ASSIGNMENT #5 (Model-Based)

Alireza Mostafizi

I coded this assignment in Python 2.7. Attached to the email you can find two python codes. One is the MDP maker for the parking lot domain (MDP Maker.py) which you can use to create a MDP, using the probabilities and rewards, to feed the **Optimistic Exploration** RL algorithm with. The other one is the code for Part III that simulate the Model-based (Optimistic Exploration) learning environment (HW5.py). I also included calculating optimal policy with value iteration in this part of code to see how the final learned policy differ from the optimal one. Please not that to run the code you have to have "NumPy" installed.

Part I is copied from Assignment 4 for your convenience. Part II is removed for the sake of this report. However, its code is used for policy evaluation purposes in part III.

Part I: Designing the MDPs

Here is a brief description of the MDP for the parking lot domain which I mostly copied from HW2, except some minor changes:

States

- If we assume n parking spots for each row, the MDP has $8n + 2$ states. Like suggested in the assignment, I considered $8n$ states, plus a "FINAL" state and a "NO" state.
  - The "FINAL" state you can get to only if the car is parked or had an accident, and it acts as a terminal state. It has +0.01 reward, and once you get there, you will remain there.
  - The "NO" state is the state where you get to when you make an inappropriate decision. For example, in case the car is parked and the "DRIVE" decision is made, you will get to "NO" state which has -0.01 reward, and once you get there, you will remain there.

To depict how the states are mapped to the numbers, let's assume $n = 5$:

	State Numbers																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Cell	A5				A4				A3				A2				A1			
Occupancy	NO	NO	O	O	NO	NO	O	O	NO	NO	O	O	NO	NO	O	O	NO	NO	O	O
Parked	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P

	State Numbers																			
	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
Cell	B1				B2				B3				B4				B5			
Occupancy	NO	NO	O	O	NO	NO	O	O	NO	NO	O	O	NO	NO	O	O	NO	NO	O	O
Parked	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P

NO = NOT OCCUPIED O = OCCUPIED NP= NOT PARKED P = PARKED

State 41 is the FINAL state, and state 42 is the NO state.

Occupancy Probability

There are some simple assumptions I made to prepare the MDP.

- The probability of ADA spot being empty is entered in the program.
- The closer the spot is to the store, the lower the probability of the spot being empty.
    - Therefore, the program asks for the probability of closest spot (A_2 and B_2) being empty (AKA Min probability) and probability of the farthest spot (A_n and B_n) being empty (AKA Max Probability)
    - I assumed that the probability of other spots not being occupied linearly decrease from max to min probability, based on their distance to the store.

Rewards

Here are assumption about the rewards. Most of them are suggested by the assignment.

- There is a fine for parking in ADA spot, which you can enter in program
- There is a huge cost for accident, which you can enter in program
- There is a relatively small cost for driving from each cell to the next one, which you can enter in the program
- You can enter the reward of parking in the closest spot (A_2 and B_2), known as Max reward, and parking in the farthest spot (A_n and B_n), known as Min reward.
    - The other rewards are drawn from linear interpolation between max reward and min reward, based on their distance to the store.

Transition Matrices

As suggested by the assignment, I considered three actions:

- 1. Drive to the next spot
    - ACTION 1 in the program
    - Cannot be taken when the car is parked
    - Will take you to the next cell considering the circulation pattern
        - Note that there are 4 states for each cell. This action will take you to the next cell's state with NO PARKED condition. Ending up in OCCUPIED or NOT OCCUPIED state is based on the probability of occupancy.
- 2. Park
    - ACTION 2 in the program
    - Cannot be taken when the car is parked
    - Will take you from NOT PARKED state in each cell to the PARKED state in the same cell.
        - The probability of ending up in OCCUPIED or NOT OCCUPIED condition is based on the probability of occupancy.
- 3. Exit
    - ACTION 3 in the program
    - Cannot be taken when the car is not parked
    - And If the car is parked (or had an accident), the next action will definitely be the exit

