Тагиров Али, ИТМО

Задание 2.1

Пользуясь алгоритмом Кросс-Энтропии для конечного пространства действий обучить агента решать Acrobot-v1 или LunarLander-v2 на выбор. Исследовать гиперпараметры алгоритма и выбрать лучшие.

Для первого задания выбрал агента Acrobot-v1, с LunarLander-v2 возникли проблемы с зависимостями на Windows.

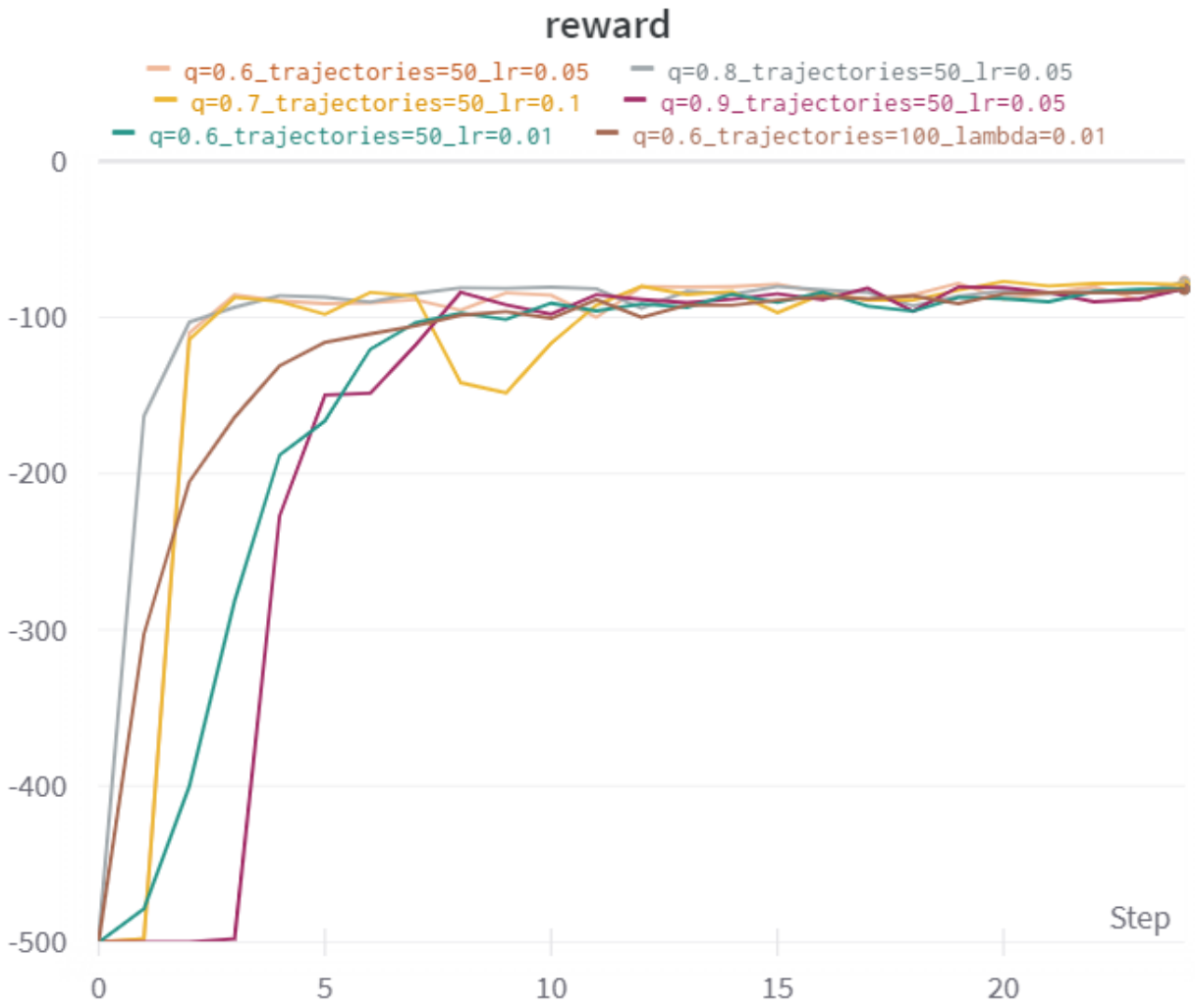
Применил алгоритмом Кросс-Энтропии для конечного пространства действий, который мы разбирали на семинаре.

Были исследованы результаты со следующими гиперпараметрами:

Сиды зафиксированы

- q: [0.6, 0.7, 0.8, 0.9]
- количество траекторий: [20, 30, 50, 100]
- learning rate: [0.01, 0.05, 0.1]
- количество итераций: 25

learning_rate	q_param	trajectories	reward ▾
0.05	0.6	50	-76.66
0.05	0.8	50	-77.52
0.1	0.7	50	-79.24
0.05	0.9	50	-81.62
0.01	0.6	50	-81.82



Лучше результаты были получены при обучении с lr = 0,05, 50 траекторий оказалось достаточно. Алгоритм обучается уже к 7 итерации. Причем видно, что высокий learning rate в связке с менее качественными траекториями приводят к более быстрому схождению алгоритма, в случае с более элитными траектории разница не так сильно заметна.

Задание 2.2

Реализовать алгоритм Кросс-Энтропии для непрерывного пространства действий. Обучить агента решать Pendulum-v1 или MountainCarContinuous-v0 на выбор. Исследовать гиперпараметры алгоритма и выбрать лучшие.

В задаче обучение агента MountainCarContinuous-v0, действие принимает значения в диапазоне [-1, 1], поэтому в качестве функции активации был выбран гиперболический тангенс, имеющую тот же диапазон.

В качестве функции потерь взял MSE

```
self.network = nn.Sequential(
    nn.Linear(self.state_dim, 128),
    nn.ReLU(),
    nn.Linear(128, 1)
)
self.tanh = nn.Tanh()
self.optimizer = torch.optim.Adam(self.parameters(), lr=1e-2)
self.loss = nn.MSELoss()
```

$$Loss(\theta) = \frac{1}{|\mathcal{T}_n|} \sum_{(a|s) \in \mathcal{T}_n} \|\pi^{\theta_n}(s) - a\|^2$$

```
def get_action(self, state):
    state = torch.FloatTensor(state)
    noise = torch.FloatTensor(state.shape[0]).uniform_(-0.01, 0.01)
    logits = self.forward(state) + noise
    return self.tanh(logits).data.numpy()
```

Обучить агента не удалось, пробовал подобрать гиперпараметры, также усложнял архитектуры нейронной сети, но обучить так и не удалось. Награды изменяются в процессе обучения не сильно топчась около нуля.

Вывод

Алгоритм Кросс-Энтропии для конечного пространства действий хорошо справился с обучением агента решать Acrobot-v1, наилучшие результаты были получены

Алгоритм Кросс-Энтропии для непрерывного пространства действий не справился с обучением агента решать MountainCarContinuous-v0. Данный алгоритм не подходит для решения этой задачи