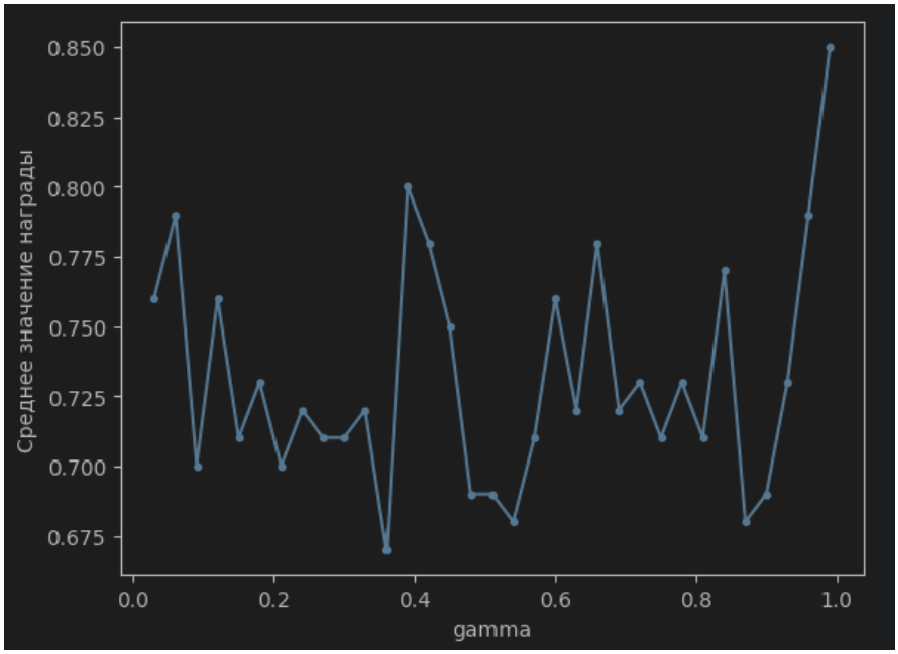
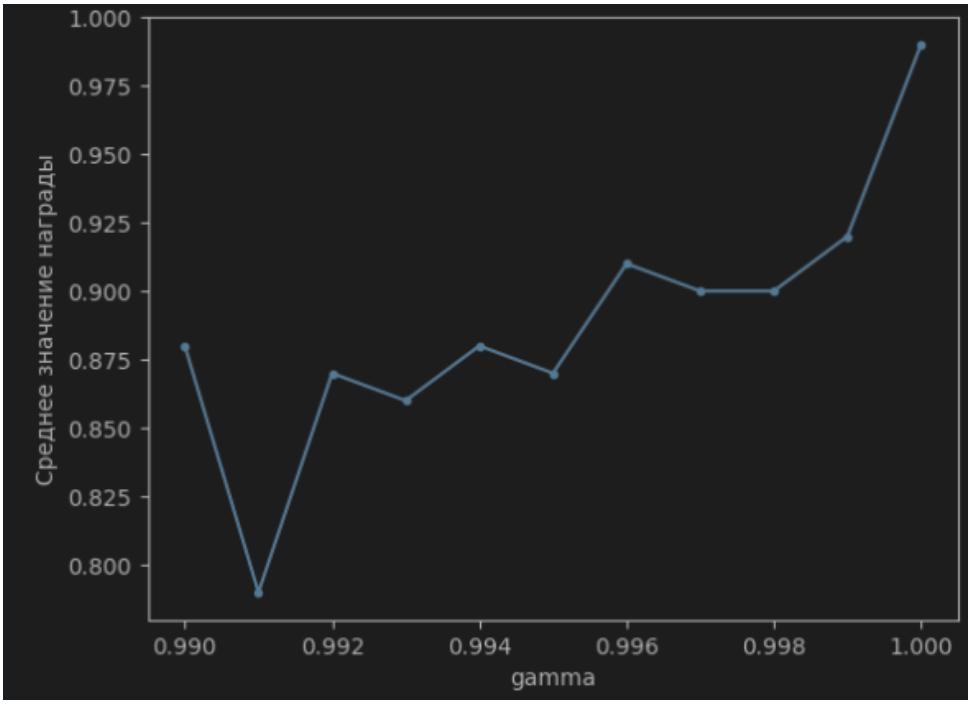
Задание # 1 Policy Iteration tuning hyperparameter gamma

Перебрав параметр gamma от 0.03 до 0.99 с шагом 0.03 видим значительный прирост средней награды в значении gamma = 0.99.



Проведем дополнительное исследование проверим промежуток от 0.99 до 1. И получаем следующий результат, что лучшим значением гиперпараметра gamma является единица.

