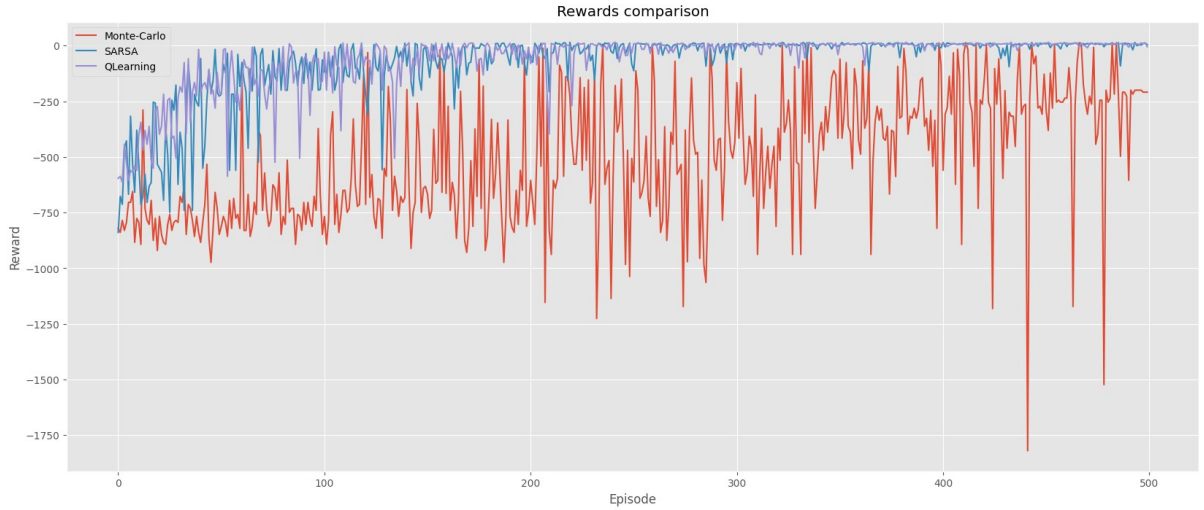
Написание алгоритма QLearning я начал с копирования описания и функции get_epsilon_greedy_action()

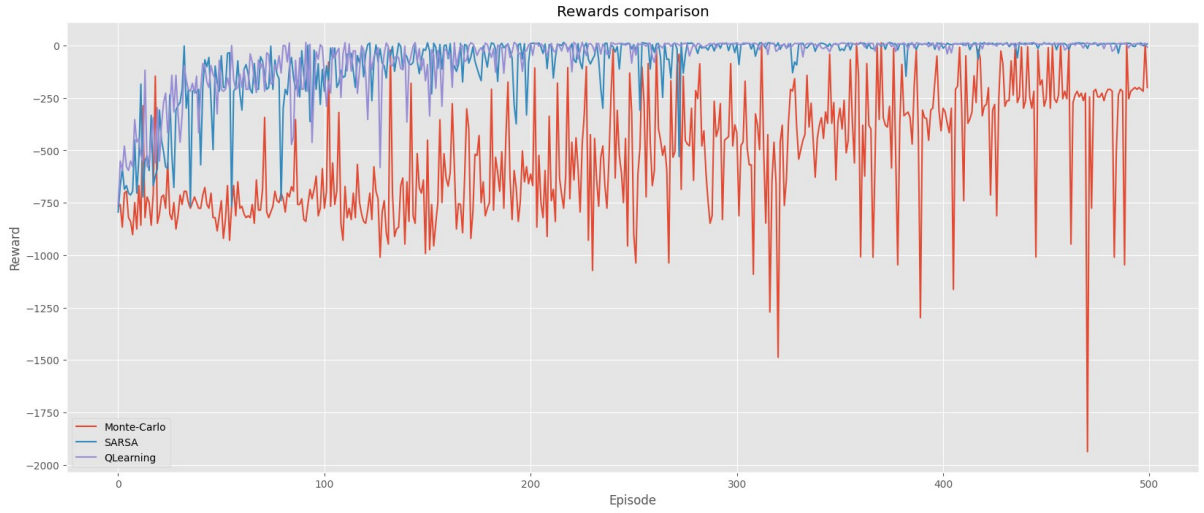
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_t + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t))$$