Procedure

I created the general-purpose simulator in which you can simulate a learning agent. Basically, what you have to do is to create the corresponding MDP for the parking lot through "MDP Maker.py," using the parameters and rewards of interest. Then you can run the algorithm with mentioned MDP to see how the learning agent comes up with the optimal policy. To compare the policy to the optimal one, I included the planner from HW2 which calculates the optimal policy using value iteration. To test the results, I considered two MDPs as following:

NOTE: MDPs' probabilities and rewards are the same as the ones I used for HW2

First MDP Description

```
N (>3): 10
COST of Driving (+): 5
COST of Accident (+): 800
FINE for ADA Spot (+): 200
Probability of ADA spot NOT being occupied (0-1): 0.05
Reward for the closest spots (NOT THE ADA) (+): 100
Reward for the farthest spots (+): 10
Probability of the closest sport NOT being occupied (0-1): 0.05
Probability of the farthest spot NOT being occupied (0-1): 0.85
Output File Name: Output1
MDP input generated: Output1.txt
```

Second MDP Description

```
N (>3): 10
COST of Driving (+): 5
COST of Accident (+): 800
FINE for ADA Spot (+): 200
Probability of ADA spot NOT being occupied (0-1): 0.05
Reward for the closest spots (NOT THE ADA) (+): 30
Reward for the farthest spots (+): 30
Probability of the closest sport NOT being occupied (0-1): 0.5
Probability of the farthest spot NOT being occupied (0-1): 0.5
Output File Name: Output2
MDP input generated: Output2.txt
```

As you can see the difference between these two parking lot MDPs is that in the first one, closer spots have higher rewards and lower probabilities to be empty. However, in the second one, all the spots have the same reward and the same probabilities. Other than that, fines for accident and parking in ADA spot and cost of driving is the same. Thus, we should expect that in the second scenario, optimal policy is to park wherever is empty. On the other hand, in the first one, agent should learn how to balance the cost of driving and the reward of the parking.

The following Tables show the optimal policy through value iteration algorithm for these two MDPs.

First MDP Optimal Policy

A10				A9				A8				A7				A6				A5				A4				A3				A2				A1			
NO	NO	O	O	NO	NO	O	O	NO	NO	O	O	NO	NO	O	O	NO	NO	O	O	NO	NO	O	O	NO	NO	O	O	NO	NO	O	O	NO	NO	O	O	NO	NO	O	O
NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P
1	3	1	3	1	3	1	3	1	3	1	3	2	3	1	3	2	3	1	3	2	3	1	3	2	3	1	3	2	3	1	3	2	3	1	3	1	3	1	3

B1				B2				B3				B4				B5				B6				B7				B8				B9				B10			
NO	NO	O	O	NO	NO	O	O	NO	NO	O	O	NO	NO	O	O	NO	NO	O	O	NO	NO	O	O	NO	NO	O	O	NO	NO	O	O	NO	NO	O	O	NO	NO	O	O
NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P
1	3	1	3	2	3	1	3	2	3	1	3	2	3	1	3	2	3	1	3	2	3	1	3	2	3	1	3	2	3	1	3	2	3	1	3	1	3	1	3

Second MDP Optimal Policy

A10				A9				A8				A7				A6				A5				A4				A3				A2				A1			
NO	NO	O	O	NO	NO	O	O	NO	NO	O	O	NO	NO	O	O	NO	NO	O	O	NO	NO	O	O	NO	NO	O	O	NO	NO	O	O	NO	NO	O	O	NO	NO	O	O
NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P
2	3	1	3	2	3	1	3	2	3	1	3	2	3	1	3	2	3	1	3	2	3	1	3	2	3	1	3	2	3	1	3	2	3	1	3	1	3	1	3

