CIS 419/519: Applied Machine Learning — Fall 2024

Homework 4

Handed Out: — *Due: 7:59 pm Nov 20*

Name: Richard Zhang

PennKey: zhank24

PennID: 19331985

1 Declaration

- **Person(s) discussed with:** *N/A*
- **Affiliation to the course: student, TA, prof etc.** *student*
- **Which question(s) in coding / written HW did you discuss?** ***N/A***
- **Briefly explain what was discussed.** *N/A*

2 Question 1

1. **"Bob loves cookie"** because we simply choose the word with the highest probability. To be more specific, $P(w_1 \mid \text{Bob})_{\max}$ gives us "loves", and $P(w_2 \mid \text{Bob, loves})_{\max}$ gives us "cookie" as both of them have the highest probability in their conditional probability lists, respectively (0.5 and 0.4).

2. (a) Case 1: "Bob loves cookie":
   We know $P(w_1 = \text{loves} \mid \text{Bob}) = 0.5$ and $P(w_2 = \text{cookie} \mid \text{Bob, loves}) = 0.4$.
   Applying the formula $\ln(P(w_1 \mid \text{Bob})) + \ln(P(w_2 \mid \text{Bob}, w_1))$, we would get:
   $$\ln(0.5) + \ln(0.4) = \ln(0.2) = -1.609$$
   (b) Case 2: "Bob hates cookie":
   We know $P(w_1 = \text{hates} \mid \text{Bob}) = 0.4$ and $P(w_2 = \text{cookie} \mid \text{Bob, hates}) = 0.2$.
   Applying the formula $\ln(P(w_1 \mid \text{Bob})) + \ln(P(w_2 \mid \text{Bob}, w_1))$, we would get:
   $$\ln(0.4) + \ln(0.2) = \ln(0.08) = -2.526$$
   Thus, the answers are -1.609 and -2.526, respectively.

3. **No.** The highest probability choice in the first step doesn't guarantee the highest probability choice in the second step. For example, our greedy sampling strategy gives us "Bob loves cookie", but other sentences like "Bob hates cherry" are more probable.

This is because the likelihood of "Bob loves cookie" is -1.609, and the likelihood of "Bob hates cherry" is $\ln(0.4) + \ln(0.7) = \ln(0.28) = -1.273$. which is more porbable than then sentence generated by the greedy sampling strategy.

4. (a) Case 1: "Bob loves cookie":
   We know $P(w_1 = \text{loves} \mid \text{Bob}) = 0.5$ and $P(w_2 = \text{cookie} \mid \text{Bob, loves}) = 0.4$.
   Applying the formula $\ln(P(w_1 \mid \text{Bob})) + \ln(P(w_2 \mid \text{Bob}, w_1))$, we would get:
   $$\ln(0.5) + \ln(0.4) = \ln(0.2) = -1.609$$

   (b) Case 2: "Bob loves Bob":
   We know $P(w_1 = \text{loves} \mid \text{Bob}) = 0.5$ and $P(w_2 = \text{cookie} \mid \text{Bob, loves}) = 0.25$.
   Applying the formula $\ln(P(w_1 \mid \text{Bob})) + \ln(P(w_2 \mid \text{Bob}, w_1))$, we would get:
   $$\ln(0.5) + \ln(0.25) = \ln(0.125) = -2.079$$

   (c) Case 3: "Bob hates cherry":
   We know $P(w_1 = \text{hates} \mid \text{Bob}) = 0.4$ and $P(w_2 = \text{cherry} \mid \text{Bob, loves}) = 0.7$.
   Applying the formula $\ln(P(w_1 \mid \text{Bob})) + \ln(P(w_2 \mid \text{Bob}, w_1))$, we would get:
   $$\ln(0.4) + \ln(0.7) = \ln(0.28) = -1.273$$

   (d) Case 4: "Bob hates cookie":
   We know $P(w_1 = \text{hates} \mid \text{Bob}) = 0.4$ and $P(w_2 = \text{cookie} \mid \text{Bob, hates}) = 0.2$.
   Applying the formula $\ln(P(w_1 \mid \text{Bob})) + \ln(P(w_2 \mid \text{Bob}, w_1))$, we would get:
   $$\ln(0.4) + \ln(0.2) = \ln(0.08) = -2.526$$

   Therefore, the top two sentences with the highest estimated log-likelihood are case 3 "Bob hates cherry" (-1.273) and case 1 "Bob loves cookie" (-1.609).

