Reinforcement Learning- Assignment 1

YaChu Yang
11571276
ya-chu.yang@student.uva.nl
JiunHan Chen
11649712
chen.jiunhan@gmail.com

November 16, 2018

Abstract

1 Introduction

Question 1.1

The fact that the number of states often grow exponentially with the number of state variables.

Question 1.2

(a) 25 x 25 = 625
(b) There are many methods to reduce the dimension of this problem. For example, we can rotate and mirror the grid world and consider some states as one single state. In this question, we choose to represent the state as relative position and consider the position of prey as origin. Combining the rotation and without considering boundary conditions, we can approximate it and reduce number of the state as 14. But we need more memory spaces to record how we map the grid world of each situation to our states. For example, we will consider the following two cases as the same state, due to the same relative position:
case 1:

```
[ ]  [ ]  [ ]  [ ]  [ ]
[ ]  [ ]  [ ]  [ ]  [ ]
[ ]  [ ]  [o]  [ ]  [ ]
[ ]  [ ]  [ ]  [ ]  [ ]
[x]  [ ]  [ ]  [ ]  [ ]
```

case 2:

```
[ ]  [ ]  [ ]  [ ]  [o]
[ ]  [ ]  [ ]  [ ]  [ ]
[ ]  [ ]  [x]  [ ]  [ ]
[ ]  [ ]  [ ]  [ ]  [ ]
[ ]  [ ]  [ ]  [ ]  [ ]
```

(c) 14 states. Because there is only 24 different relative positions after combining rotation, further more we can mirror the grid world with line $x = y$. We don't consider boundary condition because it has no much impact on strategy of predator. Predator should move toward the prey, not move toward the boundary, so the action space can be reduced too. (d) To reduce the dimension such that we can prevent from curse of dimensionality.
(e) Firstly, we can remove early finished game. Secondly, we can remove more states with rotation and mirror. For example, the following two cases can be mapped into the same states because of the symmetry of this game:
case 1:

```
 o    x   [ ]
 o    x   [ ]
[ ]  [ ]  [ ]
```

case 2:

```
[ ]   o    o
[ ]   x    x
[ ]  [ ]  [ ]
```

Question 1.3

A non-greedy agent will learn better policy comparing to the greedy policy one. Since the greedy algorithm tends to choose the action that has maximum reward, which could lead the agent fall into local minimum without enough exploration. However, the non-greedy agent tries to explore new actions, which might be the better policy.

Question 1.4

(a) We can set a decay rate $k < 1$ for epsilon such that $\epsilon_{t+1} = k \cdot \epsilon_t$, which obeys the convergence condition Greedy in the Limit with Infinite Exploration (GLIE).
(b) It depends how the opponent changes its strategy, we must update explored states. If the strategy changes in high frequency, the epsilon might not lead to enough revisiting for explored state. One technique we can use is **adaptive epsilon**, which can solve the dilemma between exploration and exploitation. If the reward have a dramatically drop compared to previous game round, we may increase the epsilon to explore the new strategy of opponent.

2 Exploration

Question 2.1

There will be 1 - ϵ probability of selecting the greedy action.

Question 2.2

$A_4 = 2$ is certain to be the result of exploration. First, we can see that the action value of choosing action 2 at previous timestamp is the most small comparing to other action values. Thus, at timestamp 4, choosing action 2 again is definitely not motivated by greedy algorithm. Therefore, we can be certain that A_4 is the result of exploration.

Question 2.3

At first, let's consider pessimistic approach. In Table 1, the 2nd column assume "+1 arm" is selected at initial random, the 3rd column assume "-1 arm" is selected, which is labeled with " * ". Q value can be computed by $Q_{t+1}(a) = Q_t(a) + \frac{1}{k_a+1}[r_{t+1} - Q_t(a)]$ or simple average. Table 2 shows optimistic approach. In step 3(fourth row of tables), it shows the result of Q-values.

step	Q of +1 arm*	Q of -1 arm	A	R	Q of +1 arm	Q of -1 arm*	A	R
0	-5	-5			-5	-5		
1	-2	-5	+1 arm	1	-5	-3	-1 arm	-1
2	-1	-5	+1 arm	1	-5	$-\frac{7}{3}$	-1 arm	-1
3	-0.5	-5	+1 arm	1	-5	-2	-1 arm	-1
reward				3				-3

Table 1: Calculation process of optimistic approach

step	Q of +1 arm*	Q of -1 arm	A	R	Q of +1 arm	Q of -1 arm*	A	R
0	5	5			5	5		
1	3	5	+1 arm	1	5	2	-1 arm	-1
2	3	2	-1 arm	-1	3	2	+1 arm	1
3	$\frac{7}{3}$	2	+1 arm	1	$\frac{7}{3}$	2	+1 arm	1
reward				1				1

