CS 533 Homework 3
Optimizing Infinite-Horizon Discounted Reward with Application to Optimal Parking

Eugene Seo

04-26-2017

Part I: Build a Planner

I implemented **value iteration** algorithm for optimizing expected infinite-horizon discounted cumulative reward.

- INPUTs:
  1. A text file describing MDP (e.g. MDP1-hw3.txt, MDParking1.txt, ...)
  2. A discount factor, β ($0 \leq \beta < 1$)

- OUTPUT: A text file named "*output_discountFactor_MDPname*" containing following:
  1. Optimal value function (V), n-dimensional vector
  2. Policy, n-dimensional vector

- Run file: python infinite.py MDPFile discount-factor (e.g. python infinite.py MDP1-hw3.txt 0.9)

- Algorithm of Value Iteration for Infinite-Horizon Policy Optimization

  I used the same bellman-backup function previously implemented in hw2. The main differences in infinite-horizon policy optimization algorithm from the finite-horizon policy optimization are
  a) applying a discount factor, β, to control the future rewards
  b) running k iterations until the value function is converged
  c) computing the value of greedy policy using policy evaluation via matrix inverse

Algorithm 1 Value Iteration Algorithm for Infinite-Horizon Policy Optimization

```
# Compute value of the greedy policy
def policy_evaluation(state_size, T, R, P, df):
        I = np.identity(state_size)
        return np.dot(inv(I-np.dot(df,T)),R)

# Value Iteraction
def policy_optimization(state_size, action_size, T, R, df):
    V = np.zeros(state_size)
    policy = np.zeros(state_size)
    T_opt = np.zeros((state_size,state_size)) # transition from an optimal policy

    V_pre = 100000
    V[:] = R # initialize to R(s)

    # compute stopping condition
    expected_diff = 0.01
    stop_point = (expected_diff * np.square(1-df)) / (2 * np.square(df))

    # 1. Valute Interaction
    while ( max(abs(V_pre-V)) > stop_point ): # until convergence
        V_pre = deepcopy(V)
        for s in range(0, state_size):
            # 2. Greedy Policy
            max_value, action = bellman_backup(state_size, action_size, T, s, V)
            V[s] = R[s] + df * max_value # discounted expected value
            policy[s] = action
            T_opt[s,:] = T[action-1,s,:] # for matrix computation

    # 3. Policy Evaluation
    V_opt = policy_evaluation(state_size, T_opt, R, policy, df) # for infinite

    return V_opt, policy
```

Part 2: Run the Planner

Run my code on two given MDPs (MDP1-hw3.txt and MDP2-hw3.txt) with two discounts factors, 0.1 and 0.9.

1. Run “python infinite.py MDP1-hw3.txt 0.1”

```
Policy        V
4         0.100048354
3         0.004835442
3         0.008748585
4         0.000495597
2         1.000483544
3         0.000483544
2         0.068270582
4         0.000483544
2         0.010004835
4         0.08953431
```

---

2. Run “python infinite.py MDP1-hw3.txt 0.9”

```
Policy        V
4         3.32103321
4         2.923350611
3         2.891354332
1         2.923350611
1         3.6900369
1         2.840745226
2         3.156383584
3         2.907148941
2         2.988929889
4         3.248167345
```

---

3. Run “python infinite.py MDP2-hw3.txt 0.1”

```
Policy        V
4         0.011444345
1         0.010027542
1         0.573256221
3         0.000980766
3         0.060405793
3         0.101010101
3         1.01010101
4         0.00792081
2         0.006040579
3         0.101005733
```

4. Run "python infinite.py MDP2-hw3.txt 0.9"

```
Policy        V
1         4.263157895
1         4.257653253
1         5.188516077
2         3.844028346
3         4.416860679
3         4.736842105
3         5.263157895
2         3.835105625
2         3.975174611
3         4.736835388
```

Part 3: Parking Domain

3-1. MDP Design Description
Design a simple MDP to represent the experience of parking a car.

To generate my MDPs, run "python makeMDP.py *MDPFile* n".

INPUTs:
1. A text file describing MDP (e.g. MDParking1.txt)
2. n, the positive number of parking spots for each A and B
(I used n=10 for this report.)

OUTPUTs: Two text files describing two my MDPs with a different set of parameters
1. MDParking1.txt
2. MDParking2.txt

- **States**

Each state is represented 4 digits ($LNOP$), where L is the name of locations such as A and B, N is the number of spots, O is the status of occupancy, P is the status of parking.

