Reinforcement Learning — Homework 1

Tarun Krishna **Dhruba Pujary**
11593040 11576200
University of Amsterdam
{dhruba.pujary, tarun.krishna}@student.uva.nl

1 Introduction

1. The 'curse of dimensionality' means that the problem's computational requirements grows exponentially with the number of state variables, or increasing number of dimension.
2. (a) The possible number of states are 625; 5×5 for predator and 5×5 for prey and total combination of such predator and prey is 25×25.
   (b) The toroidal grid is continuous in a sense that if a agent hitting the left most boundary will re-enter the grid from the right. Thus shifting the absolute position of agent relative to the grid doesn't change anything with respect to the overall dynamics of the system. Thus we can define the new state representation by keeping the position of the predator always at the center and move the prey relative to this setting. This new grid coordinates can be realized using a mathematical functions and then map it to fixed grid index from 1-25.
   (c) The possible number of states now is 25.
   (d) The advantage of doing is that the number of states reduces tremendously. In addition, the new system is configured is such a way that the distance of travelling towards the prey gradually increase in all direction direction with the corners at the farthest.
   (e) Considering the number of states of Tic-Tac-Toe naively is 3^9; i.e. for each tile/grid there are three possible values, no symbol,'X' or 'O'. Out of these states, the states where number of 'X' (or 'O') symbol is more than the number of'O'+1, are invalid since this game is played between two opponents. We can further reduce the number of states by using the symmetric property of the problem. Many of the states are mirror image with respect to the middle row, middle column or the diagonals.
3. (a) The non-greedy agent will a better policy in the end because it explore more than greedy policy. The greedy agent will look for short sighted maximization of reward which may not be the overall maximum reward that can be obtained.
4. (a) This can be done by reducing the exploring probability value ϵ by setting $\epsilon = 1/(c+t)$ where t is time step and c is some constant(default 0).
   (b) No, if the opponents strategy changes our assumption that the observations from the environment(which includes the opponents actions) remains the same or deterministic becomes invalid. One way is to keep the policy ϵ-greedy and or somehow incorporate the change in behaviour into a model of the environment and learn from it before making the actual move.

2 Exploration

1. 1-ϵ
2. Action A_4 and A_5 are result of exploration.
3. Using sample-average method,

(a) Initial value = -5

$timestep$	$action_1(+1)$	$action_2(-1)$
0	$\underline{-5}$	-5
1	$\underline{1}$	-5
2	$\underline{1}$	-5
3	$\underline{1}$	-5

Breaking tie by selecting $action_1$

(b) Initial value = 5

$timestep$	$action_1(+1)$	$action_2(-1)$
0	$\underline{5}$	5
1	1	$\underline{5}$
2	$\underline{1}$	-1
3	$\underline{1}$	-1

Breaking tie by selecting $action_1$

4. Both initialization give the same maximum reward if above tie breaking policy is followed. If tie is broken differently, the results are different.

(a) Initial value = -5

$timestep$	$action_1(+1)$	$action_2(-1)$
0	-5	$\underline{-5}$
1	-5	$\underline{-1}$
2	-5	$\underline{-1}$
3	-5	$\underline{-1}$

Breaking tie by selecting $action_2$

(b) Initial value = 5

$timestep$	$action_1(+1)$	$action_2(-1)$
0	5	$\underline{5}$
1	$\underline{5}$	-1
2	$\underline{1}$	-1
3	$\underline{1}$	-1

Breaking tie by selecting $action_2$

5. Optimistic initialization leads to better estimation.
6. Optimistic initialization method is better for exploration because every action that is not yet taken will have the maximum or optimistic initial value and eventually be selected at least once if greedy policy is followed.

3 Markov Decision Processes

1. (a) i. Chess:: *State space*: 8×8 grid and the each unique piece information.
Action space: All the action possible by each of the different pieces (according to the rule book)
Reward signal: Win(positive, +1), draw(neutral, 0) or loss(negative, -1) of a game.

ii. Petroleum refinery:: *State space*: Continuous space of all possible values result of tweaking the parameter values.
Action space: Changing the parameter values, i.e. increasing or decreasing.
Reward signal: Better cost efficient output as compared to settings suggested by engineers will be positive reward. And if wastage of material with sub-optimal output will be negative reward.

iii. Gazelle calf:: *State space*: The body contact with the floor, speed/velocity.
Action space: Force applied on its feet for moving forward or stopping as well as for orientation.
Reward signal: Positive reward $c > 1$ for moving forward and -1 for each time step.

iv. Mobile robot:: *State space*: Battery charge and information of immediate surrounding.
Action space: Forward or backward to search for garbage or re-charge the battery
Reward signal: Positive reward for collecting trash and negative reward for draining out the battery before reaching re-charge point.