Основное отличие тут заключается в том, что нужно выбрать действие с максимальным q_value для следующего состояния

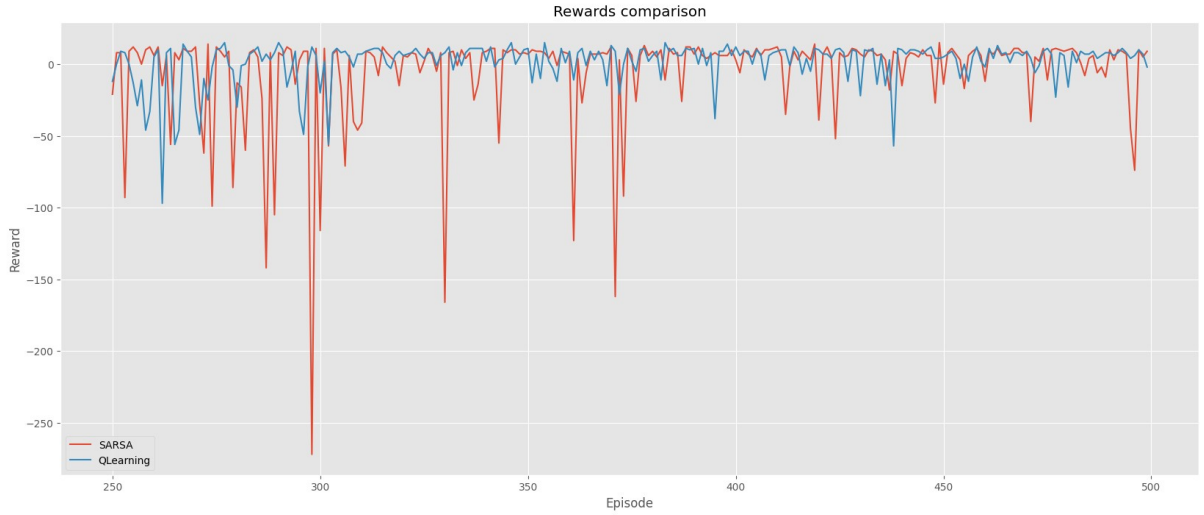
```
max_next_action = np.argmax(q[next_state])
q[state][action] = q[state][action] + alpha * (reward + gamma * q[next_state][max_next_action] - q[state][action])
```

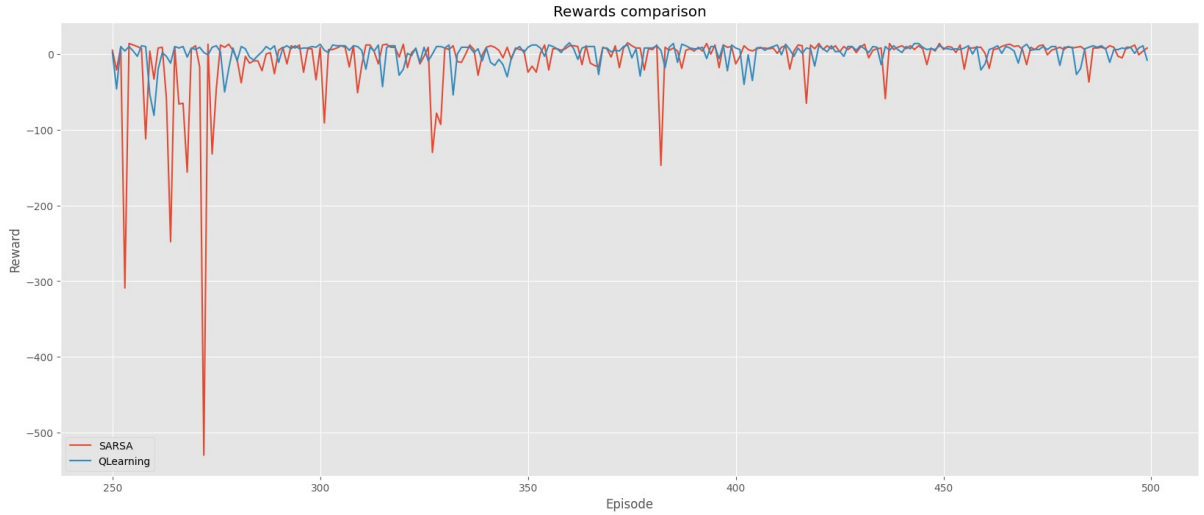
Собственно, на этом отличия и закончились, алгоритм работал, поэтому я перешел к сравнению.
Изначально я сравнивал только Monte-Carlo, SARSA и QLearning, так как CrossEntropy отличается довольно сильно.





