CS440 MP1 Han Jiang (hjiang25)

1. In WorldWithoutThief
(a) Set the discount factor to 0.9, the learning rate to 0.1, and the epsilon to 0.0 (which means your agent does not perform random actions except perhaps for ties). Simulate your agent for 1000 episodes of 10000 steps. Explain what you observe.

Ans: At the first 5 or so episodes, the robot got negative rewards, then all of the following episodes got 0 rewads.
Reason: Because there are a lot slippery blocks and the the robot got negative score at the beginning, so the robot learnt the world is dangerous and it's even better to stay in its own safe spot, being satisfied with zero score which it thought to be better than negative score.
Due to the epsilon is set to 0, the robot doesn't try to explore the way to overcome the hash situation any more.

(b) Set  as 0.1, and keep all the other settings. What did you observe? Explain it.

Ans:  During the first 90 episodes, the robot got mostly negative scores, but situation was getting better, the robot got more and more positive scores later on. At the end of 200th episode,  the robot tended to get a lot rewards that were higher than 20.
Even though the robot still occasionally got negative score, but the average is pretty good, at about 15, which is much better than with epsilon value set to be 0.

Reason: Though the robot got mostly negative reward at the beginning, the non-zero epsilon made it brave to explore the new way. Finally, the robot got some positive rewards due to seemly blunt tries, so the robot progressively learnt how to choose the action that yield best possible reward.

This story tells us that experiencing some adventure might make we smart enough to deal with hash situations, and the outcome may be much better than stay timidly in a "safe" corner.

(c) Try larger  (for example,  = 0.5) and keep all the other settings.
What did you observe? Explain it.

Ans:  After trying some larger epsilon values, I found the reward become bad again. If the epsilon were as large as 9, the robot almost always get negative rewards.

Reason: the large epsilon made the robot too blunt/brave, it didn't choose its action randomly rather than wisely. In other words, it doesn't take its experience seriously.

This story tells us to be brave but not too brave.

2. In WorldWithThief,
(a) Set epsilon = 0.05 and keep all the other settings. In the simulation command, set knows_thief as n. How is the performance of your agent? Explain it.

Ans: The robot's performance is poor, getting negative rewards in most episodes.
Reason: In this world, slippery is not as big problem as thief. This thief's position keep changing. If the robot only learnt from states whose information doesn't contains the thief's position, the price is high. Maybe the thief was at the position last time may not be at the same position this time.

(b) Set knows_thief as y. Explain any performance difference.
Ans: At the first 60 episodes, the robot got a lot negative rewards, but the rewards gets better over time. After 200 episodes, the robot seemed to be intelligent enough to get averagely good rewards at about 25.
Difference is due to taking the thief's position into account when choosing next actions. The thief's position is one of the factor defining the state. It allows the robot to choose the right action according to the specific state with specific robot position.

(c) Search for the best learning rate and epsilon. Describe your investigation and conclusions.
Ans:
My conclusion is:
The Bests are Rate: 0.2 Epsilon: 0.01
I added code in Agent interface and sub-classes to allow change epsilon and rate, so I can set rate and epsilon for the agent.

```
public  void setRate(double rateIn);
public void setEpsilon(double epsilonIn);
```

In simulator, I wrote the following code to find the rate and epsilon that generate the largest final value.

```
public void simulateAll() {
        ArrayList<ArrayList<Double>> allTotalScoreList = new ArrayList<ArrayList<Double>>();
        for (double rate = 0.1; rate < 0.99; rate += 0.1)
                for (double epsilon = 0.01; epsilon < 0.2; epsilon += 0.01) {
                        eachSimulate(rate, epsilon, allTotalScoreList);
                }
        long max = Long.MIN_VALUE;
        for (ArrayList<Double> each : allTotalScoreList) {
                long score = Math.round(each.get(0));
                if (score > max) {
                        max = score;
                        System.out.print("Rate: " + each.get(1));
                        System.out.print("  Epsilon: " + each.get(2));
                        System.out.println("Reward: " + max);
                }
```