(b) An example would be a autonomous drone. *State space*: current velocity, angular momentum, direction of wind, atmospheric pressure.
Action space: rotor speed, tilt of blades.
Reward signal: Positive reward for flying, moving forward and landing. Negative reward for crashing.

(c) An example would be predicting the election winner based on the sentiment analysis of twitter/Facebook updates, comments, likes and shares of the candidates. This cannot be easily solve because the states defined using the statistics will never be sufficient enough to capture all the previous states. In other words, it will not be a Markov process as states are not sufficient to represent the historic as well complete information of the environment.

(d) Choosing the actions as handling the accelerator, brake and steering wheel will allow the agent to learn to drive with full control on the car. This setting will keep the details of mechanism of engine or the amount of force to be applied on the muscles to rotate the steering wheel will get absorbed in the environment dynamics, giving the agent full freedom to learn the agent to drive just as we human do. The disadvantage of doing is that few small details which actually improve the operation of the car will be left out. For example, the air pressure of the tire plays an important role in efficiency of the car. If the small details are left out and there is no such state information monitoring these details or any action to deal with than a tire puncher will stop the vehicle, and the agent cannot do anything about it.

(e) Choosing the action as where to drive will reduce the number of action to a very few in counts. The disadvantage of doing so is that now the environment comprises of many changing factors which will give different states and rewards. The convergence of the algorithm will be very slow as there may be very noisy rewards for being in the same state. Its like giving the steering wheel to the agent without the control of speed of the car. When moving towards a wall, the state could give a lot of information but only a few will be useful as the only decision to make is which direction to drive. On experiencing multiple episodes of collision, moving right or left the agent will be able to avoid the wall. But if the speed is increased, the agent has to learn again to avoid the wall.

(f) The way to combine both is to have all the actions i.e. the handling of acceleration, brake, steering wheel and choosing where to drive. The first three will give the agent the ability to control the motion of the car whereas the later will give a definite goal to move the car towards. Its like driving a car from one destination to another by keeping an eye on the shortest path or paths using Google map.

2. (a)

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... + \gamma^T R_{t+T+1} \qquad = \sum_{k=0}^{T} \gamma^k R_{t+k+1}$$

(b) let $\sum_{k=0}^{\infty} \gamma^k$ be denoted as y:

$$
\begin{aligned}
y &= 1 + \gamma^1 + \gamma^2 + \gamma^3 + ... + \gamma^\infty \\
y - 1 &= \gamma^1 + \gamma^2 + \gamma^3 + ... + \gamma^\infty \\
\frac{y-1}{\gamma} &= 1 + \gamma^1 + \gamma^2 + \gamma^3 + ... + \gamma^\infty \\
\frac{y-1}{\gamma} &= y \\
y - \gamma y &= 1 \\
y &= \frac{1}{1-\gamma} \\
\sum_{k=0}^{\infty} \gamma^k &= \frac{1}{1-\gamma}
\end{aligned}
$$

(c) There is no improvement because the agent is not trying to find the optimal path to the goal. All the path to the goal, no matter how many time steps it takes, gives equal reward of 0 and a final reward of 1 upon reaching the goal. So, the agent is not trying to reduce the time taken to reach the goal rather it is randomly moving around as all the nearest states gives equal reward.

(d) By introducing the discount factor, the reward received after n step is more than $n + 1$. Thus the agent will try to earn maximum reward by reducing the time steps taken to reach the goal and hence will find the optimal path.

(e) By changing the reward function to give -1 reward for each time step and a reward of zero on reaching the goal, the agent will try to maximize its rewards. It will learn to find the shortest path in doing so.

4 Dynamic Programming

1. Under stochastic policy π: $v^\pi(s) = \sum_a \pi(a|s) q^\pi(s, a)$.
   Under deterministic policy π: $v^\pi(s) = q^\pi(s, a)$.

