Тагиров Али (ИТМО)

Задание 3.1

На графике видно, чем ближе параметр gamma к единице, чем больше средний reward, при этом видно, что при gamma < 0.99 средний reward практически не изменятся.

Задание 3.2

Изменил метод так, чтобы на каждой итерации брались v_values с прошлой итерации

```
def policy_evaluation(policy, gamma, v_values, eval_iter_n):
    # v_values = init_v_values()
    for _ in range(eval_iter_n):
        v_values = policy_evaluation_step(v_values, policy, gamma)
    q_values = get_q_values(v_values, gamma)
    return q_values, v_values
```

```
v_values = init_v_values()
for _ in range(iter_n):
    q_values, v_values = policy_evaluation(policy, gamma, v_values, eval_iter_n)
    policy = policy_improvement(q_values)
```

На графиках видно, что при использовании на шаге police evolution v_values с предыдущего шага, средний reward начинает значительно увеличиваться при значениях gamma > 0.995. При использовании такого подхода уменьшается разброс values и они начинают сходится

```
{(0, 0): 0.012312958937995513, (0, 1): 0.010474433658493836,
{(0, 0): 0.729610961361448, (0, 1): 0.6097084495333395, (0,
{(0, 0): 0.2630281308113109, (0, 1): 0.21340803874164987, (0
{(0, 0): 0.742249419641088, (0, 1): 0.6528496551276499, (0,
{(0, 0): 0.5504940042737085, (0, 1): 0.5086824356001282, (0,
{(0, 0): 0.742249419641088, (0, 1): 0.6528496551276499, (0,
{(0, 0): 0.5504940042737085, (0, 1): 0.5086824356001282, (0,
{(0, 0): 0.742249419641088, (0, 1): 0.6528496551276499, (0,
{(0, 0): 0.5504940042737085, (0, 1): 0.5086824356001282, (0,
{(0, 0): 0.742249419641088, (0, 1): 0.6528496551276499, (0,
{(0, 0): 0.5504940042737085, (0, 1): 0.5086824356001282, (0,
{(0, 0): 0.742249419641088, (0, 1): 0.6528496551276499, (0,
{(0, 0): 0.5504940042737085, (0, 1): 0.5086824356001282, (0,
{(0, 0): 0.742249419641088, (0, 1): 0.6528496551276499, (0,
{(0, 0): 0.5504940042737085, (0, 1): 0.5086824356001282, (0,
```

```
{(0, 0): 0.012312958937995513, (0, 1): 0.010474433658493836,
{(0, 0): 0.7296396685420895, (0, 1): 0.6097213578983437, (0,
{(0, 0): 0.7772052116474018, (0, 1): 0.6878968358339018, (0,
{(0, 0): 0.8307823392896284, (0, 1): 0.7875524945012631, (0,
{(0, 0): 0.8429449469650306, (0, 1): 0.8190767446889854, (0,
{(0, 0): 0.8467705624791814, (0, 1): 0.8279174001621283, (0,
{(0, 0): 0.8478933607759549, (0, 1): 0.8304117122525414, (0,
{(0, 0): 0.8491299454780094, (0, 1): 0.8315647299546773, (0,
{(0, 0): 0.8502389075007124, (0, 1): 0.8326395918318343, (0,
{(0, 0): 0.8511633412481802, (0, 1): 0.8335501961246121, (0,
{(0, 0): 0.8519339243946452, (0, 1): 0.8343109916288797, (0,
{(0, 0): 0.8525762586026665, (0, 1): 0.8349453746436986, (0,
{(0, 0): 0.853111688091181, (0, 1): 0.8354742010065477, (0,
{(0, 0): 0.8535580051396734, (0, 1): 0.8359150168482203, (0,
{(0, 0): 0.853930040877592, (0, 1): 0.8362824673062427, (0,
```

до

после

Задание 3.3

Были исследованы результаты со следующими гиперпараметрами:

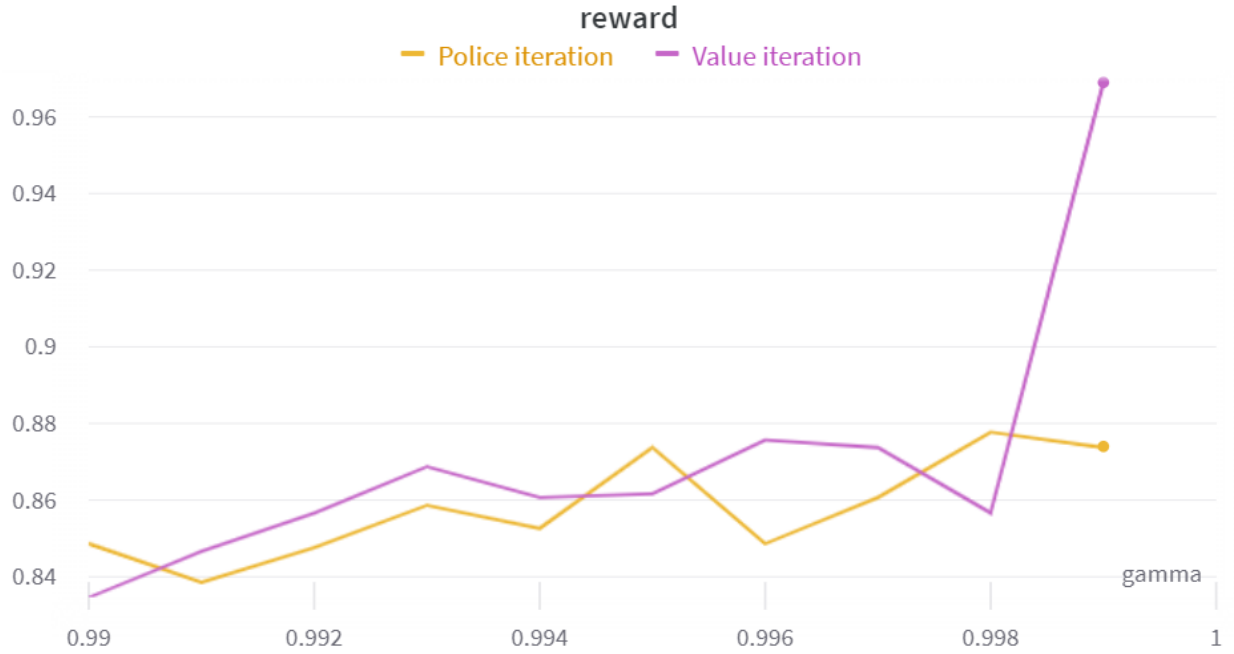
Сиды зафиксированы

- gama: [0.6, 0.7, 0.8, 0.85, 0.9, 0.95, 0.99, 0.995, 0.999]
- количество итераций: [30, 50, 100, 300]

gamma	iter_n	reward ▾
0.999	100	0.965
0.999	300	0.963
0.999	50	0.878
0.995	100	0.873
0.995	300	0.867

Па аналогии с алгоритмом police iteration параметр gamma следует выбирать наиболее близким к единице. 100 итерации хватило для достижения практически идеального результата.

Сравнение двух алгоритмов:



Как видно на рисунке Value iteration показывает более лучший результат, причем существенная разница между результатами получается только при gamma = 0.999. При этом Value Iteration показывает практически такие же результаты, как Police Iteration с использованием values обученных на предыдущем шаге.