FrozenLake-V3

Test Env.

- Python 3.5.2, OpenAI Gym; MacOS 11.2.1, 2.5G, 16GB

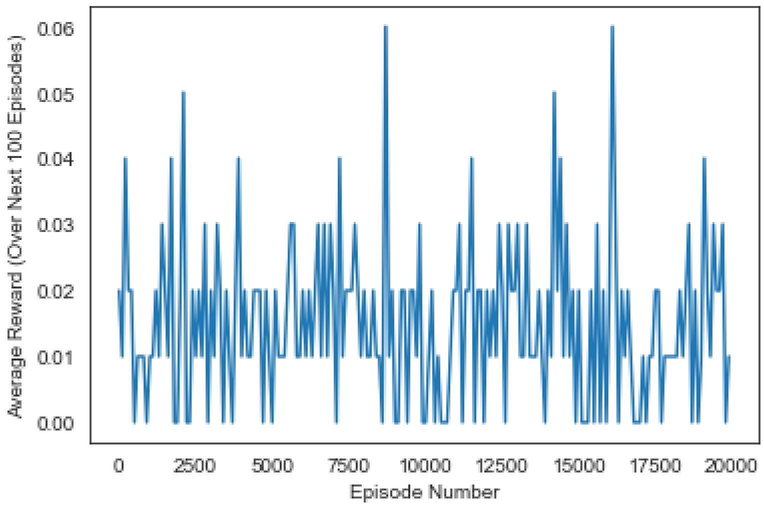
Implementation

- Q-learning(sarsamax) with exp. epsilon decay
  . Deterministic 4x4, 8x8 cases:
    - 20000 liters, gamma: 1.0, alpha:0.01, epsilon:1.0 with exp. decay

  . Stochastic 4x4: 100000 iters gamma:1.0, alpha:0.01, epsilon:1.0 with exp. decay
  . Stochastic 8x8; 100000 iters gamma:1.0, alpha:0.05, epsilon:1.0 with exp. decay

Results

1. **4x4 Deterministic State Transition**
   a. Best Average Reward over 100 Episodes: 0.06



   b. Estimated Optimal Policy (UP = 3, RIGHT = 2, DOWN = 1, LEFT = 0)

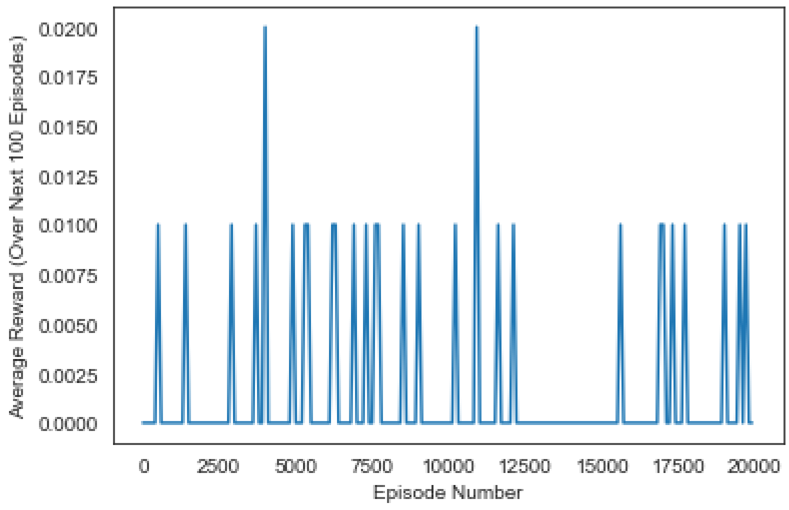
```
[[2 2 1 0]
 [3 0 1 0]
 [2 2 1 0]
 [0 2 2 0]]
```

c. Test Result

```
1. Checking Frozen_Lake
2. Q-learning
3. Testing after learning
4. Exit
select: 3
avg: 1.0
```

2. 8x8 Deterministic State Transition

a. Best Average Reward over 100 Episodes: 0.02



b. Estimated Optimal Policy (UP = 3, RIGHT = 2, DOWN = 1, LEFT = 0)