3 Question 2

1. (a) Calculating h_1:
   $$s_1^T h_1 = (0.5)(0.7) + (0.2)(0.2) + (0.4)(0.3) + (0.1)(0.1) = 0.52$$

   (b) Calculating h_2:
   $$(0.5)(0.2) + (0.2)(0.7) + (0.4)(0.3) + (0.1)(0.1) = 0.37$$

   (c) Calculating h_3:
   $$(0.5)(0.0) + (0.2)(0.6) + (0.4)(0.4) + (0.1)(0.3) = 0.31$$

(d) Calculating h_4:

$$(0.5)(0.1) + (0.2)(0.1) + (0.4)(0.0) + (0.1)(0.9) = 0.16$$

Thus, the attention score is $\mathbf{[0.52, 0.37, 0.31, 0.16]}$

2. Just plug in the values we got into the formula:

$$\alpha^t = \text{softmax}(E^t) = \left[\frac{e^{s_t^T h_1}}{\sum_{k=1}^{N} e^{s_t^T h_k}}, \ldots, \frac{e^{s_t^T h_N}}{\sum_{k=1}^{N} e^{s_t^T h_k}}\right]$$

$$\alpha^t = \text{softmax}(E^t) = \left[\frac{e^{0.52}}{\sum_{k=1}^{N} e^{s_t^T h_k}}, \frac{e^{0.37}}{\sum_{k=1}^{N} e^{s_t^T h_k}}, \frac{e^{0.31}}{\sum_{k=1}^{N} e^{s_t^T h_k}}, \frac{e^{0.16}}{\sum_{k=1}^{N} e^{s_t^T h_k}}\right]$$

After calculations, we know $e^{0.52} \approx 1.682$, $e^{0.37} \approx 1.448$, $e^{0.31} \approx 1.363$, $e^{0.16} \approx 1.174$.

$$\sum_{k=1}^{N} e^{s_t^T h_k} = 1.682 + 1.448 + 1.363 + 1.174 = 5.667$$

Thus,

$$\alpha_1 = \frac{1.682}{5.667} \approx 0.297,$$
$$\alpha_2 = \frac{1.448}{5.667} \approx 0.255,$$
$$\alpha_3 = \frac{1.363}{5.667} \approx 0.241,$$
$$\alpha_4 = \frac{1.174}{5.667} \approx 0.207$$

Thus, the attention distribution $\boldsymbol{\alpha^t}$ is:

$$\alpha^t = [0.297, 0.255, 0.241, 0.207]$$

3. We just need to use this formula: $a^1 = \sum_{i=1}^{4} \alpha_i h_i$ here:

(a) a_1^1:

$$a_1^1 = 0.297 \cdot 0.7 + 0.255 \cdot 0.2 + 0.241 \cdot 0.0 + 0.207 \cdot 0.1 = 0.28$$

(b) a_2^1:

$$0.297 \cdot 0.2 + 0.255 \cdot 0.7 + 0.241 \cdot 0.6 + 0.207 \cdot 0.1 = 0.403$$

(c) a_3^1:

$$a_1^1 = 0.297 \cdot 0.7 + 0.255 \cdot 0.2 + 0.241 \cdot 0.0 + 0.207 \cdot 0.1 = 0.262$$

(d) a_4^1:

$$a_1^1 = 0.297 \cdot 0.1 + 0.255 \cdot 0.1 + 0.241 \cdot 0.3 + 0.207 \cdot 0.9 = 0.314$$

Thus, the attention output is $a^1 = [0.28, 0.403, 0.262, 0.314]$

4 Question 3

1. **Go to state *e***. Because the discount factor is 1, there is no discounting impact at all. Thus, for f, reaching a is more rewarding than reaching h (100 versus 30). Thus, the optimal action is going to e (the direction toward a).

2. **Go to state *g***.

   (a) Left (to a): $(0.5)^5 \cdot 100 = 3.125$

   (b) Right (to h): $(0.5)^2 \cdot 30 = 7.5$

   Thus, it apparent that doing to right (the direction toward h) is more rewarding. Thus, we go to state g.

3. **The discount factor is 0.67**.

   Just set a function. Let the discount factor be x. We have:

$$x^5 \times 100 = x^2 \times 30$$

$$x \approx 0.6694 \approx 0.67$$

4. (a) State a:

   This is the terminal, so $V(a) = 0$.

   (b) State b:

$$\begin{aligned} V(b) &= max(\text{Left (to } a\text{), Right (to } c\text{)}) && (1)\\ &= max((100 + 1 \cdot V(a)), (0 + 1 \cdot V(c))) && (2)\\ &= max((100 + 0), (0 + 0)) && (3)\\ &= max(100, 0) && (4)\\ &= 100 && (5) \end{aligned}$$

   Thus, $V(b) = 100$.

   (c) State c:

$$\begin{aligned} V(c) &= max(\text{Left (to } b\text{), Right (to } d\text{)}) && (6)\\ &= max((0 + 1 \cdot V(b)), (0 + 1 \cdot V(d))) && (7)\\ &= max((0 + 100), (0 + 0)) && (8)\\ &= max(100, 0) && (9)\\ &= 100 && (10) \end{aligned}$$

   Thus, $V(c) = 100$.

   (d) State d:

$$\begin{aligned} V(d) &= max(\text{Left (to } c\text{), Right (to } e\text{)}) && (11)\\ &= max((0 + 1 \cdot V(c)), (0 + 1 \cdot V(e))) && (12)\\ &= max((0 + 100), (0 + 0)) && (13)\\ &= max(100, 0) && (14)\\ &= 100 && (15) \end{aligned}$$

Thus, $V(d) = 100$.

