P. Adam Aboud

500883647

CPS-824 Reinforcement Learning

Assignment 2

1. C. The accuracy was 100% for MC GLIE in the deterministic environment.

```
460 if __name__ == "__main__":
461     # comment/uncomment these lines to switch between deterministic/
        stochastic environments
462     env = gym.make("Deterministic-4x4-FrozenLake-v0")
463     #env = gym.make("Stochastic-4x4-FrozenLake-v0")
464
465     print("\n" + "-" * 25 + "\nBeginning First-Visit Monte Carlo\n" +
        "-" * 25)
466     Q_mc, policy_mc = mc_glie(env, iterations=1000, gamma=0.9)
467     test_performance(env, policy_mc)
468     # render_single(env, policy_mc, 100) # uncomment to see a single
        episode
469
```

```
TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE                    1: cmd
Microsoft Windows [Version 10.0.19041.867]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\pabou\Documents\GitHub\CPS824-RL>cd CPS824-Project

C:\Users\pabou\Documents\GitHub\CPS824-RL\CPS824-Project>cd ..

C:\Users\pabou\Documents\GitHub\CPS824-RL>cd cps824_assignment2

C:\Users\pabou\Documents\GitHub\CPS824-RL\cps824_assignment2>cd src

C:\Users\pabou\Documents\GitHub\CPS824-RL\cps824_assignment2\src>python mc_td_and_qlearning.py
Traceback (most recent call last):
  File "mc_td_and_qlearning.py", line 4, in <module>
    import gym
ModuleNotFoundError: No module named 'gym'

C:\Users\pabou\Documents\GitHub\CPS824-RL\cps824_assignment2\src>conda activate gym

(gym) C:\Users\pabou\Documents\GitHub\CPS824-RL\cps824_assignment2\src>python mc_td_and_qlearning.py

-------------------------
Beginning First-Visit Monte Carlo
-------------------------
The success rate of the policy across 500 episodes was 100.00 percent.
```

Fig. 1

1. D. The accuracy ranges anywhere from 5-30% at 1000 iterations. The reason the accuracy is so low is that the convergence for stochastic environments takes many more iterations. For the policy to be learned, it must overcome the randomness inherent in the environment. This requires orders of magnitude greater iterations. Figure 2 shows MC GLIE with 1000 iterations, and the characteristic low accuracy rate. Figure 3 shows MC GLIE with 100 000 iterations.

```
466     Q_mc, policy_mc = mc_glie(env, iterations=1000, gamma=0.9)
467     test_performance(env, policy_mc)
468     # render_single(env, policy_mc, 100) # uncomment to see a single
        episode
469
470     print("\n" + "-" * 25 + "\nBeginning Temporal-Difference\n" + "-"
        * 25)
471     #Q_td, policy_td = td_sarsa(env, iterations=1000, gamma=0.9,
        alpha=0.1)
472     test_performance(env, policy_td)
473     # render_single(env, policy_td, 100) # uncomment to see a single
        episode
474
475     print("\n" + "-" * 25 + "\nBeginning Q-Learning\n" + "-" * 25)
476     #Q_ql, policy_ql = qlearning(env, iterations=1000, gamma=0.9,
        alpha=0.1)
477     test_performance(env, policy_ql)
478     # render_single(env, policy_ql, 100) # uncomment to see a single
```

```
TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE                    1: cmd

Beginning First-Visit Monte Carlo
-------------------------
The success rate of the policy across 500 episodes was 0.00 percent.

-------------------------
Beginning Temporal-Difference
-------------------------
Traceback (most recent call last):
  File "mc_td_and_qlearning.py", line 472, in <module>
    test_performance(env, policy_td)
NameError: name 'policy_td' is not defined

(gym) C:\Users\pabou\Documents\GitHub\CPS824-RL\cps824_assignment2\src>python mc_td_and_qlearning.py

-------------------------
Beginning First-Visit Monte Carlo
-------------------------
The success rate of the policy across 500 episodes was 7.60 percent.
```

Figure 2.