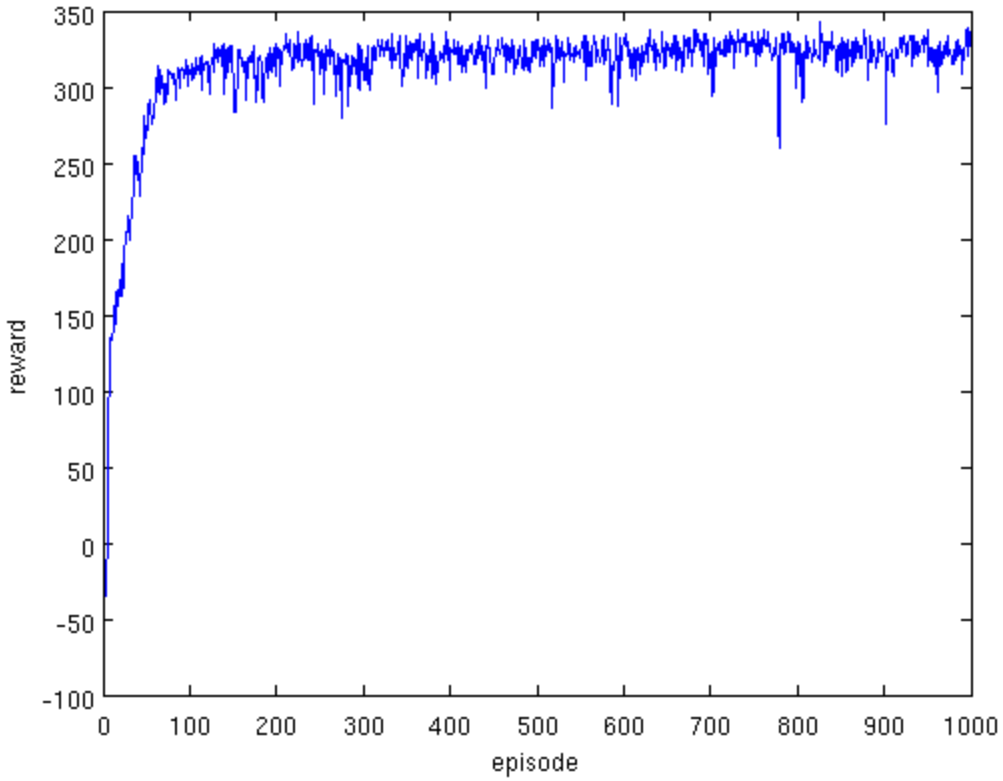
```
            }
    }
```

My conclusion is:

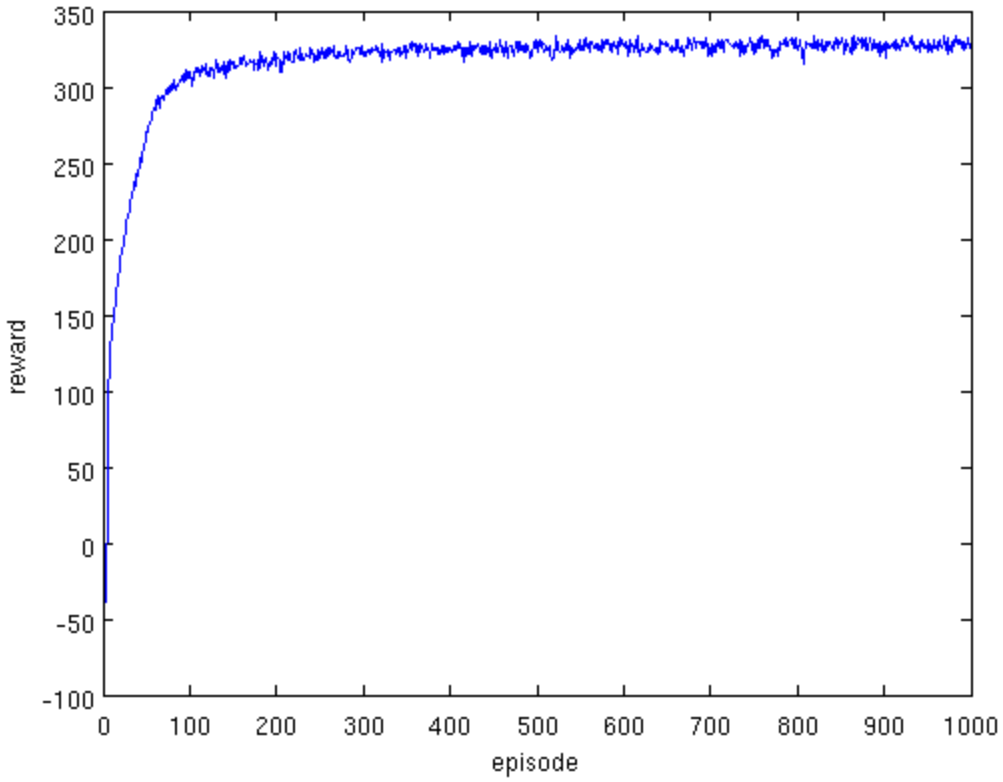
Rate: 0.2  Epsilon: 0.01

3. Using your best parameters from 2c, simulate your agent for 1000 episodes of 10000 steps. Plot the expected discounted reward achieved at the end of each episode showing how the agent improves. What do you notice? Repeat the procedure nine more times and again plot the expected discounted reward at the end of each episode but now averaged over all ten simulations. Explain what you observe.

Ans:




The reward grows quickly at the beginning, then reach a relatively stable level at about 330. However, there is a obvious fluctuation of values.




In MatLab, I import all data from 10 tries as data array and did following computation and plot.

```
>> total = try1+try2+try3+try4+try5+try6+try7+try8+try9+try10
>> ave=total/10
>> plot(ave,'DisplayName','ave','YDataSource','ave');figure(gcf)
>> ylabel('reward')
>> xlabel('episode')
```

Now we can see the plot is much more smoother and clear. The plot clearly shows the reward grow quickly at the beginning and then reaches stable at a level of about 325 after about 220 episodes.