(e) State e:

$$\begin{aligned} V(d) &= max(\text{Left (to } d\text{), Right (to } f\text{)}) && (16)\\ &= max((0 + 1 \cdot V(d)), (0 + 1 \cdot V(f))) && (17)\\ &= max((0 + 100), (0 + 0)) && (18)\\ &= max(100, 0) && (19)\\ &= 100 && (20) \end{aligned}$$

Thus, $V(e) = 100$.

(f) State f:

$$\begin{aligned} V(d) &= max(\text{Left (to } e\text{), Right (to } g\text{)}) && (21)\\ &= max((0 + 1 \cdot V(e)), (0 + 1 \cdot V(g))) && (22)\\ &= max((0 + 100), (0 + 0)) && (23)\\ &= max(100, 0) && (24)\\ &= 100 && (25) \end{aligned}$$

Thus, $V(f) = 100$.

(g) State g:

$$\begin{aligned} V(d) &= max(\text{Left (to } f\text{), Right (to } h\text{)}) && (26)\\ &= max((0 + 1 \cdot V(f)), (0 + 1 \cdot V(h))) && (27)\\ &= max((0 + 100), (30 + 0)) && (28)\\ &= max(100, 30) && (29)\\ &= 100 && (30) \end{aligned}$$

Thus, $V(g) = 100$.

(h) State h:

This is the terminal, so $V(h) = 0$.

Therefore, the answer is:

$$\begin{aligned} V(a) &= 0\\ V(b) &= 100\\ V(c) &= 100\\ V(d) &= 100\\ V(e) &= 100\\ V(f) &= 100\\ V(g) &= 100\\ V(h) &= 0 \end{aligned}$$

5. (a) State h:

    This is the terminal, so $V(h) = 0$.

   (b) State g:

$$\begin{aligned} V(d) &= max(\text{Left (to } f\text{), Right (to } h\text{)}) && (31)\\ &= max((0 + 1 \cdot V(f)), (30 + 1 \cdot V(h))) && (32)\\ &= max((0 + 0), (30 + 0)) && (33)\\ &= max(0, 30) && (34)\\ &= 30 && (35) \end{aligned}$$

    Thus, $V(g) = 30$.

   (c) State f:

$$\begin{aligned} V(d) &= max(\text{Left (to } e\text{), Right (to } g\text{)}) && (36)\\ &= max((0 + 1 \cdot V(e)), (0 + 1 \cdot V(g))) && (37)\\ &= max((0 + 0), (0 + 30)) && (38)\\ &= max(0, 30) && (39)\\ &= 30 && (40) \end{aligned}$$

    Thus, $V(f) = 30$.

   (d) State e:

$$\begin{aligned} V(d) &= max(\text{Left (to } d\text{), Right (to } f\text{)}) && (41)\\ &= max((0 + 1 \cdot V(d)), (0 + 1 \cdot V(f))) && (42)\\ &= max((0 + 0), (0 + 30)) && (43)\\ &= max(0, 30) && (44)\\ &= 100 && (45) \end{aligned}$$

    Thus, $V(e) = 30$.

   (e) State d:

    It is the same with the step e, so $V(d) = 30$.

   (f) State c:

    It is the same with the step e, so $V(c) = 30$.

   (g) State b:

$$\begin{aligned} V(b) &= max(\text{Left (to } a\text{), Right (to } c\text{)}) && (46)\\ &= max((100 + 1 \cdot V(a)), (0 + 1 \cdot V(c))) && (47)\\ &= max((100 + 0), (0 + 30)) && (48)\\ &= max(100, 30) && (49)\\ &= 100 && (50) \end{aligned}$$

    Thus, $V(b) = 100$.

(h) State a:

This is the terminal, so $V(a) = 0$.

Therefore, the answer is:

$$\begin{aligned} V(a) &= 0 \\ V(b) &= 100 \\ V(c) &= 30 \\ V(d) &= 30 \\ V(e) &= 30 \\ V(f) &= 30 \\ V(g) &= 30 \\ V(h) &= 0 \end{aligned}$$

6. (a) **1 iteration** because all the values have reached their optimal values only after 1 iteration (100). This means the value function is converged (left to right). State b immediately propagates the optimal value to its right states in only 1 iteration.

   (b) **6 iterations** because the highest reward (100) at state a need to propagates through the right states once a time. That's to say, each iteration would let the optimal value to propagate one step further to the right states.