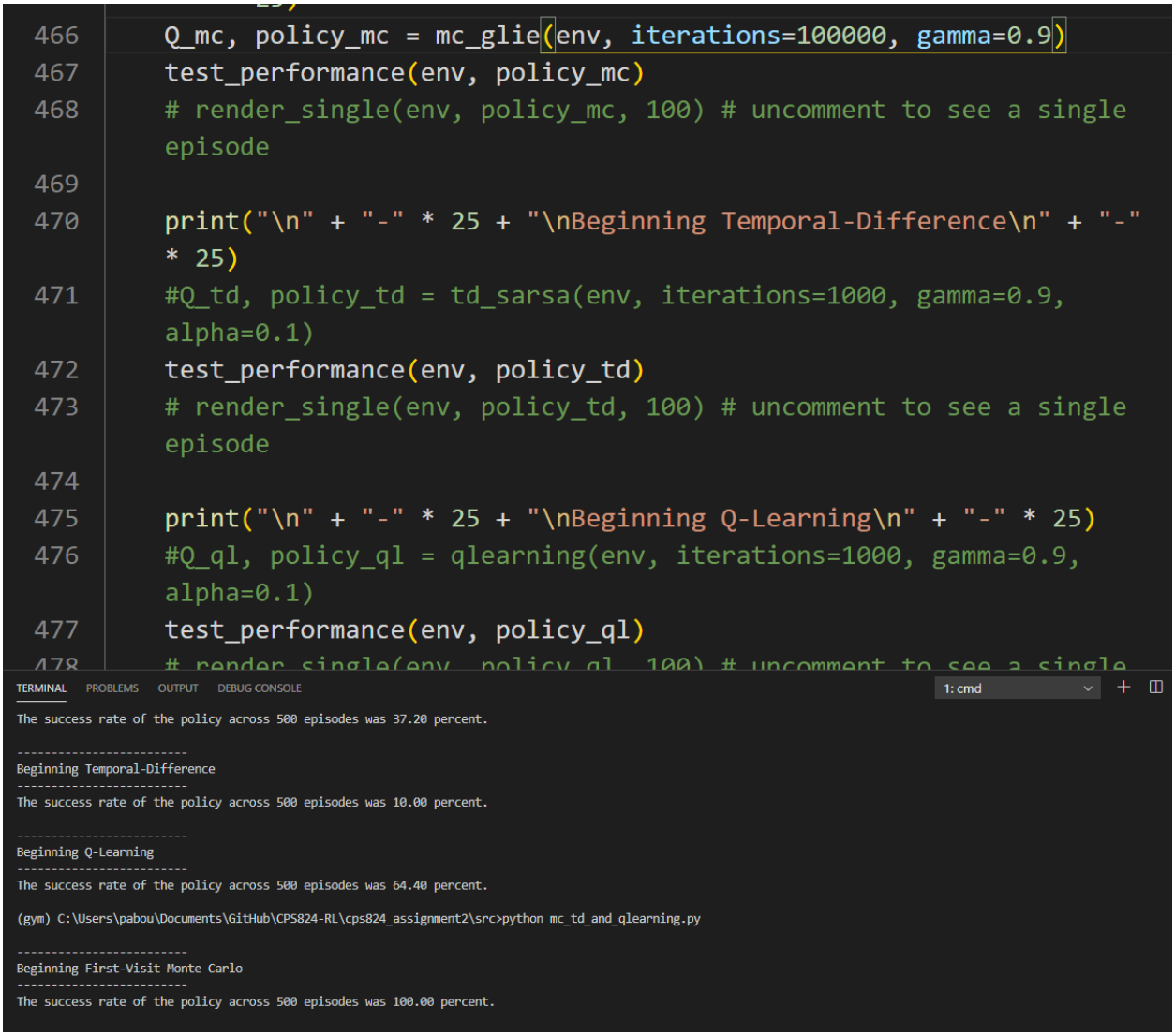


Figure 3.

2. A. I decided to conduct an experiment whereby each algorithm was run with increasing iteration values. The algorithms were run 10 times at each iteration interval and their results were averaged. Figure 4 demonstrates the performance increases for TD SARSA from the described experiment. As you can see the performance does not improve much, with the success rate hovering around 25-30%. This is largely due to the nature of the stochastic environment.