Table 2: Calculation process of pessimistic approach

Question 2.4

For pessimistic approach, the expectation of reward is $\frac{3+-3}{2} = 0$, and for optimistic approach the expectation of reward is $\frac{1+1}{2} = 1$ in three steps. If the initialization is not random for tie break, in an alternative way, we can always select the first arm. But it depends on the first arm is "+1 arm" or "-1 arm". For pessimistic approach, if the first arm is "+1 arm", it will always select "+1 arm", which can maximize the return. But if the first arm is "-1 arm", the final result will be negative infinity. For optimistic approach, because it can correctly estimate Q value, no matter which is the first arm, optimistic approach is able to maximize the return.

Question 2.5

Optimistic approach is better for estimation of the Q-values. Because this approach motivates the agent to do exploration between arms instead of always choosing the arms with highest reward, every arms' Q-values can be improved and learned.

Question 2.6

Optimistic approach is better for exploration. In this approach, arms which have already been visited will have a value lower than the value of arms which haven't yet been visited due to much higher initial action-values comparing to the largest possible reward, which is 3. Thus, arms haven been visited will be more attractive and better for exploration.

3 Markov Decesion Process

Question 3.1

(a)
1. Chess:
States: positions of chess pieces
Actions: which piece and moves to where
Reward: It could be win = 1 and lose = -1

2. Petroleum refinery:
States: It could be continuous parameters space and current stage of refinement.
Actions: Adjusts parameters.
Reward: It could be optimal target function of yield/cost/quality.

3. Gazelle Calf
States: positions and velocity of body parts, tense of muscles.
Actions: Changing tense of muscles.
Reward: It could be speed with considering stability.

4. Mobile robot
States: Distance between itself and charging station and current battery level
Actions: Search room or charge battery.
Reward: Out of battery = -100, Finish room search = 1

(b) A robot solves a maze can be modeled by MDP. The maze is a grid. Thus, the state space is composed by every cell in the grid. The action space is going back, going straightforward, turning left or turning right. Furthermore, for the reward, when a robots finds the exit, it can get +1, otherwise, when a robot moves one cell to another cell, it will cost the robot -0.04 points to encourage the robot finds the shortest path.

(c) MDP framework means that the future state is independent of past states given the current state. Thus, for time series prediction which needs past memory, it doesn't fit the framework. For instance, stock prediction or a language model for predicting next word belongs to the problem with difficulty solving with an MDP.

(d) Accelerator, brake, and steering wheel are easier too be quantified. Quantification is important when we need to model our world and do the training in our model. Also these actions can be directly changed by manipulator. But the disadvantage might emerge when we have a far distance target to visit. The low level actions are hard to lead our car to our destination because these actions more focus on local and current situation. Another disadvantage might come from when we must accurately control our machine to specific quantified value. For example, we restrict the range of accelerator as [0,1], and we want to set the accelerator as 1. But it's impossible that we can accelerate immediately, there must be a short time difference and some limitations of machine. This property might cause problem when we do the reinforcement learning.

(e) I assume our car has auto navigation which can move to the place we want. It's convenient especially if we want to build a self-driving car, after the training, the user only needs to input the destination and the car will lead the user to destination. But the problem is that the action focus too much on global situation. It might ignore some random events, for example, pedestrians walking on the road, the car might not have enough action states to deal with such case.

(f) We can simply combine low level action state and high level action state. But most of situation we only need the high level action states, and in some emergency cases, low level action states will take over it. Our action states can be accelerator, brake, steering, destination and one bit to decide if low level should take over. We can set the reward as safety and time, less time and safer are better.

Question 3.2

(a) $G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$

(b)

$$\begin{aligned}
s &= \sum_{k=0}^{\infty} \gamma^k \\
\gamma s &= \sum_{k=0}^{\infty} \gamma^{k+1} \\
\Rightarrow s - \gamma s &= 1 \\
\Rightarrow s &= \frac{1}{1-\gamma} \\
\Rightarrow \sum_{k=0}^{\infty} \gamma^{k+1} &= \frac{1}{1-\gamma}
\end{aligned}$$

(c) Equation 3.7, $G_t = R_{t+1} + R_{t+2}, ... + R_T$, is problematic since every state's value in every episodes will all be 1 if we don't include discount factor in the equation. That means the robot can't find the shortest path and the policy can't be improved according to the state value.