2. Policy Evaluation for $Q^\pi(s, a)$ is given by:

   (a) **Initialization**

   $Q(s, a) \in R$ and $\pi(s) \in A(s)$ arbitrarily for all $s \in S$
   Policy Evaluation
   Loop:
   $\Delta \leftarrow 0$
   Loop for each $s \in S$:
   $q \leftarrow Q(s, \pi(s))$
   $Q(s, \pi(s)) \leftarrow \sum_{s',r} p(s'.r|s, \pi(s))[r + \gamma \sum_a \pi(a|s') Q(s', a)]$
   $\Delta \leftarrow max(\Delta, |q - Q(s, \pi(s))|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy improvement for $Q^\pi(s, a)$ (continuing from above):

   (a) *policy-stable* $\leftarrow true$
   Loop for each $s \in S$:
   old-action $\leftarrow \pi(s)$
   $\pi(s) \leftarrow argmax_a Q(s, a)$
   If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $Q \approx q*$ and $\pi \approx \pi*$; else go to **Policy Evaluation**

4. $q_{k+1}(s, a) = \sum_{s',r} p(s'.r|s, a)[r + \gamma \max_a q_k(s', a)]$

5 Monte Carlo

1. (a) First-visit MC $V(s_0) = \frac{5}{3}\gamma^2(1+\gamma+\gamma^2) = 3.6585$
   (b) Every-visit MC $V(s_0) = \frac{5}{12}\big(3(1+\gamma+\gamma^2) + 2\gamma^3(1+\gamma)\big) = 4.268$
2. The variance of ordinary importance sampling is in general unbounded because the variance of the ratios can be unbounded. As a result ordinary importance sampling could result in high variance.
3. Weighted importance sampling is biased.

6 Temporal Difference Learning (Application)

1. (a) TD(0) estimate: V(A) = -0.893, V(B) = 0.433
   (b) 3-step TD V(A) = -0.983, V(B) = -0.17
   (c) SARSA: Q(A, 1) = -0.57, Q(B, 2) = 0.1, Q(A, 2) = -0.43, Q(B, 1) = 0.4
   (d) Q-learning: Q(A, 1) = -0.53, Q(A, 2) = -0.4, Q(B, 1) = 0.4, Q(B, 2) = 0.1
2. A deterministic policy could be greedy as:
   $\pi(a|s = A) = argamx_a(Q(A,1), Q(A,2)) =$ action 2
   $\pi(a|s = B) = argmax_a(Q(B,1), Q(B,2)) =$ action 1
3. (a) With π_{random} we will be able to obtain the true estimates always but will take a while to converge to true estimate. While in case of $\pi_{student}$ (greedy policy) we might not be able to converge to true estimates because of the greedy actions.
   (b) Using a π_{random} policy doesn't take the advantage of learned values or in other words doesn't exploit. Using $\pi_{student}$ policy which is greedy policy is also not good because of it lacks exploring property. The agent is short-sighted for immediate good rewards which doesn't always leads to optimal/maximum rewards.
   (c) Using a ϵ-greedy policy allows the agent to explore by choosing a random action with probability ϵ. This will make sure that all the actions are explored which may give low immediate rewards but overall optimal reward by following the policy. It also utilize the learned values by taking greedy action with probability $1-\epsilon$ which is better than taking random actions every time.

7 Temporal Difference Learning (Theory)

1.

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

Hence α_M can be verified as $\frac{1}{M}$.

2. (a)

$$\begin{aligned}\mathbb{E}[\delta_t|S_t=s] &= \mathbb{E}[R_{t+1}+\gamma V(S_{t+1})-V(S_t)|S_t=s]\\ &= \mathbb{E}[R_{t+1}+\gamma V_\pi(S_{t+1})-V_\pi(S_t)|S_t=s]\\ &= \mathbb{E}[R_{t+1}+\gamma V_\pi(S_{t+1})|S_t=s]-V_\pi(s)\\ &= \mathbb{E}[R_{t+1}+\gamma G_{t+1}|S_t=s]-V_\pi(s)\\ &= V_\pi(s)-V_\pi(s)\\ &= 0\end{aligned}$$

(b)