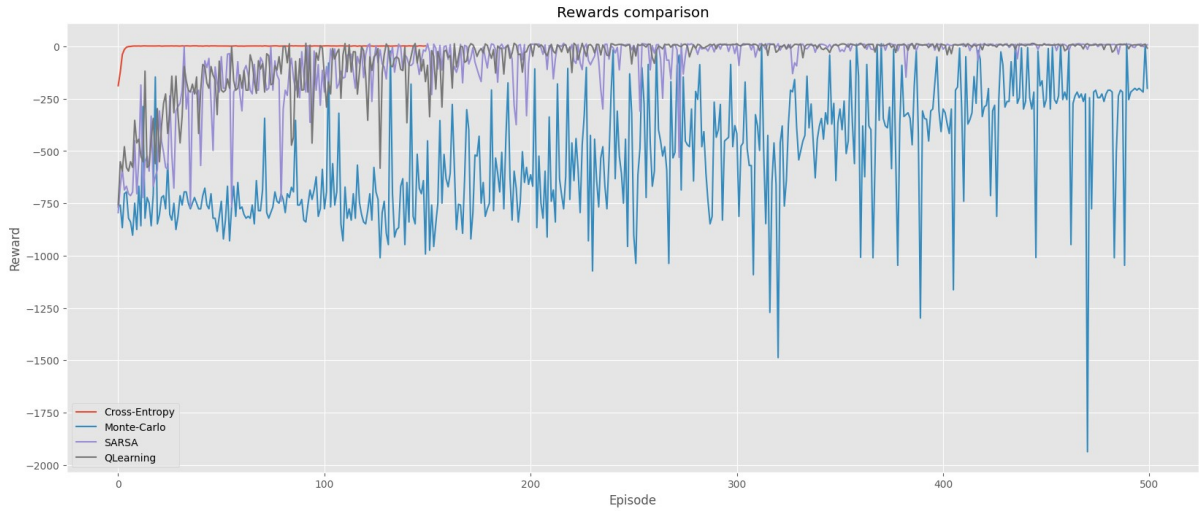
Вот общий график сравнения наград (запускал два раза для большего количества информации), как видно MonteCarlo сильно уступает, возможно я проделаю повторное сравнение после выполнения третьего задания этого дз. Также SARSA дает больше отрицательных выбросов на поздних стадиях обучения, это можно рассмотреть поближе (epsilon для алгоритмов однаковый: epsilon = 1 / (episode + 1)).