(d) Adding the discount factor, the state's value won't be the same anymore. Thus, the agent can modify its policy according to the state value. Furthermore, the agent tries to select actions so that the sum of discounted reward received by the agent in the future is maximized, which means the agent will try to find the shortest path to get out the maze.

(e) Instead of receiving the reward of 0 when a agent doesn't escape from the maze, we can set the reward of the negative value, like -0.01, so that a robot is encouraged to find the shortest path and improve its policy to receive the maximum reward.

4 Dynamic Programming

Question 4.1

1. stochastic: $v^{\pi}(s) = \Sigma_a \pi(a|s) \Sigma_{s',r} p(s^{'}, r|s, a)[r + \gamma v^{\pi}(s^{')]}$
2. deterministic: $v^{\pi}(s) = \Sigma_{s',r} p(s^{'}, r|s, a)[r + \gamma v^{\pi}(s^{')]}$

Question 4.2

$Q^{\pi}(s, a) = \Sigma_{s',r} p(s^{'}, r|s, a)[r + \gamma v^{\pi}(s^{'})] = \Sigma_{s',r} p(s^{'}, r|s, a)[r + \gamma Q^{\pi}(s^{'}, a^{'})]$

Question 4.3

$\pi(s) \longleftarrow argmax_a Q^{\pi}(s, a) = argmax_a \Sigma_{s',r} p(s^{'}, r|s, a)[r + \gamma Q^{\pi}(s^{'}, a^{'})]$

Question 4.4

$Q(s, a) \longleftarrow \Sigma_{s',r} p(s^{'}, r|s, a)[r + \gamma max_{a'} Q(s^{'}, a^{'})]$

5 Monte Carlo

Question 5.1

(a) $5 \times (0.9^2 + 0.9^4 + 0.9^3)/3 = 3.6585$
(b) $5 \times (0.9 + 0.9^2 + 0.9 + 0.9^2 + 0.9^3 + 0.9^4 + 0.9^5 + 0.9 + 0.9^2 + 0.9^3 + 0.9^4)/12 = 3.5377875$

Question 5.2

Although in first-visit method, ordinary importance sampling is unbiased, the disadvantage is that it suffers from unbounded variance of its ratio $\rho_{t:T(t)-1}$. For single observation, if the ratio is 10, the estimation will be 10 times of observed return, which might cause unstable estimation.

Question 5.3

For a single observation, the observed return is sampled from behavior policy not from target policy. The fact implies weighted importance sampling is biased in both first-visit and every-visit. Even with above disadvantage, weighted importance sampling is still strongly preferred. For a single observation, the ratio will be canceled, and only G_t is left. So the variance of ratio is relatively stable.

6 Temporal Difference Learning

Question 6.1

(a)

```
updata V(A): -0.30000000000000004
updata V(B): 0.37000000000000005
updata V(A): -0.7000000000000001
updata V(A): -0.893
updata V(B): 0.43300000000000005
V(A) = -0.893, V(B) = 0.433
```

(b)

```
updata V(A): -0.30000000000000004
updata V(B): -0.33
updata V(A): -0.9030000000000001
updata V(A): -1.0127000000000002
updata V(B): -0.197
V(A) = -1.0127, V(B) = -0.197
```

(c)

```
updata Q(A, 1): -0.30000000000000004
updata Q(B, 1): 0.4
updata Q(A, 2): -0.43
updata Q(A, 1): -0.5700000000000001
updata Q(B, 2): 0.13999999999999999
Q(A,1) = -0.57, Q(A,2) = -0.43
Q(B,1) = 0.4, Q(B,2) = 0.14
```

(d)

```
updata Q(A, 1): -0.30000000000000004
updata Q(B, 1): 0.4
updata Q(A, 2): -0.4
updata Q(A, 1): -0.53
updata Q(B, 2): 0.13999999999999999
Q(A,1) = -0.53, Q(A,2) = -0.4
Q(B,1) = 0.4, Q(B,2) = 0.14
```

Question 6.2

The initial state is A, and according to Q-learning, $\underset{a}{\text{argmax}}\ Q(s=A,a)=2$ and $\underset{a}{\text{argmax}}\ Q(s=B,a)$ $=1$. So our policy $\pi(s=A)=[0.0,1.0],\pi(s=B)=[1.0,0.0]$, and the entries of array from left to right are the probability of choosing action 1 and 2.