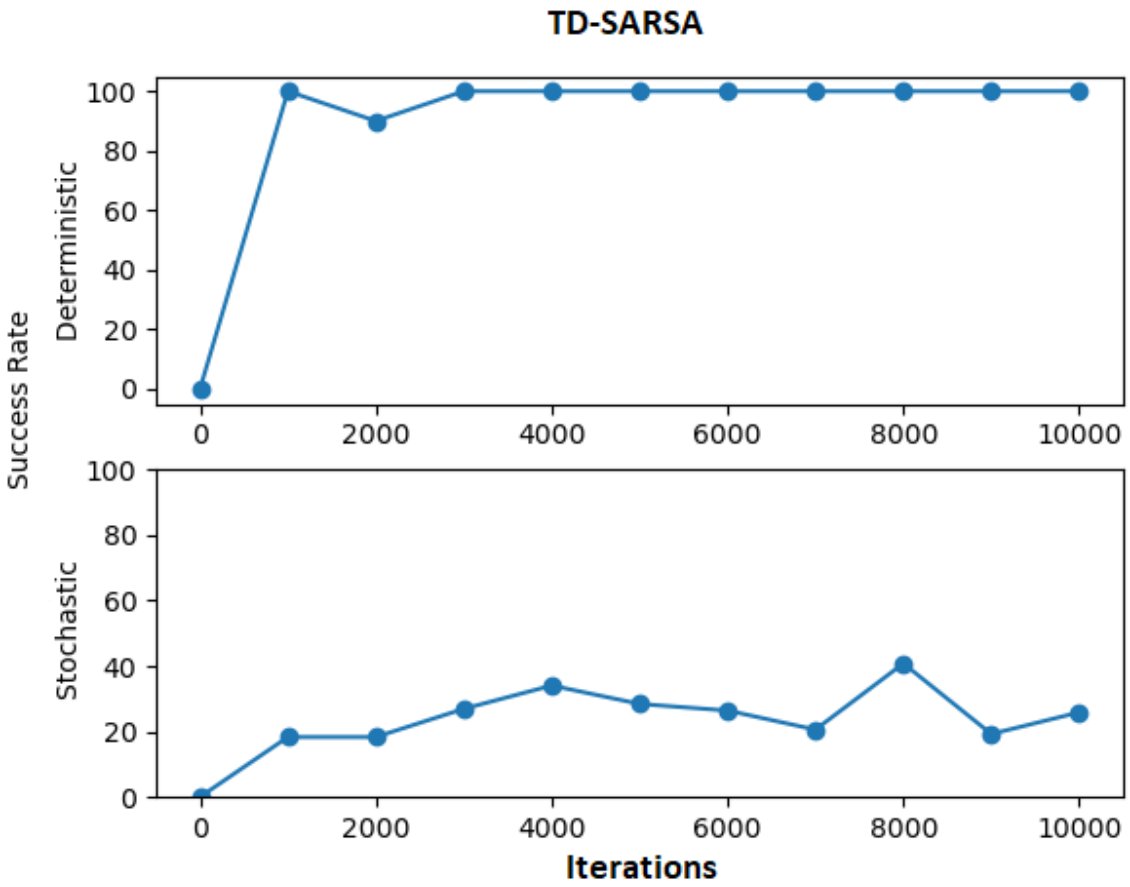
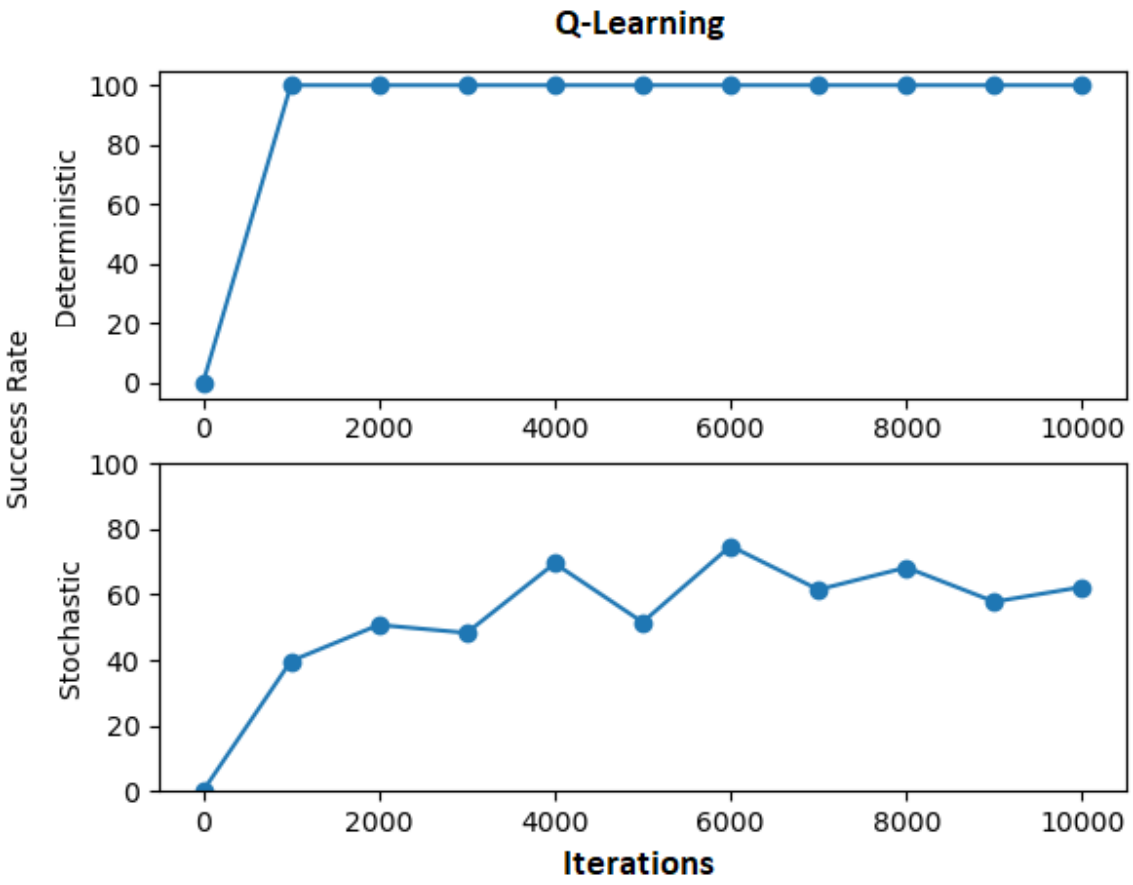


Fig. 4

2. B. I conducted the same experiment using Q-learning. The results are displayed in Figure 5. Q learning is vastly superior to TD SARSA. I suspect this is because of the nature of the environment FrozenLake. There are no large negative rewards, and therefore the 'riskier' algorithm (Qlearning) overall has better performance.


Q-Learning
Success Rate
Deterministic
Stochastic
Iterations
0
20
40
60
80
100
0
2000
4000
6000
8000
10000