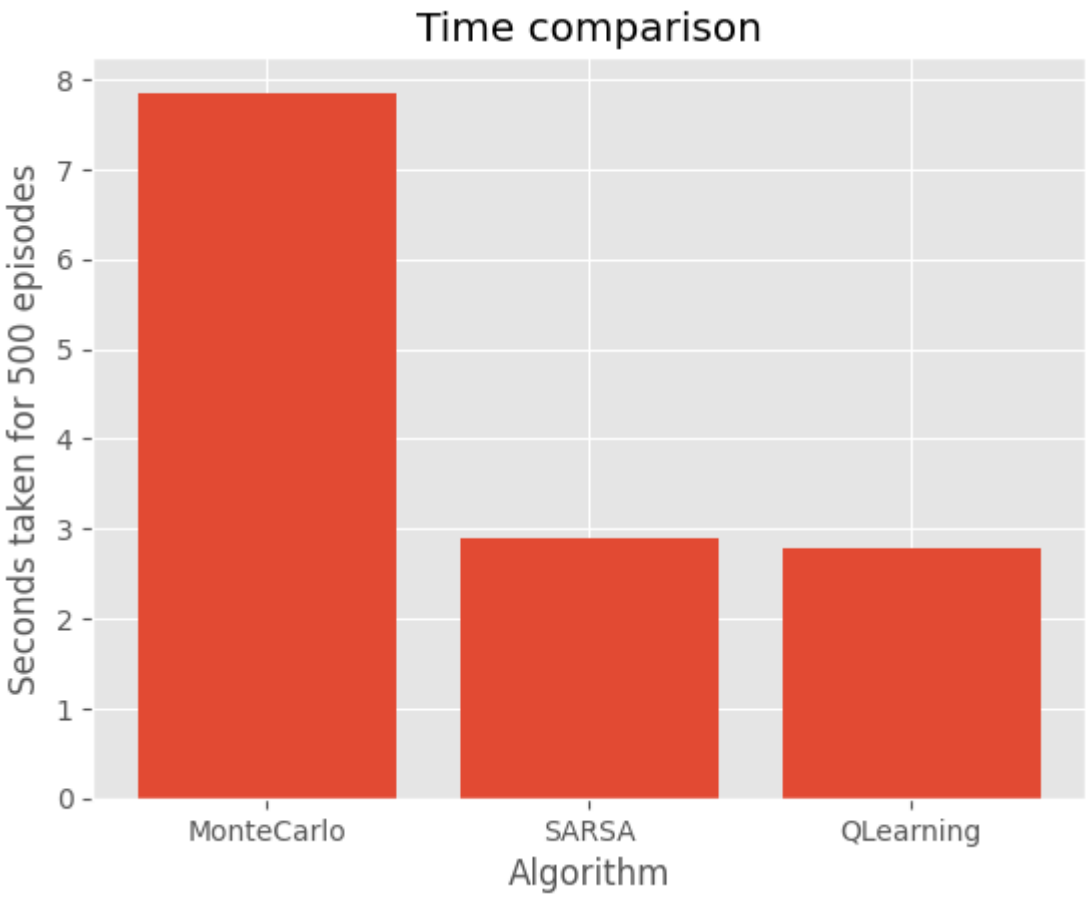


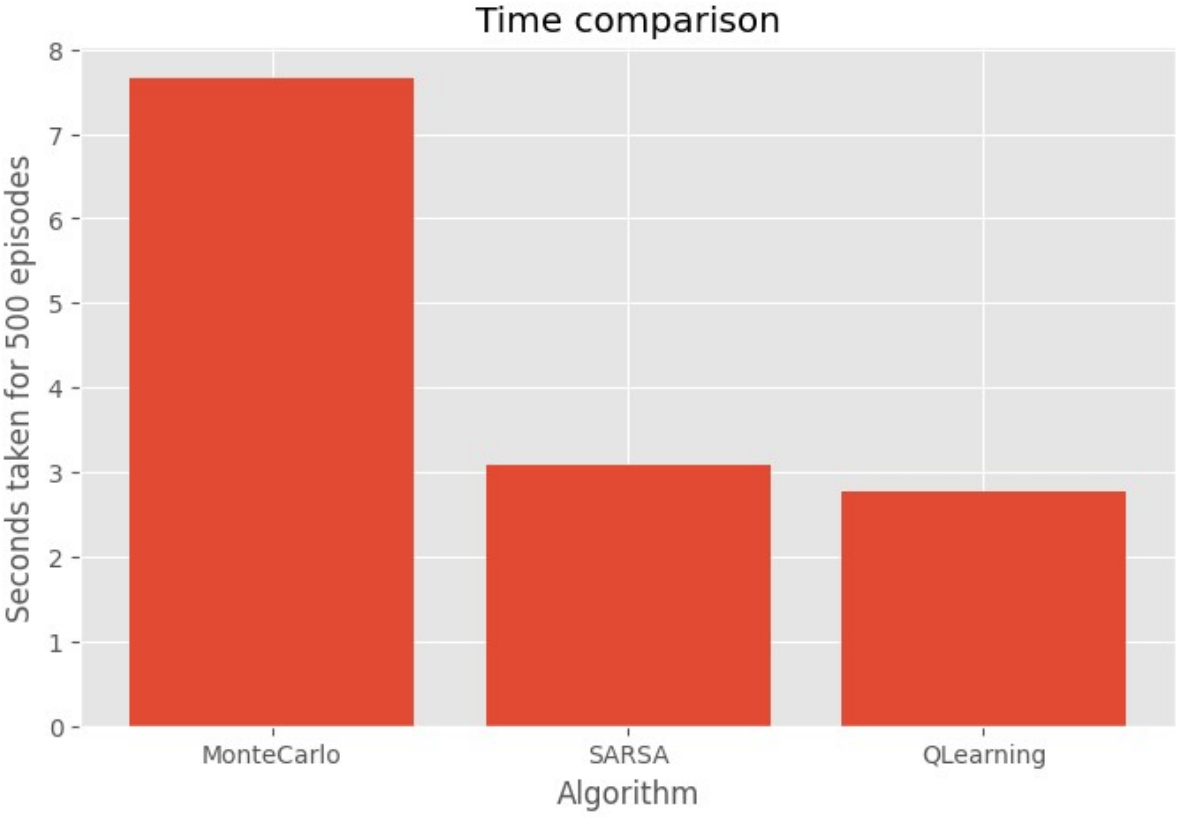
Вот это сравнение SARSA и QLearning для последних 250 эпох. Как видно, SARSA больше подвержен отрицательным выбросам, вероятнее всего это можно исправить, поставив epsilon = 0 после какого-то количества эпох.