   For example:

   Values after one iteration:

$$\begin{aligned} V(a) &= 0 \\ V(b) &= 100 \\ V(c) &= 30 \\ V(d) &= 30 \\ V(e) &= 30 \\ V(f) &= 30 \\ V(g) &= 30 \\ V(h) &= 0 \end{aligned}$$

   Values after two iterations:

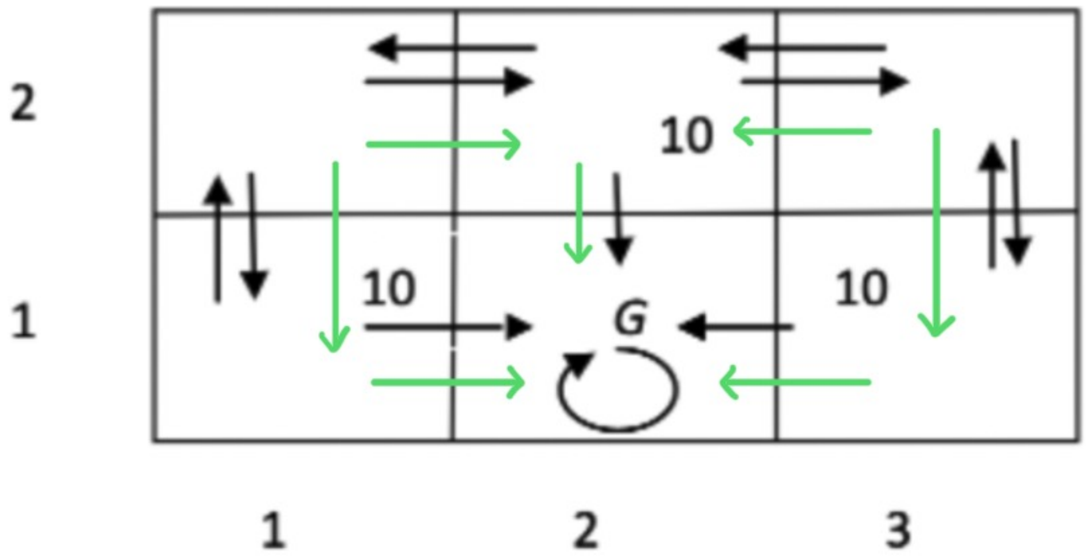
$$\begin{aligned} V(a) &= 0 \\ V(b) &= 100 \\ V(c) &= 100 \\ V(d) &= 30 \\ V(e) &= 30 \\ V(f) &= 30 \\ V(g) &= 30 \\ V(h) &= 0 \end{aligned}$$

...

Values after six iterations (converged):

$$\begin{aligned} V(a) &= 0 \\ V(b) &= 100 \\ V(c) &= 100 \\ V(d) &= 100 \\ V(e) &= 100 \\ V(f) &= 100 \\ V(g) &= 100 \\ V(h) &= 0 \end{aligned}$$

5 Question 4

1. 

2. (a) Calculating $V(2,1)$: We know any action from this state would lead back to itself. The immediate reward of any action from this state is 0. Therefore, it easy to deduce:

$$V(2,1) = R(2,1,a) + \gamma V^*(2,1) = R + 0.8 \cdot 0 = 0 + 0 = 0$$

   (b) Calculating $V(1,1)$:

$$V(1,1) = max(R(1,1,a) + \gamma V^*(1,1)) = R + 0.8 \cdot 0 = 10 + 0 = 10$$

   This is the reward of going right from $(1,1)$.

(c) Calculating $V(2,2)$:

$$V(2,2) = max(R(2,2,a) + \gamma V^*(2,2)) = R + 0.8 \cdot 0 = 10 + 0 = 10$$

This is the reward of going down from $(2,2)$.

(d) Calculating $V(1,2)$:

$$V(1,2) = max(R(1,2,a) + \gamma V^*(1,2)) = R + 0.8 \cdot 10 = 0 + 8 = 8$$

This is the reward of going down or right from $(1,2)$.

Thus, the answer is $\boldsymbol{V(1,2) = 8}$.

3. We know the path is $(1,1) \rightarrow (1,2) \rightarrow (2,2) \rightarrow (3,2) \rightarrow (3,1) \rightarrow (2,1)$.

The TD update rule is:

$$Q_{\text{new}}(s,a) = Q(s,a) + \alpha \left[R(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$$

(a) $(3,1) \rightarrow (2,1)$, reward is 10:

$$Q_{\text{new}}((3,1),\text{left}) = 0 + 0.1\,[10 + 0.8 \times 0 - 0] = 1$$

(b) $(3,2) \rightarrow (3,1)$:

$$Q_{\text{new}}((3,2),\text{down}) = 0 + 0.1\,[0 + 0.8 \times 1 - 0] = 0.08$$

(c) $(2,2) \rightarrow (3,2)$:

$$Q_{\text{new}}((2,2),\text{right}) = 0 + 0.1\,[0 + 0.8 \times 0.08 - 0] = 0.0064$$

(d) $(1,2) \rightarrow (2,2)$:

$$Q_{\text{new}}((1,2),\text{right}) = 0 + 0.1\,[0 + 0.8 \times 0.0064 - 0] = 0.000512$$