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

3. (a)

$$\begin{aligned}G_t-V_t(S_t) &= R_{t+1}+\gamma G_{t+1}-V_t(S_t)\\ &= R_{t+1}+\gamma G_{t+1}-V_t(S_t)+\gamma V_t(S_{t+1})-\gamma V_t(S_{t+1})\\ &= \delta_t+\gamma(G_{t+1}-V_t(S_{t+1}))\\ &= \delta_t+\gamma(R_{t+2}+\underbrace{\alpha\delta_t}_{\text{immediate correction/reward}}+\gamma G_{t+2}-V_t(S_{t+1}))\\ &= \delta_t+\gamma(\delta_{t+1}+\alpha\delta_t+\gamma(G_{t+2}-V_{t+1}(S_{t+2})))\\ &= \delta_t+\gamma(\delta_{t+1}+\alpha\delta_t)+.....+\gamma^{T-t}(0+\alpha\delta_{T-1})\\ &= \delta_t+\gamma^{T-t}\alpha\delta_{T-1}+\sum_{k=t+1}^{T-1}\gamma^{k-t}\Big(\delta_k+\alpha\delta_{k-1}\Big)\end{aligned}$$

(b)

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

8 Maximization Bias

1. Q-learning: Q(A,R)=1.5, Q(A,L)=2, Q(B,1)=1, Q(B,2)=1, Q(B,3)=2, Q(B,4)=0
   SARSA: Q(A,R)=1.5, Q(A,L)=1, Q(B,1)=1, Q(B,2)=1, Q(B,3)=2, Q(B,4)=0
2. For Q-Learning this problem suffer from maximum bias as maximum reward over all action from state B to terminal is 2 whereas on average it should be 1. But SARSA doesn't suffer from this as all the action are considered equally likely giving an average reward of 1.

3. Double Q-learning has two sets of play, and two estimates of action-value function; Q_1 and Q_2. Q_1(or Q_2) is used to determine the maximizing action $A^* = argmax_a Q_1(a)$ and other Q_2 (or Q_1) is used to provide the estimated value, $Q_2(A^*) = Q_2(argmax_a Q_1(a))$. In the above example, we can divide given the episodes in two sets, say first rewards of episodes of B to Terminal state for all the 4 action is one set and rest another. If say Q_1 selects the action that gives the maximum reward of 2 it is not guaranteed that the Q_2 selecting the same action would yield the same reward of 2. This is evident from the distribution of the rewards for all four action of B towards terminal state(except action 3 in this example but in general this is valid). Thus, the maximization bias is removed by using Double Q-learning as maximum value is not always selected and the values converge to true average state-action value.
4. Q(A,R)=1.5, Q(A,L)=1, Q(B,1)=1, Q(B,2)=1, Q(B,3)=1, Q(B,4)=0

9 Model-based RL

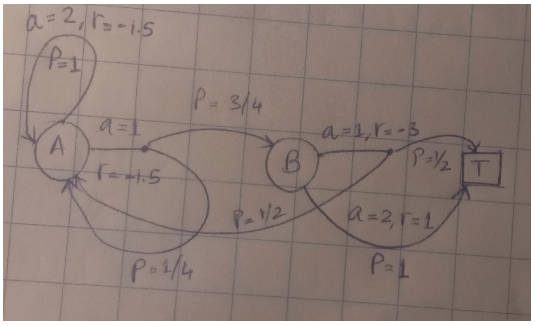
1. 

Figure 1: State transition diagram for the given data.

2. (a) Dyna-Q algorithm is a table based method which assumes the environment is deterministic which is in contradiction with data samples generated following some MDP.
   (b) Instead of making the model deterministic we can make it stochastic by learning Maximum likelihood of (s,a,s') as we did in Figure 1.
   (c) No, the suggested modification will not handle the changing environment because the model will still have the old estimated rewards for a new good path. It is very unlikely that the agent after following too many exploratory moves to will discover the better path and update the model. This can be addressed by updating the model with new information of a better path/action-state function values, as done in Dyna-Q+.
3. (a) There will be no difference (approximately same) Q-values for both the methods in **first episode**. They will have same amount of experience to learn and plan through.
   (b) Q-values will be different after the second episode because planning will generate a more extensive policy using all the states and actions seen so far including states and actions from previous episodes as well while Q-learning operates on the current episode. As result of this Dyan-Q converges faster to optimal values/policy as compared to Q-learning.
   (c) Both the schemes will converge if we continued sampling.

References