CS440/ECE 448: Introduction to AI
MP1 Part 2
Martin McEnroe, mcenroe2@illinois.edu
September 29, 2014

Problem 1. WorldWithoutThief

1a. Discount factor 0.9, learning rate 0.1 epsilon = 0.0

```
Episode 1: reward = -2.5
Episode 2: reward = -4.0
Episode 3: reward = -2.0
Episode 4: reward = -2.0
Episode 5: reward = 0.0
```

Every episode after 5 has a reward of 0. The agent quickly experiences package loss when it encounters slippery areas. There is no path that doesn't cross a slippery square and so the agent learns to avoid them. There is no mechanism for the agent to explore past the penalties to find out that there are rewards for delivering packages to customers. So the agent learns to do nothing and by doing so avoids penalty. From the agent's perspective it has maximized the possible reward. It never had a chance to learn about anything better.

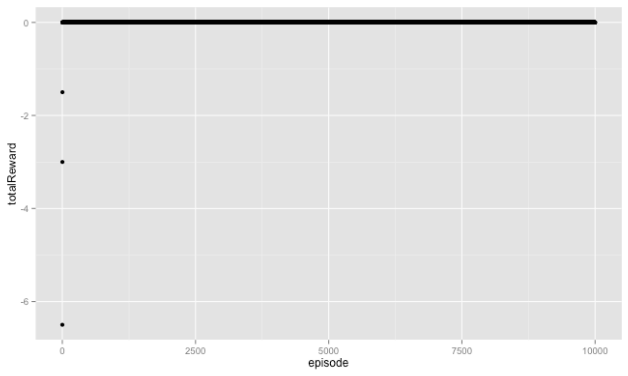


Figure 1a - Epsilon = 0.0

1b. Change epsilon to 0.1

Here the agent is given a chance to explore. In this diagram, after about 1200 iterations the agent develops a policy map that has it move aggressively. Unfortunately due to the nature of the randomness of the slippery zones the agents performance is uneven from episode to episode, however the average reward is clearly higher upon visual inspection.

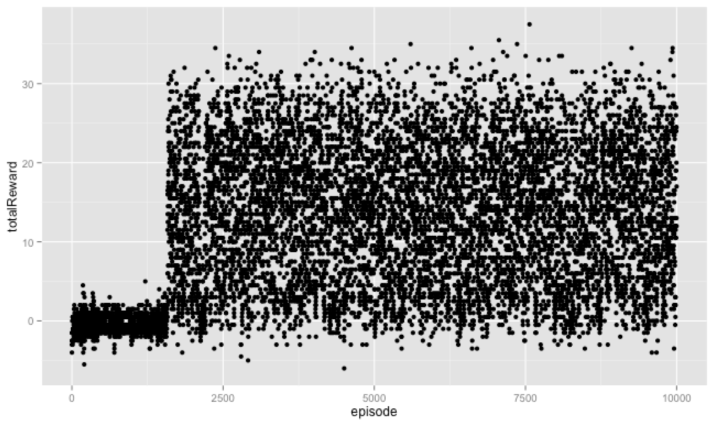


Figure 1b. Epsilon = 0.1

1c. Change epsilon to a larger value, for example 0.5

In this case random decisions are made half the time and the result is that randomness dominates the agent's overall behavior.