B1				B2				B3				B4				B5				B6				B7				B8				B9				B10			
NO	NO	O	O	NO	NO	O	O	NO	NO	O	O	NO	NO	O	O	NO	NO	O	O	NO	NO	O	O	NO	NO	O	O	NO	NO	O	O	NO	NO	O	O	NO	NO	O	O
NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	P
1	3	1	3	2	3	1	3	2	3	1	3	2	3	1	3	2	3	1	3	2	3	1	3	2	3	1	3	2	3	1	3	2	3	1	3	2	3	1	3

NO = NOT OCCUPIED O = OCCUPIED NP= NOT PARKED P = PARKED

Actions: 1-Drive 2-Park 3-Exit

Optimal Policies make sense because:

- As it is shown in the above tables, optimal policy for these two MDPs are different.
- As expected, the second optimal policy is to park wherever you can. Since the probabilities and the rewards for all the sports (excluding the ADA spots) are the same.
- However, in the first one, it is not optimal to park on A10, A9, A8, and B10 since they are too far from the store and they have low rewards.

End of Copy

Part III: Reinforcement Learning

As I mentioned before, I implemented a model-based (Optimistic Exploration) reinforcement learning agent. The procedure is as following:

1. The initial Transition Matrices and Rewards (Initial Model) is all zero.
    - Basically, the initial policy would be to drive in every state

2. The greedy policy is calculated based on optimistic value iteration.
    - N_e is set to 5
    - V_{max} is set to 1000

3. In each trial, the agent chooses its set of actions (20 actions which typically ends up in terminal state) based on the optimistic greedy policy calculated in step 2, and also regarding the exploration exploitation strategy as following:
    - $\epsilon - Greedy$ algorithm
      Just like GLIE, ϵ decreases as the number of trials goes higher, but in a linear pattern instead of $1/x$. Based on my observations, in this domain, the strategy I used converges to the optimal policy slightly faster.

4. Update the transition matrices and the rewards.
    - After each 100 trials, find the optimal policy (NOT optimistic) based on the learned transition matrices and the rewards, and report the value of the policy. So N is set to 100 in this assignment.

5. Go to state 2
    - Repeat this process for 40 set of trials (40*100 in total)
    -

Explore/Exploit Strategy:

- Epsilon-Greedy:

  Using constant Epsilon-Greedy strategy was not very functional in this domain. It did not converge to the near-optimal policy even after 10k iterations.

- Drive-More:
  I tried the strategy I proposed in the previous assignment, and it hardly converged to near-optimal policy. Here is a description:
    - In each state, agents are willing to drive to the next spot with probability of 0.7.
    - Otherwise, they will choose the action that they have tried the least before.

- **GLIE:**
  This strategy worked the best, but I slightly modified it to descending linear pattern as mentioned. Since the last strategy works the best, I just included the results of that. So all the following learning curves are based on this exploring strategy.

Notes

- For all the purposes in this assignment, β is considered to be 0.99.
- In general, the implementation of the model-based learning environment was much harder than implementation of model-free reinforcement learning, and in terms of computation, Q-learning is faster than model-based reinforcement learning. (It takes roughly 30 minutes to perform 1k trials the way I coded this, which does not seem to be efficient)
- However, as shown in the learning curves, it seems that model-based approach with GLIE converges to the optimal policy with less trials than model-based does (with the proposed exploration exploitation strategy proposed in the previous assignment). Also the learning curves are much smoother and it is not noisy.
- I also tried ADP, but it hardly converged to the near-optimal policy since this particular domain needs specific strategies to be explored thoroughly.

Optimal learned policies are very close to the real optimal policies presented in the Part I.

So I ignored to double mention them here.