(e) $(1,1) \rightarrow (1,2)$:

$$Q_{\text{new}}((1,1),\text{up}) = 0 + 0.1\,[0 + 0.8 \times 0.000512 - 0] = 0.0004096$$

(f) **Thus, the changed Q-values are:**

$$\begin{aligned} Q((3,1),\text{left}) &= 1.0 \\ Q((3,2),\text{down}) &= 0.08 \\ Q((2,2),\text{right}) &= 0.0064 \\ Q((1,2),\text{right}) &= 0.000512 \\ Q((1,1),\text{up}) &= 0.0004096 \end{aligned}$$

6 Question 5

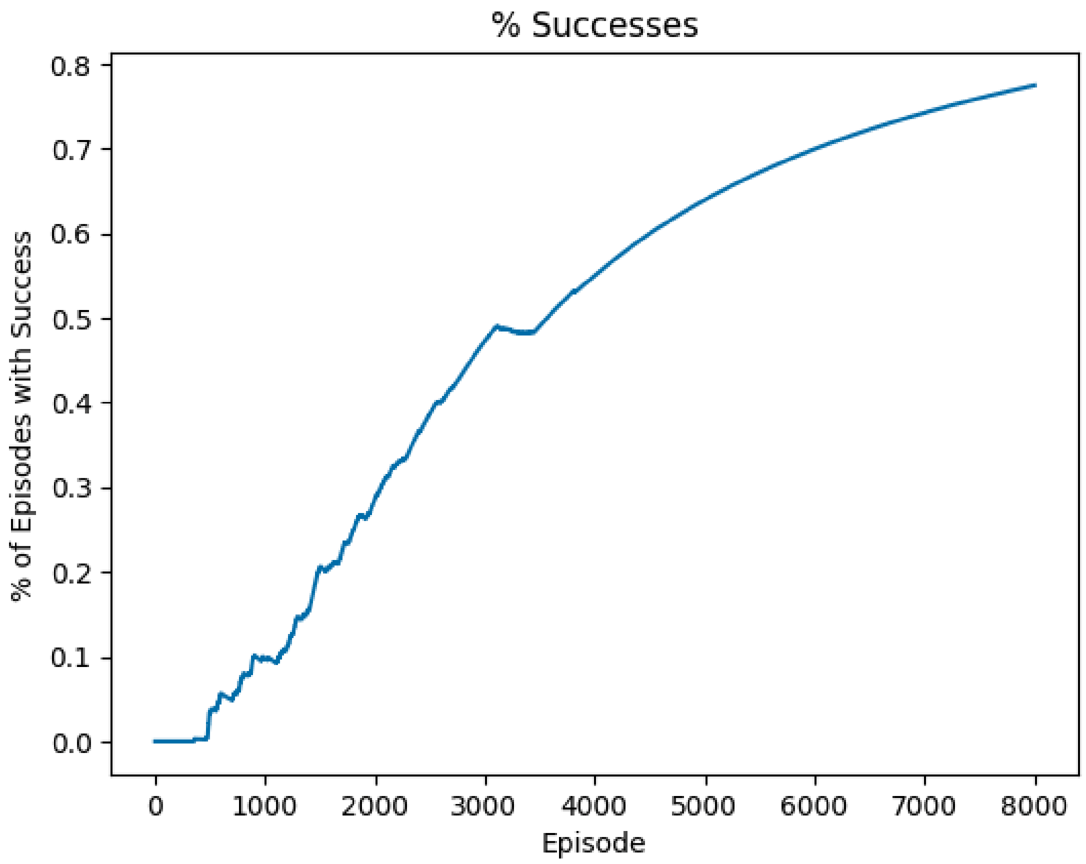
1. **No**, and the reason is very simple: the reward mechanism we set gives a fixed value of +1 for reaching the goal, regardless of the time (steps) taken. Thus, from the perspective of the robot, as long as it reach the goal, it would receive the same amount of award, regardless of how long it takes or how many steps it uses. There is no way to differentiate efficient and inefficient paths so that it doesn't has the motivation to find the most optimal way. Thus, further training wouldn't fix the policy to learn a more efficient route.

   An easy way to tackle this issue is to set a penalizing mechanism (AKA small negative reward) that punishes the robot for every additional step it takes. For example, for every additional step the robot takes, there will be -0.01 reward. In such way, the robot is likely to find more efficient route and, possibly, the most optimal route because it is motivated to do so.

