Домашнее задание №2

2.1 Пользуясь алгоритмом Кросс-Энтропии для конечного пространства действий обучить агента решать Acrobot-v1. Исследовать гиперпараметры алгоритма и выбрать лучшие.

In [1]:
```
from acrobot import CEMDL
from acrobot import CEMDL_update
from acrobot import main_func
import gym
import matplotlib.pyplot as plt
import numpy as np
```

```
Warning: Gym version v0.24.0 has a number of critical issues with `gym.make` such that the `reset` and `step` fu
nctions are called before returning the environment. It is recommend to downgrading to v0.23.1 or upgrading to v
0.25.1
```

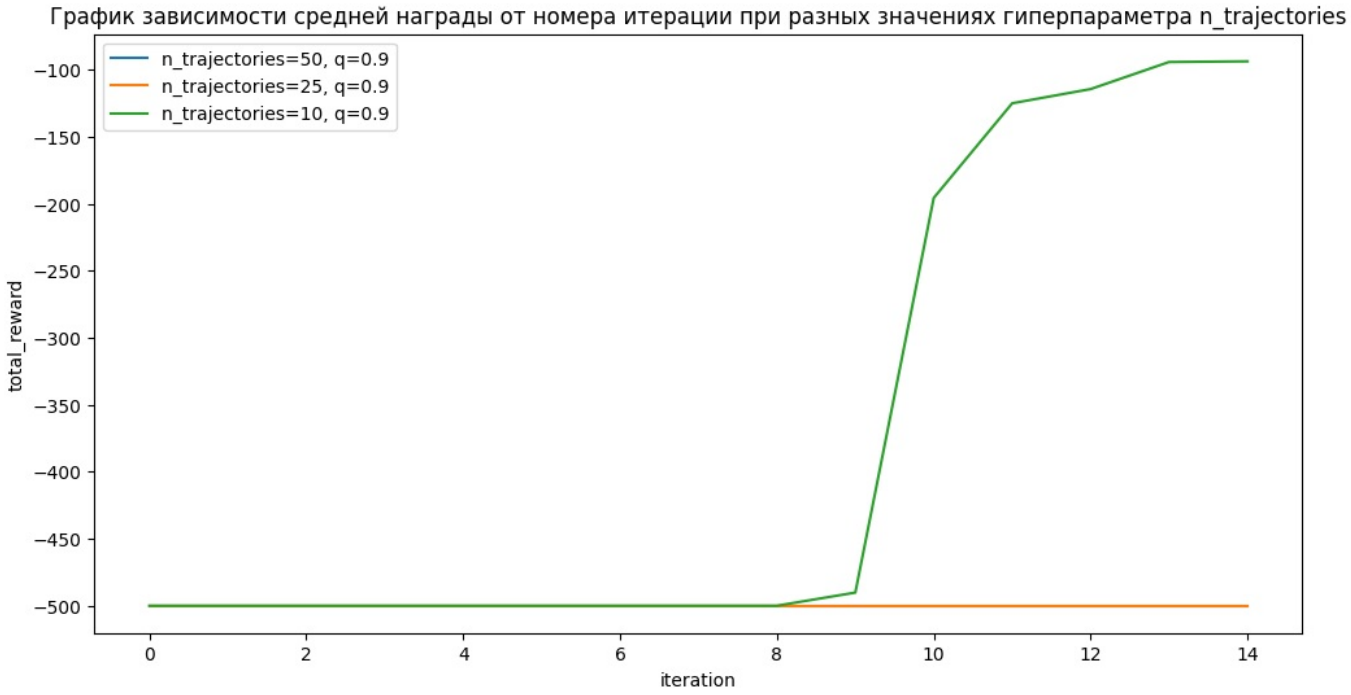
In [2]:
```
env = gym.make('Acrobot-v1')
action_n = 3
```

In [3]:
```
agent = CEMDL(action_n, eps=0.0)
rewards_n50_q09 = main_func(env, agent, n_iteration=15, n_trajectories=50, q=0.9)
```