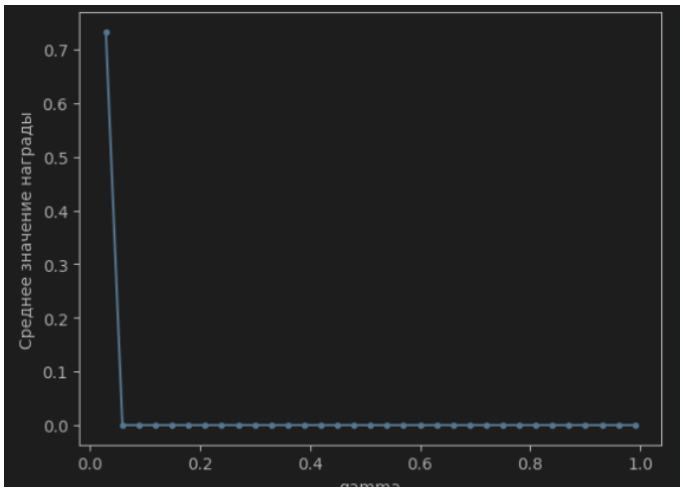


Задание # 2 На каждом шаге Policy Evaluation начинать с values обученных на предыдущем шаге

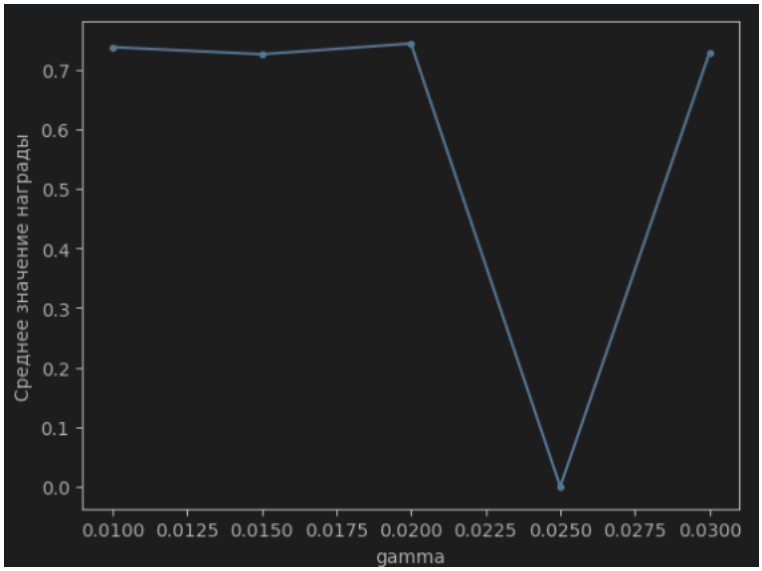
Переписали шаг политики обновления следующим образом. Больше не инициализируем новые v_values:

```
def policy_evaluation_step(env, v_values, policy, gamma):
    q_values = get_q_values(env, v_values, gamma)
    for state in env.get_all_states():
        for action in env.get_possible_actions(state):
            v_values[state] += policy[state][action] * q_values[state][action]
    return v_values
```

Однако после данного изменения алгоритм перестал сходиться при gamma > 3.



Более того, даже при значениях gamma <= 0.03 алгоритм не всегда сходится:

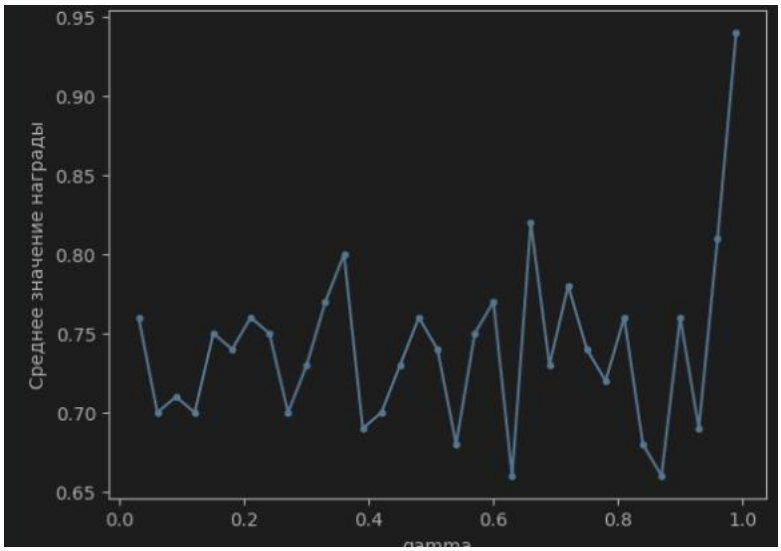


Казалось бы результат должен был быть лучше, засчет прироста к скорости сходимости, однако мы видим, что наоборот видимо нарушение

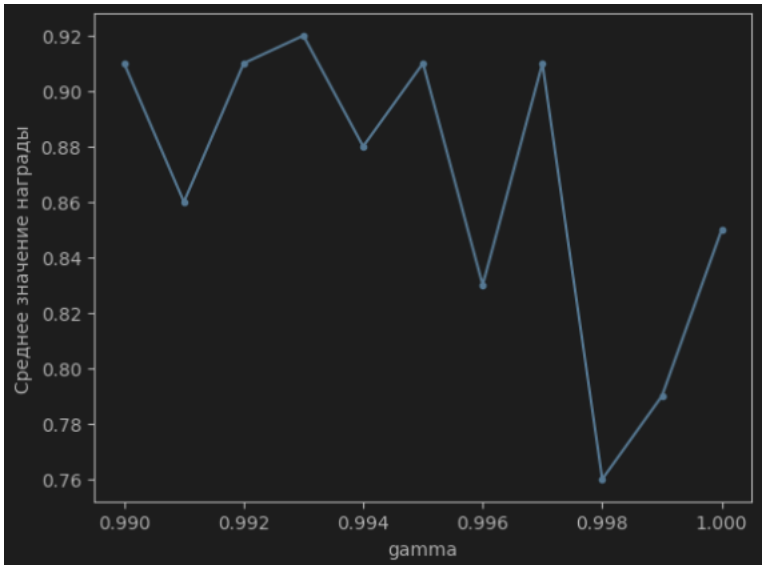
независимости v_values между итерациями приводит к несходимости алгоритма.