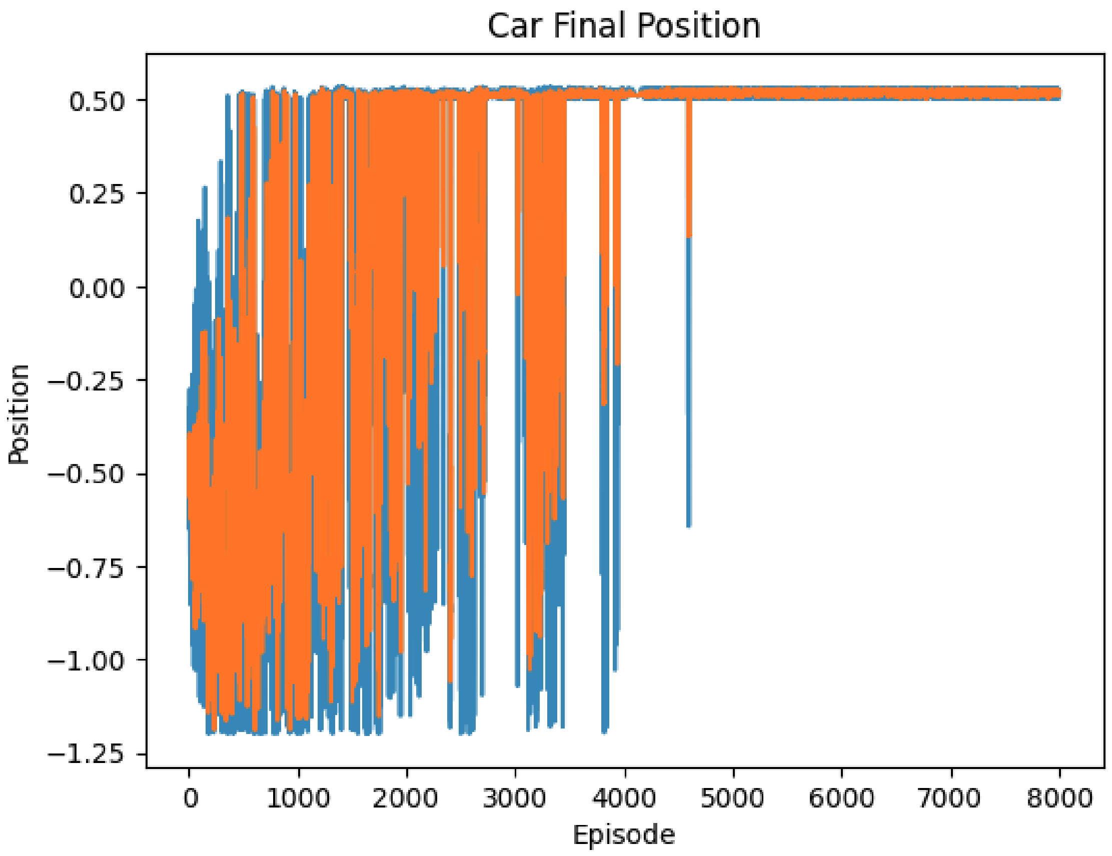
   Another similar method is to set a discount factor that has little impact on the overall reward if the robot takes only a small number of steps to reach the goal but become fairly huge if the robot take many steps to reach the goal. For example, let the discount factor be 0.95 and set the reward formula to:

$$R_{total} = r_1 + r_2 \cdot 0.95 + r_3 \cdot (0.95)^2 + \cdots + r_n \cdot (0.95)^{n-1}$$

7 Python Programming Question 2.7

1. 

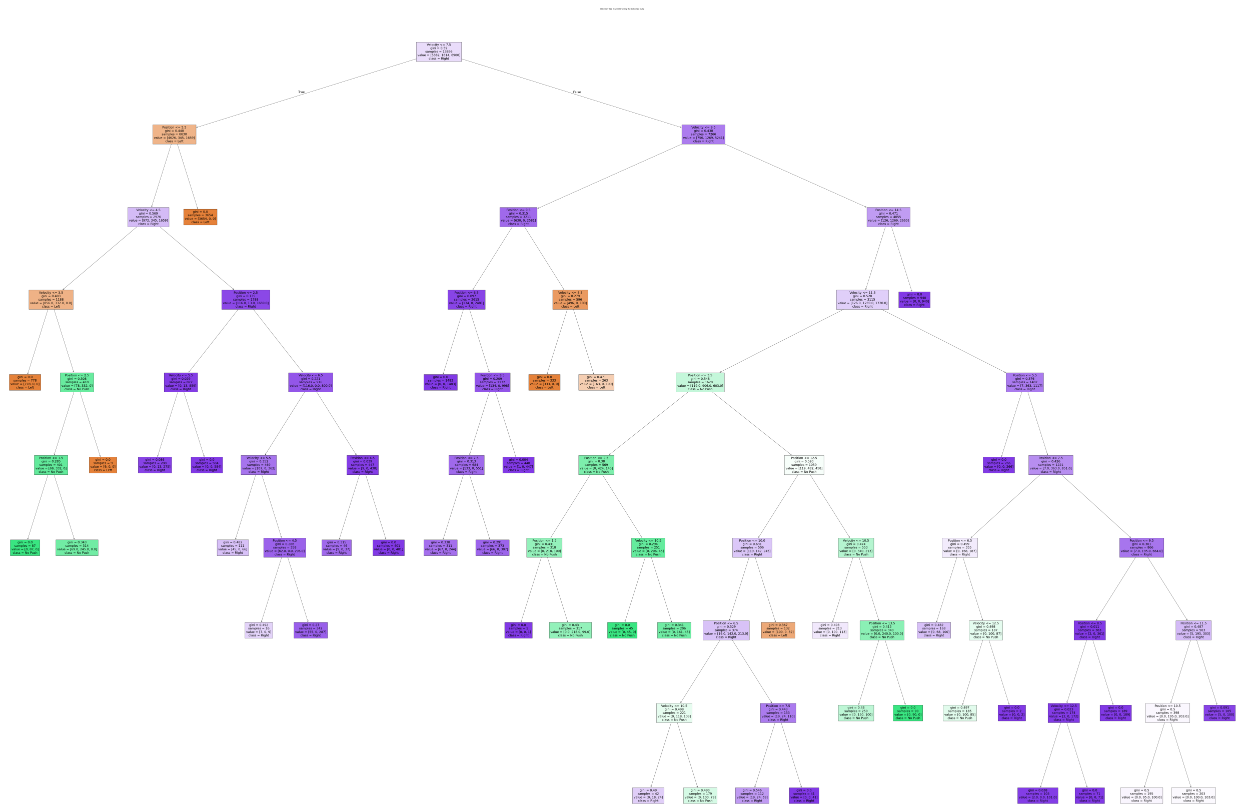
.