Question 6.3

(a) For a fixed policy $\pi_{student}$, it only updates two state-action values, which are pair (s=A, a=2) and (s=B, a=1). For the random policy, it will update all Q-value. (b) For π_{random}, it explores too much. The learning should be able to exploit the best Q in later stage, and only adjust minor actions to achieve better precision. For $\pi_student$, it exploits too much. The learning should be able to explore more actions and states to create enough samples for every action-state pair. (c) $\varepsilon-greedy$ can be beneficial. Because it combines exploration and exploitation to not only create enough samples for every action-state pair, but also search more important action-state space to achieve better precision.

7 Temporal Difference Learning(Theory)

Question 7.1

$$\begin{aligned}
V_M(S) &= \frac{1}{M}\sum_{n=1}^{M} G_n(S) \\
&= \frac{1}{M-1}\sum_{n=1}^{M-1} G_n(S) - \frac{1}{M-1}\sum_{n=1}^{M-1} G_n(S) + \frac{1}{M}\sum_{n=1}^{M} G_n(S) \\
&= \frac{1}{M-1}\sum_{n=1}^{M-1} G_n(S) + \frac{1}{M}[\sum_{n=1}^{M} G_n(S) - \frac{M}{M-1}\sum_{n=1}^{M-1} G_n(S)] \\
&= \frac{1}{M-1}\sum_{n=1}^{M-1} G_n(S) + \frac{1}{M}[G_M(S) + \sum_{n=1}^{M-1} G_n(S) - \frac{M}{M-1}\sum_{n=1}^{M-1} G_n(S)] \\
&= \frac{1}{M-1}\sum_{n=1}^{M-1} G_n(S) + \frac{1}{M}[G_M(S) - \frac{1}{M-1}\sum_{n=1}^{M-1} G_n(S)] \\
&= V_{M-1}(S) + \frac{1}{M}[G_M(S) - V_{M-1}(S)]
\end{aligned}$$

Question 7.2

(a)

$$\begin{aligned}\delta_t &= R_{t+1} + \gamma V^\pi(S_{t+1}) - V(S_t)\\ \mathbb{E}[\delta_t|S_t = s] &= \mathbb{E}[R_{t+1} + \gamma V^\pi(S_{t+1})|S_t = s] - V(s)\\ &= V^\pi(s) - V(s)\end{aligned}$$

(b)

$$\begin{aligned}\delta_t &= R_{t+1} + \gamma V^\pi(S_{t+1}) - V(S_t)\\ \mathbb{E}[\delta_t|S_t = s, A_t = a] &= \mathbb{E}[R_{t+1} + \gamma V^\pi(S_{t+1})|S_t = s, A_t = a] - \mathbb{E}[V(S)|S_t = s, A_t = a]\\ &= q^\pi(s,a) - q(s,a)\end{aligned}$$

Question 7.3

(a)

$$\begin{aligned}G_t - V(S_t) &= R_{t+1} + \gamma G_{t+1} - [V(S_t) - \alpha\delta_t] + \gamma V[S_{t+1}] - \gamma V[S_{t+1}]\\ &= \delta_t + \gamma(G_{t+1} - V(S_{t+1})) + \alpha\delta_t = (1+\alpha)\delta_t + \gamma(G_{t+1} - V(S_{t+1}))\\ &= (1+\alpha)\delta_t + \gamma(1+\alpha)\delta_{t+1} + \gamma^2(G_{t+2} - V(S_{t+2}))\\ &= (1+\alpha)[\delta_t + \gamma\delta_{t+1} + \gamma^2\delta_{t+2} + ... + \gamma^{T-t-1}\delta_{T-1}] + \gamma^{T-t}(G_T - V(S_T))\\ &= (1+\alpha)\sum_{k=t}^{T-1}\gamma^{k-t}\delta_k\end{aligned}$$

(b)

$$\begin{aligned}G_{t:t+n} - V(S_t) &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... + \gamma^n V(S_{t+n}) - V(S_t) + \gamma V[S_{t+1}] - \gamma V[S_{t+1}]\\ &= \delta_t + \gamma[R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + ... + \gamma^{n-1}V(S_{t+n}) - V(S_{t+1})]\\ &= \delta_t + \gamma[G_{t+1:t+n} - V(S_{t+1})]\\ &= \delta_t + \gamma\delta_{t+1} + \gamma^2\delta_{t+2} + ... + \delta^{n-1}[G_{t+n-1:t+n} - V(S_{t+n-1})]\\ &= \delta_t + \gamma\delta_{t+1} + \gamma^2\delta_{t+2} + ... + \gamma^{n-1}[R_{t+n} + \gamma V(S_{t+n}) - V(S_{t+n-1})]\\ &= \delta_t + \gamma\delta_{t+1} + \gamma^2\delta_{t+2} + ... + \gamma^{n-1}\delta_{t+n-1}\\ &= \sum_{k=t}^{n-1}\gamma^{k-t}\delta_k\end{aligned}$$