In [4]:
```
agent = CEMDL(action_n, eps=0.0)
rewards_n25_q09 = main_func(env, agent, n_iteration=15, n_trajectories=25, q=0.9)
```

In [5]:
```
agent = CEMDL(action_n, eps=0.0)
rewards_n10_q09 = main_func(env, agent, n_iteration=15, n_trajectories=10, q=0.9)
```

In [6]:
```
plt.figure(figsize=(12,6))
plt.plot(np.arange(0, 15), rewards_n50_q09, label='n_trajectories=50, q=0.9')
plt.plot(np.arange(0, 15), rewards_n25_q09, label='n_trajectories=25, q=0.9')
plt.plot(np.arange(0, 15), rewards_n10_q09, label='n_trajectories=10, q=0.9')
plt.title('График зависимости средней награды от номера итерации при разных значениях гиперпараметра n_trajecto
plt.xlabel('iteration')
plt.ylabel('total_reward')
plt.legend()
plt.savefig('plot_n_trajectories.png')
plt.show()
```



Основная цель этой картинки показать, что алгоритм больше зависит не от количества траекторий (хотя косвенно от него, но лишь потому что чем их больше, тем больше все таки шанс что получится решить задачу хоть у одной), а от начальной инициализации весов (при неудачной инициализации алгоритму так и не удается найти решение и награда остается -500, как показано на картинке выше). Также очевидно необходимость задания маленького значения q - учиться надо на решающих задачу траекториях.

Решением этой проблемы является внедрение параметра eps, который для данной задачи будет означать что действовать по случайной стратегии до первого получения элитных траекторий (вхождении в fit, где он зануляется),а дальше переходить к обучению. Продемострируем что описанный подход работает.

In [3]:
```
rewards_eps_0 = []
for _ in range(100):
    agent = CEMDL(action_n, eps=0.0)
    rewards_eps_0.append(main_func(env, agent, n_iteration=15, n_trajectories=10, q=0.9)[-1])
```

In [4]:
```
rewards_eps_1 = []
for _ in range(100):
    agent = CEMDL(action_n, eps=1.0)
    rewards_eps_1.append(main_func(env, agent, n_iteration=15, n_trajectories=10, q=0.9)[-1])
```

In [5]:
```
len_eps_0 = 0
for reward in rewards_eps_0:
    if reward < -490.:
        len_eps_0 += 1
print(f'при отсутствии эпсилон обучение не сошлось в {len_eps_0}%')
```

```
при отсутствии эпсилон обучение не сошлось в 41%
```

In [6]:
```
len_eps_1 = 0
for reward in rewards_eps_1:
    if reward < -490.:
        len_eps_1 += 1
print(f'при включении эпсилон обучение не сошлось в {len_eps_1}%')
```

```
при включении эпсилон обучение не сошлось в 87%
```

Очень интересный результат: получается, что инициализация рандомных весов нейросети почти в 2 раза лучше равномерного распределения по действиям. Гипотеза, что это является следствием наличия в равномерном распределении действия действия 1 (ничего не делать). Изменим распредление вероятностей действия убрав вариант ничего не делать и проведем тест.

In [3]:
```
rewards_eps_1_update = []
for _ in range(100):
    agent = CEMDL_update(action_n, eps=1.0)
    rewards_eps_1_update.append(main_func(env, agent, n_iteration=15, n_trajectories=10, q=0.9)[-1])
```

In [4]:
```
len_eps_1_update = 0
for reward in rewards_eps_1_update:
    if reward < -490.:
        len_eps_1_update += 1
print(f'при включении изменении равномерной политики (исключении действия "ничего не далать") обучение не сошло
```

```
при включении изменении равномерной политики (исключении действия "ничего не далать") обучение не сошлось в 10%
```

Вывод по заданию 1:

При работе в среде Acrobot-v1 на первый план выходит задача получения хотя бы одного положительного решения задачи, что может достигаться за счет увеличения количества траекторий, введением выполнения задачи равномерной политикий (с учетом отсутствия бездействия) или переинициализацией агента без эпсилон (нового распределения весов) если на нескольких шагах получаем награду -500.

In [ ]:
```
```