2. 
Car Final Position
Position
Episode
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.25
0
1000
2000
3000
4000
5000
6000
7000
8000

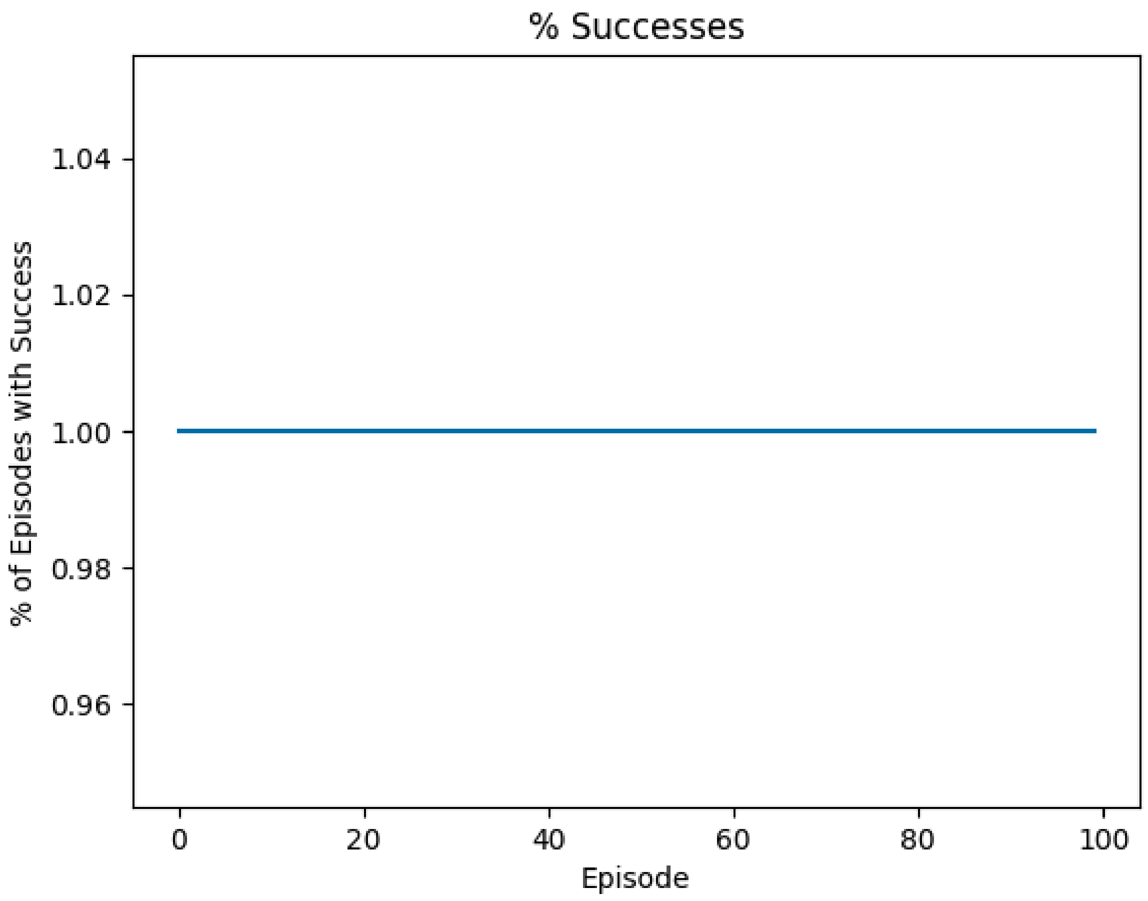
3.