L: A = 1. B = 2
N: 1 to n ($\star$ n is the max number of spots)
O: not occupied = 0, occupied = 1
P: not parked = 0, parked = 1

The total number of state is 8 × n because we have 2 × n spaces from A and B, and each space has 4 different states. For example, A1 has following 4 states:

A100 - A1 space which is not occupied and the agent does not park here.
A101 - A1 space which is not occupied and the agent parks here (terminal state)
A110 - A1 space which is occupied and the agent does not park here
A111 - A1 space which is occupied and the agent parks here (collision, terminal state)

The order of states is based on the circular motion as follows.
$A1(s1 - s4) > B1(s5 - 8) > B2(s9 - 12) > .... > Bn > An > A(n-1) > ... > A2$

- **Actions**

  Actions = {DRIVE=1, PARK=2, EXIT=3}

- **Transition**
  - T_drive: the transition function for action DRIVE
    - Case 1) When P of the current state is 1, it's self-looping because it's a terminal state.

$$T(s_i, DRIVE, s_j) = \begin{cases} 1, & j = i \\ 0, & \text{otherwise} \end{cases}$$

    - Case 2) When OP of the current state is 00, it moves to next space based on the circular order, and flips the coin to decide whether that space is occupied or not. The probability will be determined by the degree of closeness toward the store.

$$T(s_i, DRIVE, s_j) = \begin{cases} P, & j = i + 4 \\ 1 - P, & j = i + 6 \\ 0, & \text{otherwise} \end{cases}$$

      where P is the probability that the space is available.

$$P = \begin{cases} 0.9, & i = 1, \text{ handicap spots} \\ 1.0 - (1.0/(i + \alpha)), & \text{otherwise} \end{cases}$$

      where i is the number of spots. It means that the probability that a parking spot is available is smaller the closer a spot is to the store, and the probability that the handicap spot is available is high.

* Case 3) When OP of the current state is 10, similarly, it moves to next space based on the circular order, and flips the coin to decide whether that space is occupied or not. The probability will be determined by the degree of closeness of a store.

$$T(s_i, DRIVE, s_j) = \begin{cases} P, & j = i + 2 \\ 1 - P, & j = i + 4 \\ 0, & \text{otherwise} \end{cases}$$

Simple example of T_drive

	B200	B201	B210	B211	B300	B301	B310	B311
s5:B100	0.5	0	0.5	0	0	0	0	0
s6:B101	0	0	0	0	0	0	0	0
s7:B110	0.5	0	0.5	0	0	0	0	0
s8:B111	0	0	0	0	0	0	0	0
s9:B200	0	0	0	0	0.67	0	0.33	0
s10:B201	0	1	0	0	0	0	0	0
s11:B210	0	0	0	0	0.67	0	0.33	0
s12:B211	0	0	0	1	0	0	0	0

– T_park: the transition function for action PARK

* Case 1) When P of the current state is 1 (parking status), it's self looping.

$$T(s_i, PARK, s_j) = \begin{cases} 1, & j = i \\ 0, & \text{otherwise} \end{cases}$$

* Case 2) When P of the current state is 0 (not parking status), it changes the status of P from 0 to 1, which means it moves the next state.

$$T(s_i, PARK, s_j) = \begin{cases} 1, & j = i + 1 \\ 0, & \text{otherwise} \end{cases}$$

Simple example of T_park

	s1	s2	s3	s4	s5	s6	s7	s8
s1	0	1	0	0	0	0	0	0
s2	0	1	0	0	0	0	0	0
s3	0	0	0	1	0	0	0	0
s4	0	0	0	1	0	0	0	0
s5	0	0	0	0	0	1	0	0
s6	0	0	0	0	0	1	0	0
s7	0	0	0	0	0	0	0	1
s8	0	0	0	0	0	0	0	1

– T_exit: The transition function for action EXIT

Action EXIT makes a self looping for all states without changing any status, and it leads the state, where P is true, to be a terminal state. Thus, T_exit is just like an identity matrix.

$$T(s_i, EXIT, s_j) = \begin{cases} 1, & j = i \\ 0, & \text{otherwise} \end{cases}$$

- **Reward**

There are four types of the reward/penalty.
1. penalty_collision: It has a high penalty and is given to the state, where OP is 11.
2. penalty_handicap: It describes the penalty of parking in handicap spots, which is given to states of A101 and B101
3. best_reward: It denote the best parking reward when parking A2 or B2, so it is assigned to the states of A201 and B201. For the state of parking other places, it has a smaller reward than A2 and B2 based on the distance.
4. penalty_driving: The cost of driving is given to the remaining states.