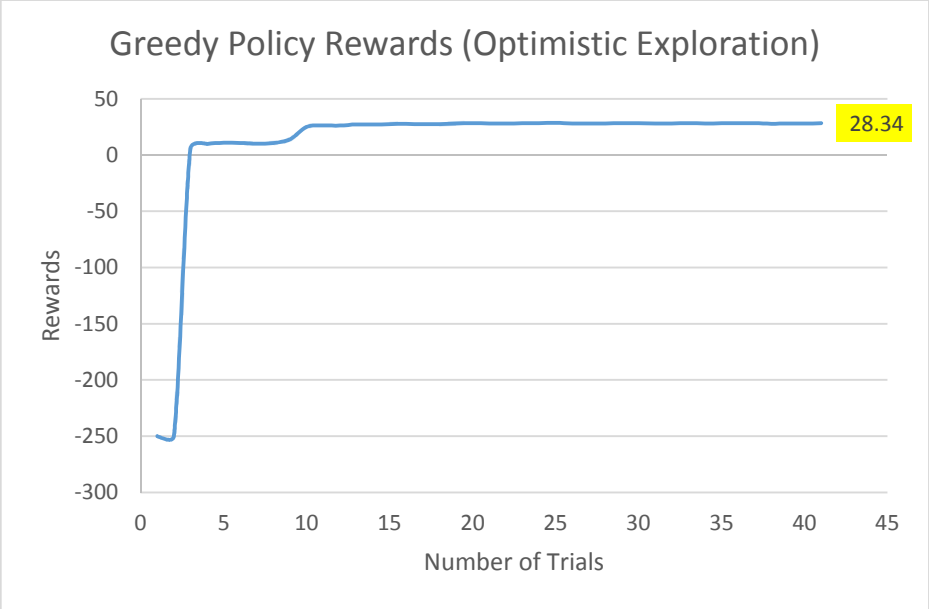
The following pages include the learning curves for two different mentioned MDPs (in Part I). I kept some of the learning curves of q-learning (from previous assignment) for comparison purposes.

Since the value of policy is calculated with 10000 trials (rather than 1000 trials in the previous assignment), model-based learning curves are less noisy rather than the ones in previous assignment.

As you can see in the learning curves, model-based converges faster (in less trials) to the near-optimal policy, but the very first policies are not good at all. On the other hand, model-free gives a “good” policy sooner than model-based, but it converges to the near-optimal policy relatively later than model-based.

MDP 1

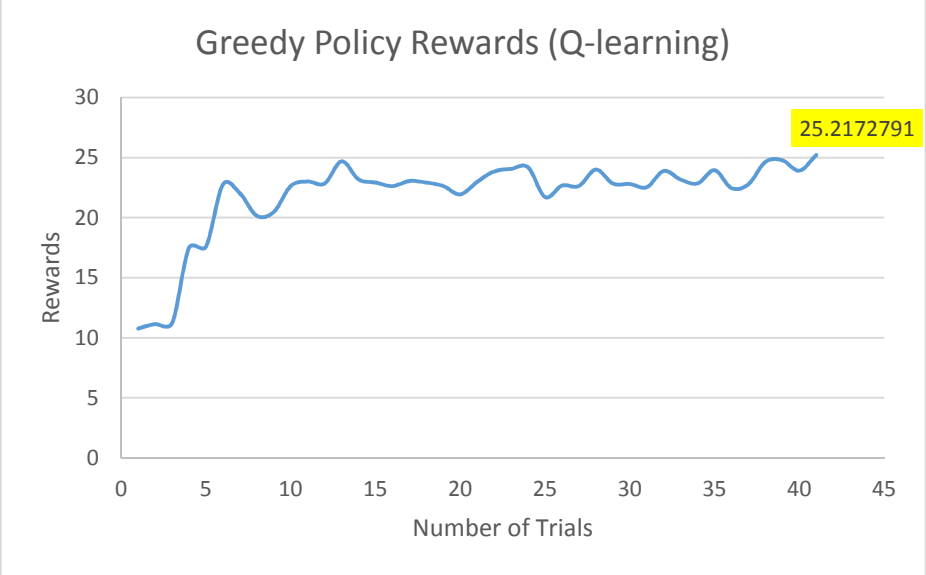
Model-Based: Optimistic Exploration



N = 100

- It converged to the value of optimal policy (28.4)
- It is less noisy due to the nature of model-based RL and also greater number of trials for policy evaluation
- It has converged in less trials than model-free below