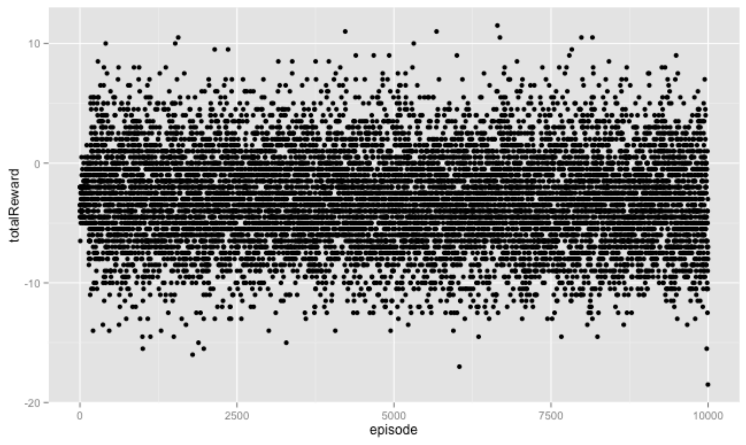


Figure 1c1 - Epsilon = 0.5. Plot of rewards vs. episodes

Curiously this looks almost Gaussian. What are the statistics:

```
Min.    :-18.500
1st Qu.: -5.500
Median : -3.000
Mean   : -3.005
3rd Qu.: -0.500
Max.   : 13.000
```

Let's do a simple histogram. Using R:

```
hist(episodes$totalReward,breaks=100)
```

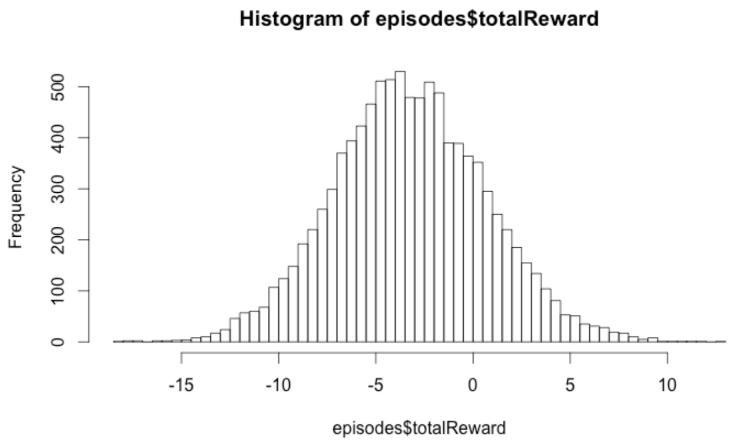


Figure 1c2- Epsilon = 0.5. Histogram of reward outcome frequency in bins =100

Histograms have a notable weakness that the shape is dependent on the size of the bins. It is better to apply the kernel density function. Using R:

```
b<-density(episodes$totalReward)
plot(b)
```

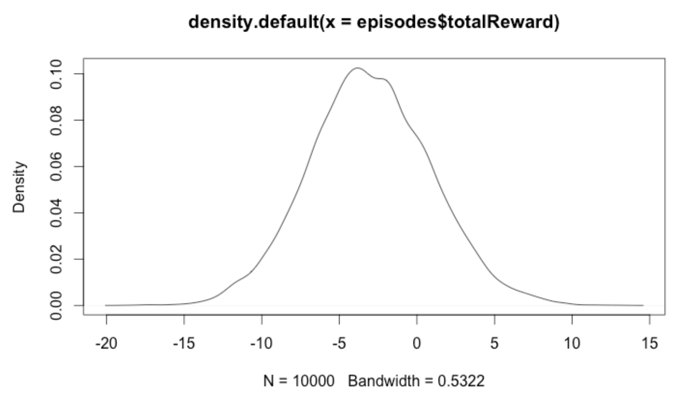


Figure 1c3- Epsilon = 0.5. Kernel density plot of total reward frequency

From the two graphs it is reasonable to infer that we roughly have a random agent.

Problem 2. WorldWithThief

2a. Set epsilon = 0.05. Set knows_thief = 0.