Задание # 3 Value Iteration tuning hyperparameter gamma

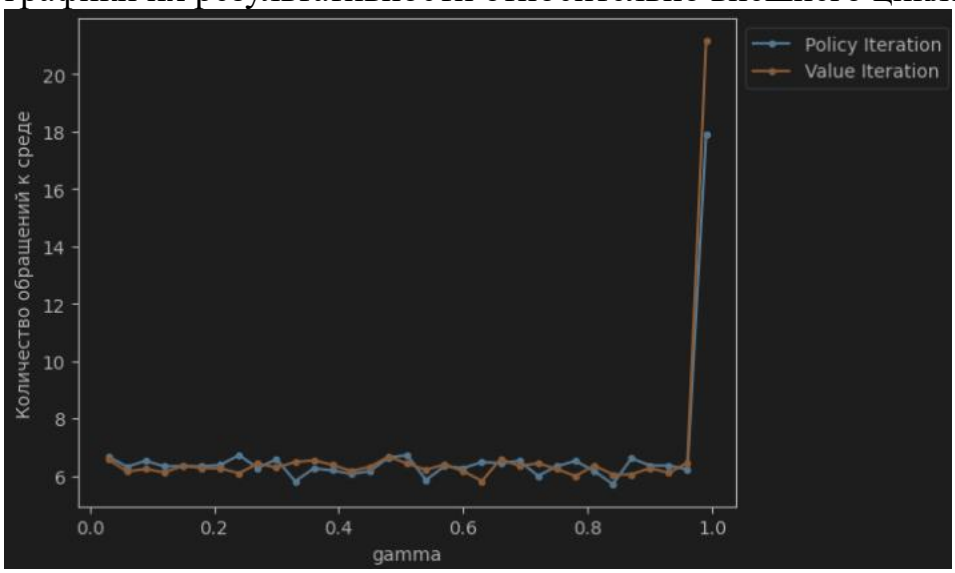
Перебрав параметр gamma от 0.03 до 0.99 с шагом 0.03 видим значительный прирост средней награды в значении gamma = 0.99.



Проведем дополнительное исследование проверим промежуток от 0.99 до 1. И получаем следующий результат, что лучшим значением гиперпараметра gamma является 0.993.



Сравним Policy Iteration и Value Iteration по количеству обращений к среде (кол-во итераций до финиша в среднем). (Поскольку в Policy Iteration есть еще внутренний цикл, то адекватным сравнением алгоритмов будет не графики их результативности относительно внешнего цикла)



Из графика невозможно, точно определить какой алгоритм быстрее.
Посмотрим на лучшие показатели гамма и метрики для двух алгоритмов:

- Policy iteration: {'gamma': 1.0, 'mean_total_rewards': 0.98, 'env_infer_count': 127.03}
- Value iteration: {'gamma': 0.993, 'mean_total_rewards': 0.92, 'env_infer_count': 18.26}
  Награда выше у Policy iteration, однако быстрее находит финиш Value iteration.

Сравнив инфренс на всех этапах тоже получил равенство, однако по средней награде лучше все-таки value-iteration:

```
df_policy = pd.concat([pd.DataFrame.from_dict(res1_1), pd.DataFrame.from_dict(res1_2)])
df_policy = df_policy.rename(columns={"env_infer_count": "policy_env_infer_count", "mean_total_rewards": "policy_mean_total_rewards"})

df_value = pd.concat([pd.DataFrame.from_dict(res3_1), pd.DataFrame.from_dict(res3_2)])
df_value = df_value.rename(columns={"env_infer_count": "value_env_infer_count", "mean_total_rewards": "value_mean_total_rewards"})

res_df = df_policy.merge(df_value, on="gamma")
res_df.loc[:, "is_policy_faster"] = res_df.policy_env_infer_count > res_df.value_env_infer_count
res_df.loc[:, "is_policy_better"] = res_df.policy_mean_total_rewards > res_df.value_mean_total_rewards
res_df.is_policy_faster.value_counts()
```

Executed at 2023.11.12 17:52:09 in 10ms

2 rows · Length: 2, dtype: int64 pd.Series

is_policy_faster	count
True	23
False	23

```
res_df.is_policy_better.value_counts()
```

Executed at 2023.11.12 17:52:17 in 4ms

2 rows · Length: 2, dtype: int64 pd.Series

is_policy_better	count
False	26
True	20

Вывод: Для разных gamma различные методы являются лучшими.