3-2. Run My Planner on Two MDPs with a Different Set of Parameters

The two sets of parameter values (rewards and probabilities) as follows:

```
filename = "MDParking1.txt"
# =================== 1st Parameter Setting ================= #
# Parameters for rewards
penalty_driving = -1
penalty_handicap = -10
penalty_collision = -100
best_reward = n + 1

# Parameters of probabilities that a parking spot if available
prob_avail_handicap = 0.9
prob_T = 1.0 # probability temperature
# ============================================================ #

filename = "MDParking2.txt"
# =================== 2nd Parameter Setting ================= #
# Parameters for rewards
penalty_driving = -100
penalty_handicap = -1
```

```
penalty_collision = -100
best_reward = n + 1

# Parameters of probabilities that a parking spot if available
prob_avail_handicap = 0.9
prob_T = 5.0 # probability temperature
# ============================================================ #
```

The probability temperature, prob_T, controls the degree of space availability. The lower value makes most spaces are occupied while the higher value make most spaces are available.

1. 1st MDP: MDParking1.txt
   - Parameters:
     # Reward parameters
     penalty_driving = -1
     penalty_handicap = -10
     penalty_collision = -100
     best_reward = n + 1

     # Probability parameters
     prob_avail_handicap = 0.9
     prob_T = 1.0

   - Expectation:
     Since we have a higher penalty with handicap parking and collision, we would not even try to park in handicap stops or the space occupied. Rather, we would park whenever the states except handicap spots is available.

   - Result: Run "python infinite.py MDParking1.txt 0.9"

```
V
A100 53.53        B500 53.0         B1000 24.02        A600 47.61
A101 -100.        B501 60.0         B1001 10.0         A601 50.0
A110 53.53        B510 36.27        B1010 24.02        A610 47.61
A111 -1000        B511 -1000        B1011 -1000        A611 -1000
B100 60.58        B600 44.0         A1000 27.80        A500 54.01
B101 -100.        B601 50.0         A1001 10.0         A501 60.0
B110 60.58        B610 28.79        A1010 27.80        A510 54.01
B111 -1000        B611 -1000        A1011 -1000        A511 -1000
B200 80.0         B700 35.0         A900 32.00         A400 62.0
B201 90.0*        B701 40.0         A901 20.0          A401 70.0
B210 56.86        B710 21.48        A910 32.00         A410 58.51
B211 -1000        B711 -1000        A911 -1000         A411 -1000
B300 71.0         B800 26.0         A800 36.66         A300 71.0
```

```
B301 80.0        B801 30.0        A801 30.0        A301 80.0
B310 50.68       B810 17.55       A810 36.66       A310 56.22
B311 -1000       B811 -1000       A811 -1000       A311 -1000
B400 62.0        B900 20.61       A700 41.85       A200 80.0
B401 70.0        B901 20.0        A701 40.0        A201 90.0*
B410 43.68       B910 20.61       A710 41.85       A210 47.17
B411 -1000       B911 -1000       A711 -1000       A211 -1000

Policy
A100 1.0         B500 2.0*        B1000 1.0        A600 1.0
A101 1.0         B501 1.0         B1001 1.0        A601 1.0
A110 1.0         B510 1.0         B1010 1.0        A610 1.0
A111 1.0         B511 1.0         B1011 1.0        A611 1.0
B100 1.0         B600 2.0*        A1000 1.0        A500 1.0
B101 1.0         B601 1.0         A1001 1.0        A501 1.0
B110 1.0         B610 1.0         A1010 1.0        A510 1.0
B111 1.0         B611 1.0         A1011 1.0        A511 1.0
B200 2.0*        B700 2.0*        A900 1.0         A400 2.0*
B201 1.0         B701 1.0         A901 1.0         A401 1.0
B210 1.0         B710 1.0         A910 1.0         A410 1.0
B211 1.0         B711 1.0         A911 1.0         A411 1.0
B300 2.0*        B800 2.0*        A800 1.0         A300 2.0*
B301 1.0         B801 1.0         A801 1.0         A301 1.0
B310 1.0         B810 1.0         A810 1.0         A310 1.0
B311 1.0         B811 1.0         A811 1.0         A311 1.0
B400 2.0*        B900 1.0         A700 1.0         A200 2.0*
B401 1.0         B901 1.0         A701 1.0         A201 1.0
B410 1.0         B910 1.0         A710 1.0         A210 1.0
B411 1.0         B911 1.0         A711 1.0         A211 1.0
```

As I expected, it does not try to park in handicap spaces such as A1XX and B1XX, rather the agent parks in B2-B8 and A2-A4 whenever the spaces are available. Additionally, it also prefers the front places which are closer to the store because that places have higher rewards while driving cost is quite small. In this regard, the highest value is given to A2 and B2.