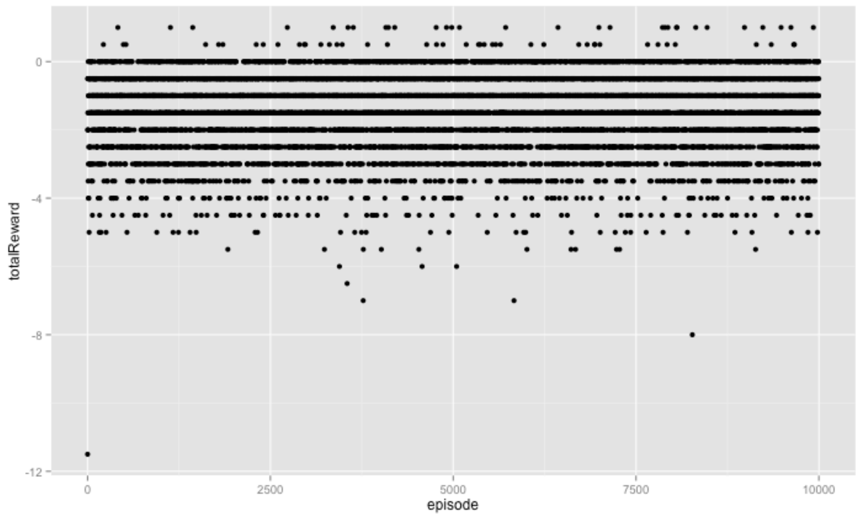


Figure 2a1- Agent behavior over time when agent does not know about thief

The agent displays distinct reward levels and does not appear to improve. Plotting the frequency of each reward shows that the when the agent is not aware there is a thief, it cannot learn and thus adapt behavior to avoid the penalty. Nearly all rewards are zero or negative.

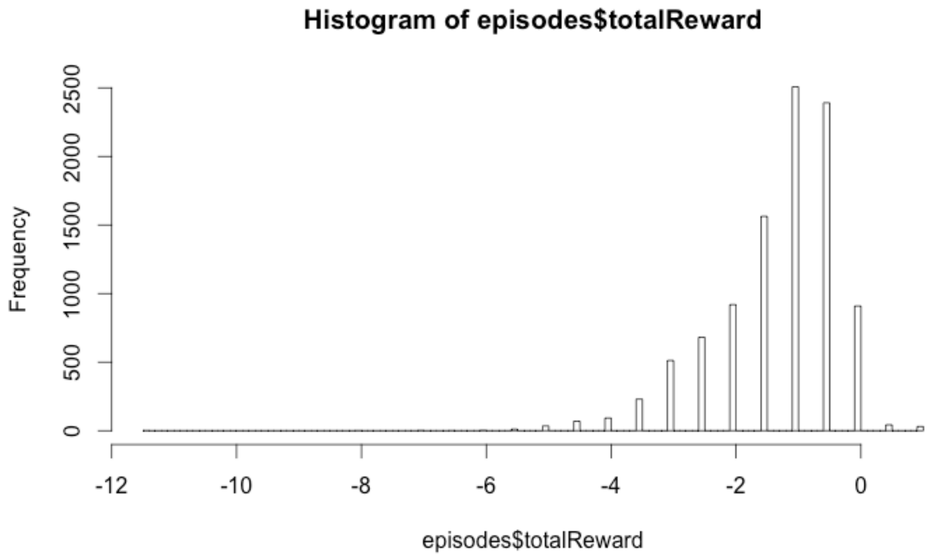


Figure 2a2- Frequency of each reward when agent does not know about the thief

2b. Keep all other settings. Set knows_thief = 1.

Now the agent displays some remarkable behavior. After a brief period of rapid improvement the agent consistently performs in the 15 to 35 point reward zone.