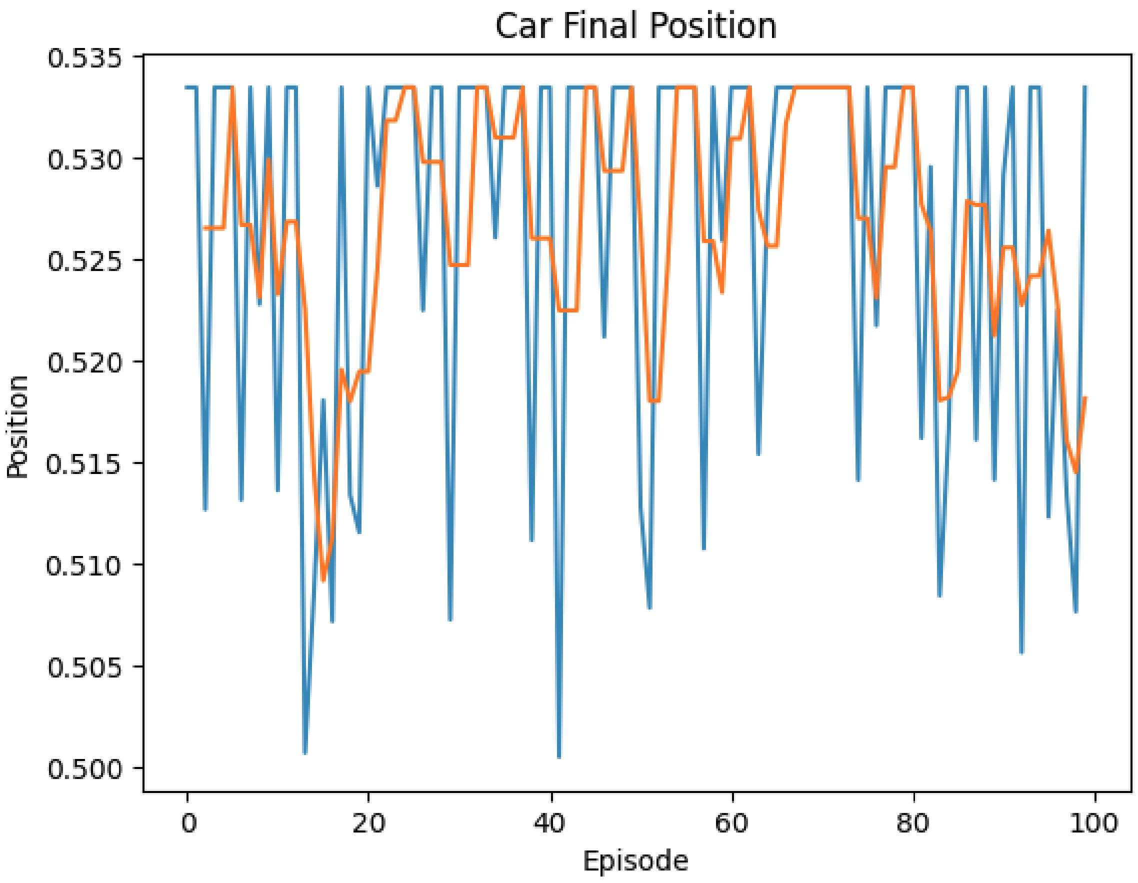
Once [✓ Loop] Reflect

8 Python Programming Question 2.9

1. 

9 Python Programming Question 2.11

1. 



2. In 2.7, because we use Q-learning, we can see the success rate gradually improved over time, reaching around 78% after sufficient training episodes. This steady improvement reflects how Q-learning learns through exploration and exploitation, constantly updating the Q-values to optimize its policy. On thew contrary, in 2.11, the decision tree classifier's success rate remains flat at just 1%, indicating that it is not able to fully mimic the expert policy learned in Q-learning. This poor performance is because the classifier's inability to generalize well to unseen states.

   Speaking of the position graphs, I noticed that in Q-learning, the variability in the car's final positions was fairly huge during the early stage (when the agent is exploring different strategies and make random actions). However, later, the agent consistently reached the goal (position ≥ 0.5) with high reliability and lower variability. This shows reinforcement learning let the agent always refines its strategy through interacting with the environment. On the other hand, the decision tree classifier in 2.11 showed tightly clustered final positions near the goal during testing, but its overall success rate remained very low. To me, this implies while the classifier was able to replicate the expert's actions for familiar states, it struggled significantly with unfamiliar scenarios or edge cases that weren't included in the training data. I also feel like that the decision tree's inability to explore or adapt further makes it less effective in dynamic environments like this one.

   The main reason for this stark difference lies in the underlying approaches. Q-learning dynamically learns and improves its policy through exploration, whereas the decision tree simply mimics the expert policy based on the training data.

I think the main reason for this huge difference is the fundamental difference in thew two methods. Q-learning can dynamically learn and improves its policy through explorations, while the decision tree can only mimics the expert policy based on the training data. I believe that if the expert data lacks diversity, the classifier wouldn't be able to generalize effectively, which would inevitably lead to poor results.