Сравнение, включающее cross-entropy method

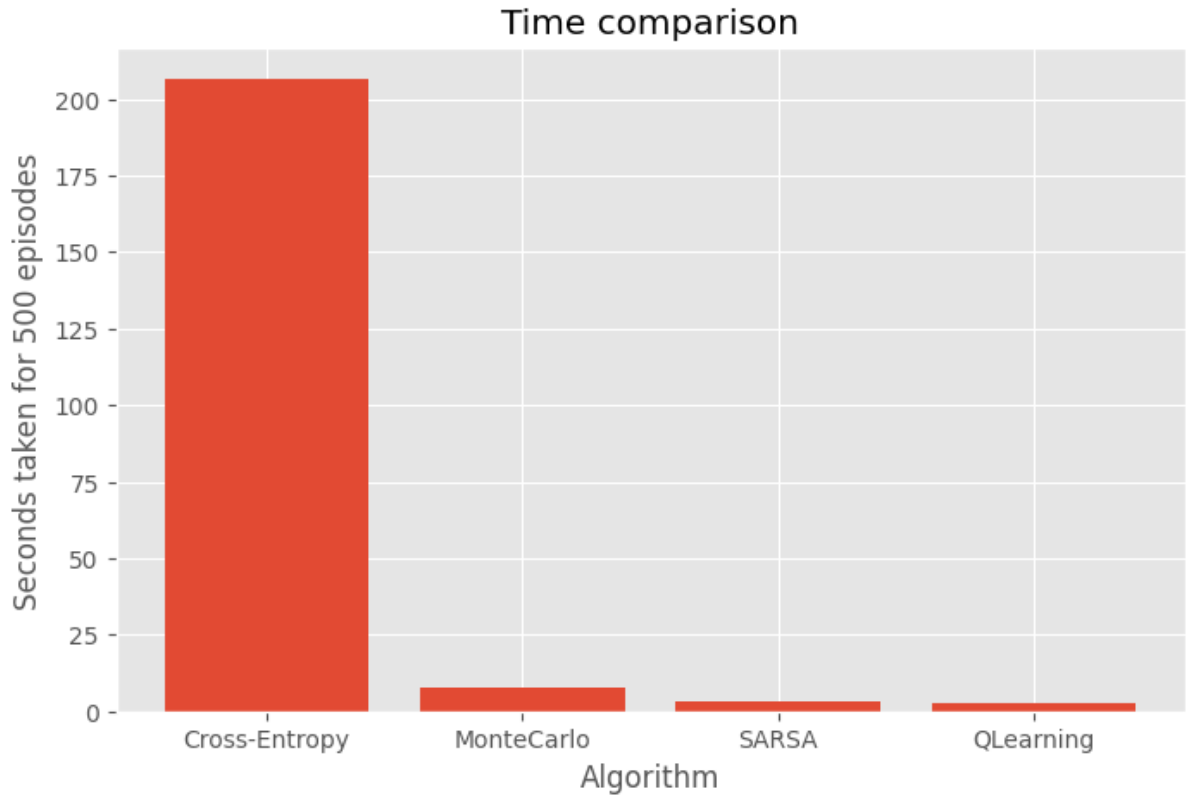


Как видно, cross-entropy достигает оптимального значения гораздо раньше и после выходит на плато. Однако, также стоит сравнить количество времени, затрачиваемое на обучение.
Сначала также представлю барплоты без cross-entropy:


Time comparison
Seconds taken for 500 episodes
0
1
2
3
4
5
6
7
8
MonteCarlo
SARSA
QLearning
Algorithm


Time comparison
Seconds taken for 500 episodes
0
1
2
3
4
5
6
7
8
MonteCarlo
SARSA
QLearning
Algorithm

Как видно, MonteCarlo требовал заметно больше времени на обучение, чем SARSA и QLearning, и это, вероятнее всего связано с недостаточно оптимизированной имплементацией алгоритма (много циклов).



И сравнение, включающее cross-entropy.

2. Acrobot

Я решил работать с Acrobot, так как работал с ним ранее. Пространство действий дискретно. Пространство состояний выглядит следующим образом:

Observation Space

The observation is a `ndarray` with shape `(6,)` that provides information about the two rotational joint angles as well as their angular velocities:

Num	Observation	Min	Max
0	Cosine of `theta1`	-1	1
1	Sine of `theta1`	-1	1
2	Cosine of `theta2`	-1	1
3	Sine of `theta2`	-1	1
4	Angular velocity of `theta1`	~ -12.567 (-4 * pi)	~ 12.567 (4 * pi)
5	Angular velocity of `theta2`	~ -28.274 (-9 * pi)	~ 28.274 (9 * pi)

theta1 - угол первой палки относительно Oy, theta2 - угол второй палки относительно первой.
Основное, что стоит заметить при попытке дискретизировать пространство событий, это тригонометрические свойства этого пространства. Самое очевидное, это то что для obs_space[0] и obs_space[1], равно как для obs_space[2] и obs_space[3] выполняется основное тригонометрическое тождество. Стоит отметить, что как связь между [0] и [1] так между [2] и [3] довольно проста и детерминирована тождеством, но связь между ([0], [1]) и ([2], [3]) скорее всего есть, но описать ее довольно сложно.
И тут ко мне закралась мысль, что первые четыре параметра пространства состояний можно дискетизировать, написав синусы и косинусы от 0 до 360.

```
a1 = np.linspace(0, 2 * np.pi, 360)
sins, coss = np.sin(a1), np.cos(a1)
```

360 точек (градусов) от нуля до двух пи.

Оставалось понять, откуда берется угловое ускорение. Это довольно непростая задача, как я понял в самой среде оно считается в этой функции. Однако, есть более прямолинейный способ: так как первые четыре параметра я решил ограничить 360 точками, а угловое ускорение в самой среде ограничено, как представлено на скриншоте выше, можно просто вызвыть np.linspace.

```
a2 = np.linspace(-4 * np.pi, 4 * np.pi, 360)
a3 = np.linspace(-9 * np.pi, 9 * np.pi, 360)
```

Далее оставалось понять, как все эти параметры связаны друг с другом.

Пока что на этом я остановился, планирую вернуться к заданию позже.