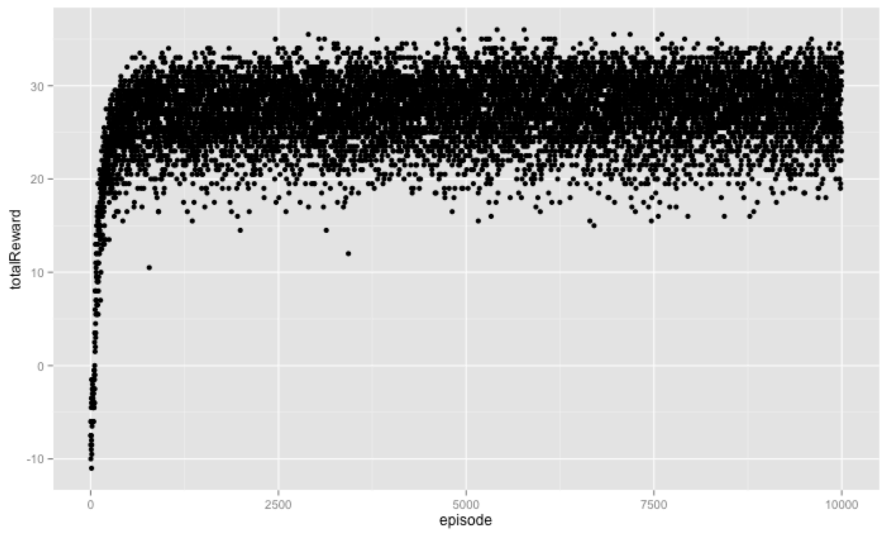


Figure 2b- Frequency of each reward when agent DOES know about the thief

Setting a baseline on statistics:

```
      > summary(episodes$totalReward)
         Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
        -11.0    25.0    27.5    27.1    30.0    36.0
```

What if we eliminate the ramp up?

```
      > summary(episodes[800:10000,]$totalReward)
         Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
        12.00   25.50   28.00   27.58   30.00   36.00
```

Later experimentations will have much more gradual ramps so for the purposes of comparison we can always cut out the first half of the episodes with only minor effect to our baseline:

```
> summary(episodes[5000:10000,]$totalReward)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   15.0    25.5    28.0    27.8    30.5    36.0
```

2c. Search for the best learning rate and epsilon.

In order to quickly search I ran the simulator three times with a configuration and I chose the best median of the three for the latter half of the episodes and used that to determine which direction to search. Here is a record of my explorations:

alpha = 0.1

epsilon	best median of three
0.1	21.5
0.15	15.5
0.11	20.5
0.09	22.5
0.08	24.0
0.07	25.5
0.06	28.0
0.05	28.0
0.04	29.0
0.03	30.5
0.02	31.5
0.01	33.0
0.005	34.0
0.001	25.0

I then switched to calculating the mean of three tries around the point 0.005

0.008	33.27694
0.007	**33.54752**
0.006	33.1803
0.005	33.21042
0.004	32.57225

So I feel relatively confident that epsilon = 0.007 is near the maximum for epsilon. Now to test the learning rate with epsilon = 0.007

alpha	mean of three tries
0.08	33.09138
0.09	32.74535
1.0	**33.54752 (from before, above)**
0.11	33.37169
0.12	33.60931
0.13	32.31507

let's retry the 0.12

0.12	33.38082

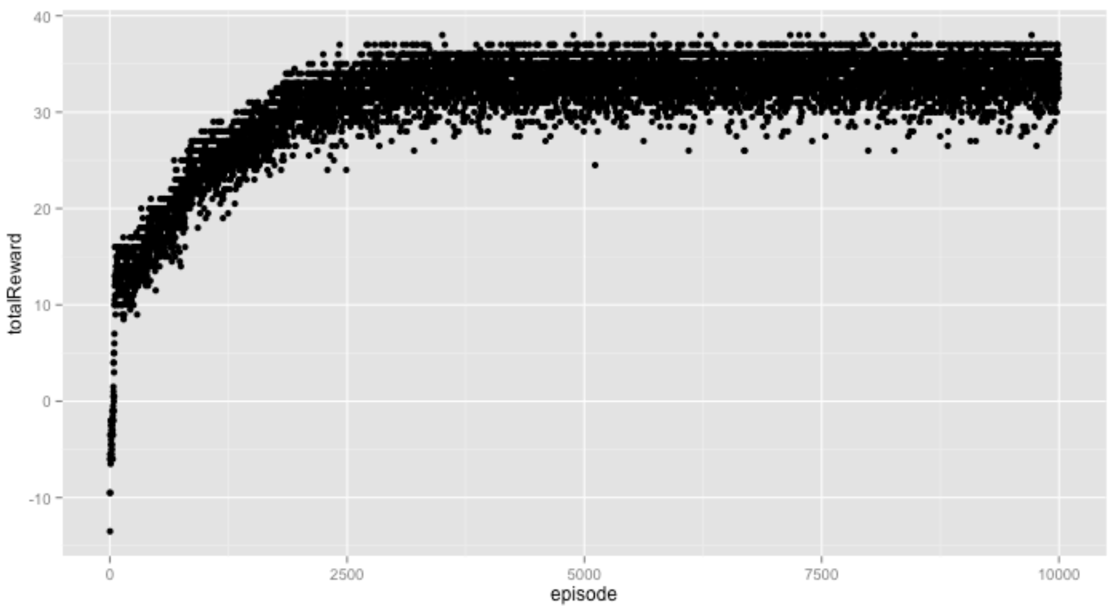
The problem is that there is some inherent variability to the simulation that masks the true variation of the parameters. Perhaps trying a hundred simulations and computing the average might yield a more definitive or a higher confidence but based on the simulation runs I did I found