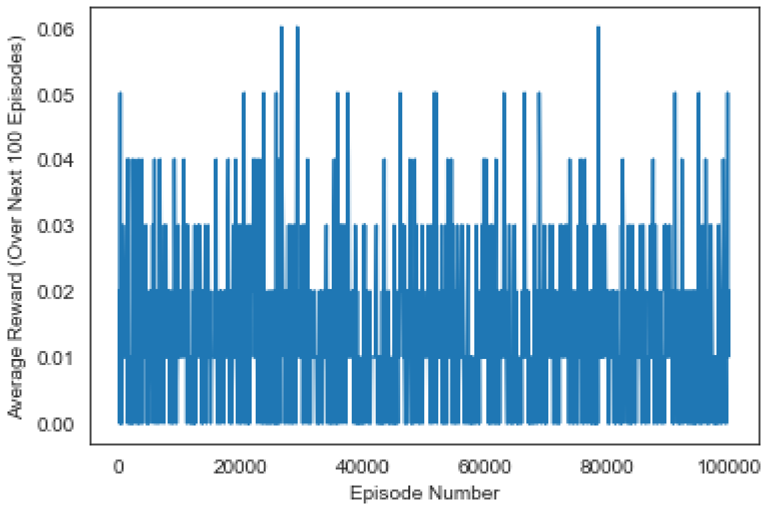
```
[[1 1 1 1 1 2 2 1]
 [2 2 2 2 2 2 2 1]
 [3 3 3 0 2 2 2 1]
 [3 3 3 0 3 0 2 1]
 [3 3 3 0 2 2 2 1]
 [3 0 0 2 3 3 0 1]
 [3 0 1 3 0 3 0 1]
 [3 0 0 0 0 3 0 0]]
```

c. Test Result

```
1. Checking Frozen_Lake
2. Q-learning
3. Testing after learning
4. Exit
select: 3
avg: 1.0
```

3. 4x4 Stochastic State Transition

a. Best Average Reward over 100 Episodes: 0.06



b. Estimated Optimal Policy (UP = 3, RIGHT = 2, DOWN = 1, LEFT = 0)

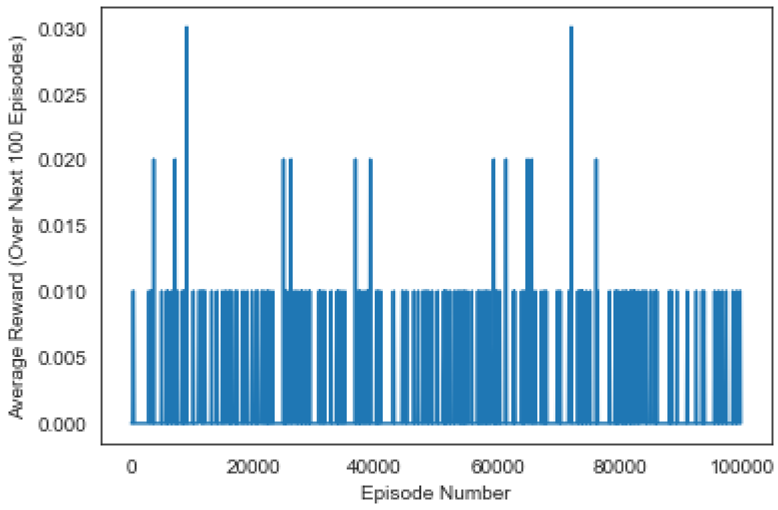
```
[[0 3 3 3]
 [0 0 0 0]
 [3 1 0 0]
 [0 2 1 0]]
```

c. Test Result

```
1. Checking Frozen_Lake
2. Q-learning
3. Testing after learning
4. Exit
select: 3
avg: 0.827
```

4. 8x8 Stochastic State Transition

a. Best Average Reward over 100 Episodes: 0.03



Estimated Optimal Policy (UP = 3, RIGHT = 2, DOWN = 1, LEFT = 0):

```
[[3 2 3 2 2 3 2 2]
 [3 3 3 3 3 3 3 2]
 [0 3 0 0 2 3 2 2]
 [0 0 0 3 0 0 2 2]
 [0 3 3 0 3 1 3 2]
 [0 0 0 2 2 0 0 2]
 [0 0 2 0 0 3 0 2]
 [0 1 0 0 1 1 2 0]]
```

b. Test Result

```
1. Checking Frozen_Lake
2. Q-learning
3. Testing after learning
4. Exit
select: 3
avg: 1.0
```