1. 2nd MDP2: MDParking2.txt
   - Parameters:
     # Reward parameters
     penalty_driving = -100
     penalty_handicap = -1
     penalty_collision = -100
     best_reward = n + 1

# Probability parameters
prob_avail_handicap = 0.9
prob_T = 5.0

I changed some parameter values in my 2nd MDP to increase the availability of all spaces by modifying prob_T value, and to prevent the agent from a long driving by providing the high penalty for driving, which is now -100 from -1. I also changed the penalty of parking in handicap from -10 to -1.

– Expectation:
In this MDP, the agent is better to park immediately once the current place is available since driving cost is now very high. Also, since the penalty of parking in handicap is lower than before, the agent can even dare to drive that place to save the cost of driving.

– Result: Run "python infinite.py MDParking2.txt 0.9"

```
V
A100 -109.        B500 -46.0        B1000 -91.0        A600 -55.0
A101 -10.0        B501 60.0         B1001 10.0         A601 50.0
A110 -200.        B510 -159.        B1010 -187.        A610 -151.
A111 -1000        B511 -1000        B1011 -1000        A611 -1000
B100 -109.        B600 -55.0        A1000 -91.0        A500 -46.0
B101 -10.0        B601 50.0         A1001 10.0         A501 60.0
B110 -135.        B610 -166.        A1010 -180.        A510 -144.
B111 -1000        B611 -1000        A1011 -1000        A511 -1000
B200 -19.0        B700 -64.0        A900 -82.0         A400 -37.0
B201 90.0*        B701 40.0         A901 20.0          A401 70.0
B210 -140.        B710 -173.        A910 -172.         A410 -140.
B211 -1000        B711 -1000        A911 -1000         A411 -1000
B300 -28.0        B800 -73.0        A800 -73.0         A300 -28.0
B301 80.0         B801 30.0         A801 30.0          A301 80.0
B310 -145.        B810 -181.        A810 -165.         A310 -145.
B311 -1000        B811 -1000        A811 -1000         A311 -1000
B400 -37.0        B900 -82.0        A700 -64.0         A200 -19.0
B401 70.0         B901 20.0         A701 40.0          A201 90.0*
B410 -152.        B910 -187.        A710 -158.         A210 -206.
B411 -1000        B911 -1000        A711 -1000         A211 -1000

Policy
A100 2.0*         B500 2.0*         B1000 2.0*         A600 2.0*
A101 1.0          B501 1.0          B1001 1.0          A601 1.0
A110 1.0          B510 1.0          B1010 1.0          A610 1.0
A111 1.0          B511 1.0          B1011 1.0          A611 1.0
```

```
B100 2.0*       B600 2.0*       A1000 2.0*       A500 2.0*
B101 1.0        B601 1.0        A1001 1.0        A501 1.0
B110 1.0        B610 1.0        A1010 1.0        A510 1.0
B111 1.0        B611 1.0        A1011 1.0        A511 1.0
B200 2.0*       B700 2.0*       A900 2.0*        A400 2.0*
B201 1.0        B701 1.0        A901 1.0         A401 1.0
B210 1.0        B710 1.0        A910 1.0         A410 1.0
B211 1.0        B711 1.0        A911 1.0         A411 1.0
B300 2.0*       B800 2.0*       A800 2.0*        A300 2.0*
B301 1.0        B801 1.0        A801 1.0         A301 1.0
B310 1.0        B810 1.0        A810 1.0         A310 1.0
B311 1.0        B811 1.0        A811 1.0         A311 1.0
B400 2.0*       B900 2.0*       A700 2.0*        A200 2.0*
B401 1.0        B901 1.0        A701 1.0         A201 1.0
B410 1.0        B910 1.0        A710 1.0         A210 1.0
B411 1.0        B911 1.0        A711 1.0         A211 1.0
```

As I expected, the agent now parks every state where the space is not occupied even including handicap spots because the penalty is lower than the cost of driving. Thus, this outcome shows a learning that the price of penalty should take a consideration of other costs of factors influencing parking behavior. Without careful considerations, a small penalty would not make any changes to prevent people from undesirable behavior.