alpha = 0.1 and epsilon = 0.007 to be the best combination.

3. Plot of best parameters

alpha = 0.1 and epsilon = 0.007

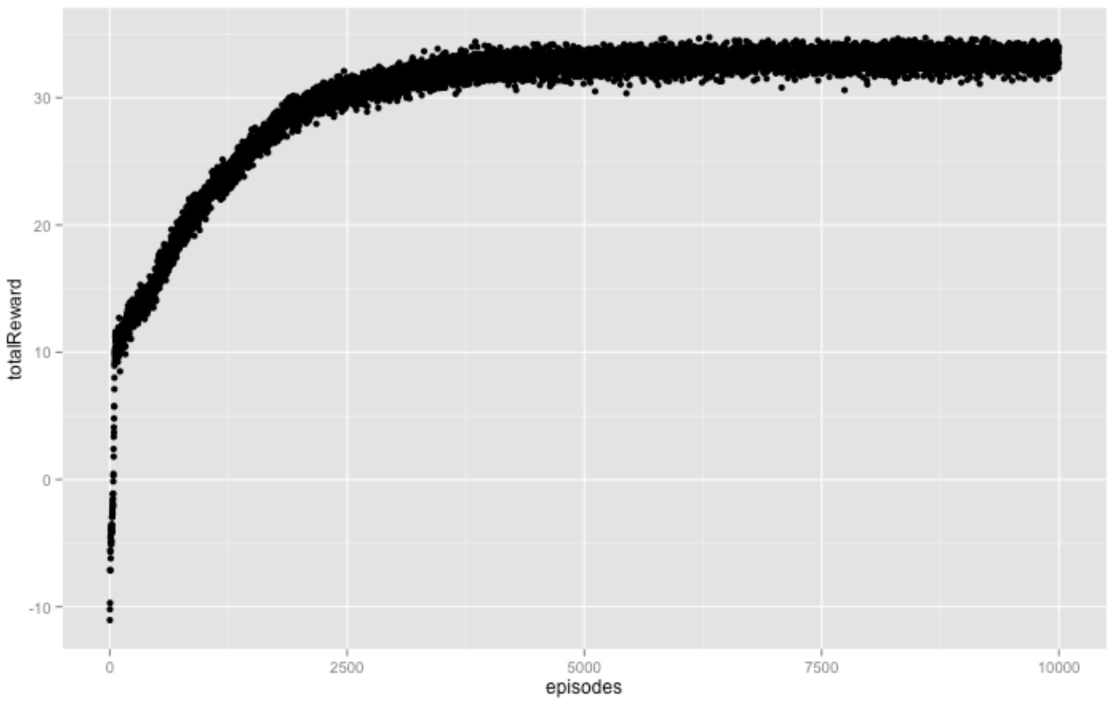
```
> summary(episodes[5000:10000,]$totalReward)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   24.5    33.0    34.0    33.8    35.0    38.0
```



After a very brief learning period the agent quickly becomes productive and continues to climb with tight clusters (small difference between first quartile and third quartile) that continue as the agent levels off in performance.

Repeating the procedure nine more times but taking the average of each episode and then plotting the average point. The clusters have tightened.

```
> summary(pme[5000:10000,]$totalReward)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  30.35   32.65   33.05   33.05   33.45   34.75
```


totalReward
30
20
10
0
-10
0
2500
5000
7500
10000
episodes