Model-Free: Q-learning



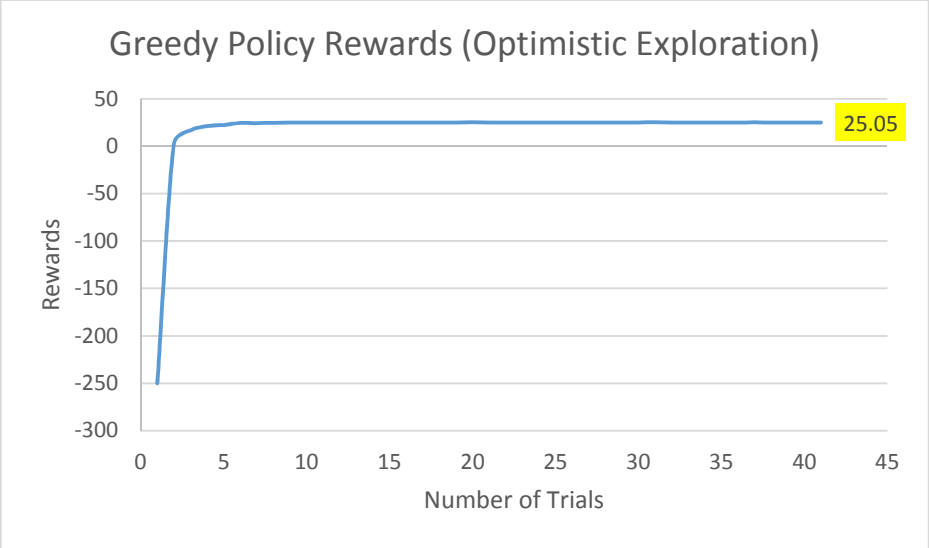
N = 100

$\alpha = 0.1$

- It fairly has converged to the optimal policy value. But not as good as model-based above. (28.4)
- It has converged later than the above

MDP 2

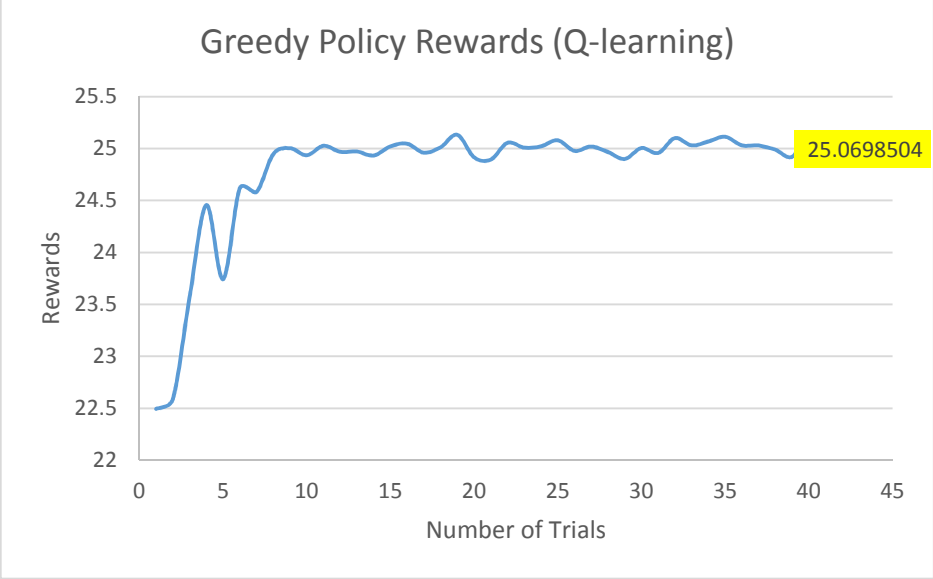
Model-Based: Optimistic Exploration



N=100

- It is much smoother than the one below.
- It has converged faster. (in less trials)
- It has converged to the optimal policy value. (25.05)

Model-Free: Q-learning



N = 100

$\alpha = 0.1$

- It's a bit noisy
- It has converged to the optimal policy value finally. (25.05)
- But later than the one above