8 Maximization Bias

Question 8.1

	SARSA	Q-learning
Q(B, a1)	1	1
Q(B, a2)	1	1
Q(B, a3)	2	2
Q(B, a4)	0	0
Q(A, left)	1	2
Q(A, right)	1.5	1.5

Question 8.2

For Q-learning, when the agent is at state A, it will choose going to the left because Q(A, left) is larger than Q(A. right). This is the result of the maximization bias since in expectation, it would be beneficial to take the action right with higher expected reward of 1.5, which is higher than the expected reward of 1 when going left.
Furthermore, from the table above, we can see that only Q-learning suffers from this bias. This is all because the update for Q-learning involves a max over all the actions in the next state, which could lead to the overestimated value of some actions. For Sarsa, since it is on-policy method, the update of Q always follows the policy, it doesn't have the problem of maximization bias.

Question 8.3

The concept of Double Q-learning is that using two different Q function, Q1 and Q2, to determine the maximizing action and to estimate the value of action-state. Through this, we can avoid overestimating value of action-state.
We can look the example when the agent is at state B, executing action a_3. We assume that $Q_1(B, a3) = 2$ and $Q_2(B, a3) = 0$. Though we use Q_1 to determine the maximizing action, the value of $Q_1(A, left)$ will still be 0, not 2, since using Q_2 to estimate the value of $Q_1(A, left)$. Therefore, double q-learning solves the issue that the agent keeps choosing the action(in this example, going left from state A) that might be too positive.

Question 8.4

We assume $\gamma = 1$ in this question.

	SARSA	Q-learning
Q(B, a1)	1	1
Q(B, a2)	1	1
Q(B, a3)	1	1
Q(B, a4)	1	1
Q(A, left)	1	1
Q(A, right)	1.5	1.5

9 Model-based RL

Question 9.1

The transition probability and the reward are represented as a tuple, (P(s'|s,a), R) in the table.

	s' = state A	s' = state B	s' = state T
state A, action 1	($\frac{1}{4}$, 3)	($\frac{3}{4}$, -3)	(0, 0)
state A, action 2	(1, -1.5)	(0, 0)	(0, 0)
state B, action 1	($\frac{1}{2}$, 4)	(0, 0)	($\frac{1}{2}$, -10)
state B, action 2	(0, 0)	(0, 0)	(1, 1)

Question 9.2

(a)The model learning in Dyna-Q assumes that the environment is deterministic. However, the reward of doing Action 2 from State A to State A is not deterministic in this example.

(b) We can change the model learning,

$$\textit{Model(S,A)} \longleftarrow R, S',$$

in the table of Dyna-Q Algorithm in the book in Section 8.2 to

$$\textit{Model(S,A)} \longleftarrow max(R_{current}, R_{model}), S',$$

where R_{model} is the last-observed next reward recorded in the model, and $R_{current}$ is the reward observed at current timestamp. Through choosing the maximum over different rewards received at the same state, the model tends to be optimistic in predicting the greater reward than its true value. Then the planned policy has high probability to do these actions with overestimated reward. Through multiple times of exploration, the policy will find out the overestimation and correct the model error.

(c) This modification can't deal with the situation that the environment becomes better than is was, however, the previous correct policy doesn't even detect the better option for the agents. We can fix this issue by adding the concept of how long a state-action hasn't been tried since last visit, which is denoted as τ. Through these records, we can encourage the agent to try these long-untried state-action, and give the agent more reward of k$\sqrt{\tau}$ than its original reward, r. Therefore, the agent will have more change to discover the model error caused by changing of the environment.

Question 9.3

(a) After the first episode, both methods end up with same Q-value. Because for Dyna-Q, the model is learned from only one episode, then the reward and the next state got from the model and observed in real experience are exactly identical to each other. Therefore, doing planning is actually using the same data as doing learning. Thus, when both methods go converge, their Q-value will be the same.

(b) After the second episode, their Q-value ends up with being different. It is because that for Q-learning, after discarding the first episode, the agent will only learn from the second episode; however, for Dyna-Q, the agent will update its Q value according to both episodes since the model already learns from the first episode though which is discarded.

(c) Q value of both methods would converge to the same value if we continued sampling new episodes indefinitely. When the number of sampled episodes increasing, Q value in Q learning should converge to the expected value; and for Dyna-Q learning, the learned model will be more similar to the real world model, which can also result in the convergence the update of Q value in planning to the expected value.