Name: James Yang
Student ID: 2127388

CSE 573 Spring 2022: HW 2

Due 5/27/2022 by 11:59 pm
Total: 100 points

Instructions:

1) The homework should be done individually. Don't forget to write your name.
2) We highly recommend typing your homework, but writing and scanning, or annotating a PDF are also acceptable.
3) Keep your answers brief but provide enough explanations and details to let us know that you have understood the topic.
4) The assignment is due on May 27, 11:59 pm (AoE).
5) You should upload your assignments through gradescope

Topics:	Points
Short Answers	10
Value Iteration	20
Q-Learning	15
Bayes Nets	20
Probability	12
Machine Learning	23

Problem 1. Short Answer[10 pts]

1.1. [3 pts] Briefly explain what “epsilon-greedy” means in Q-learning and discuss what aspects of off-policy learning it aims to improve.

Epsilon-greedy within Q-learning looks to improve the exploration vs. exploitation idea. When exploring randomly, we want to have an epsilon that gives the probability that the agent moves randomly. That’s why the agent switches between learned and random. If we decrease epsilon as time goes on, it will slowly converge to an answer. The reason Q-learning is off policy is because it updates its Q-values using the Q-value of the next state and the greedy action. It estimates the return of the total discounted future reward for a state action. Epsilon greedy is a method of controlling how much exploration and how much exploitation is needed to be done by choosing the greedy policy of which move will give you the highest predicted next state step.

1.2. [4 pts] In practice, we often use approximate Q-learning by learning the following function, $Q(s,a) = g(f_1(s,a), f_2(s,a) \ldots, f_n(s,a))$. What is the advantage of using this approach compared to tabular (or exact) Q-learning?

Exact Q-learning is not a flexible method of learning. The reason behind this is because environments can subtly change and throw off a tabular style of Q-learning where all nodes are labeled with a state and action.

An example of this is a Pacman map where all ghosts and all dots are located in the same spot, but the difference between two scenarios is one dot is missing on the other side of the map. This dot makes has no influence on where Pacman is currently at and what his situation/reward is for the next actions, but in an exact Q-learning scenario it would alter and throw off everything due to this subtle difference.

We want to use the approximate Q-learning method as shown above because it is able to describe a state using a vector of features. It is able to effectively understand environments even when they are subtly different from each other without having it disrupt the entire reward system due to some obsolete discrepancies. It can scale and share environmental rewards that help balance out it’s interpretability of the map.

1.3 [3 pts] Short Answer – Briefly describe a sign of overfitting in Naive Bayes learning, and how it can be avoided.

In general, when we have a high training accuracy and low testing accuracy it is an indicator of overfitting. An example of overfitting in Naïve Bayes could be in predicting numbers within an MNIST dataset. Lets say we use a very small training dataset and a large testing dataset where it contains small sets of squares being activated within a grid representing the mark of a number. If we were to test a uniquely written number that hadn't been seen very much in the training set, it would have a very poorly predicted result. That's not to say that the training dataset needs to contain this type of handwritten number, it's more to say that it needs to train on more types of handwritten numbers to derive a better sense of probability within the marked squares. If a Naïve Bayes model has never seen an action before, it will represent it with a false (0 probabilistic chance).

To avoid this overfitting prediction, we would need to first analyze more training data (randomized, split, etc.) and try to develop a better model to predict off. The model needs to be more generalized, producing a smaller MSE value. Secondly, it needs to have better tuned hyperparameters to ensure that it is weighting marked squares more properly.

Problem 2. Value Iteration (MDPs) [20 pts]

Consider the 101x3 world below. In the start state, the agent has a choice of two deterministic actions, Up or Down, but in the other states the agent always takes the deterministic action, Right. Grayed out parts are not accessible to the agent. You can assume that the ±50 states correspond to t=0.

+50	-1	-1	-1	-1	…	-1	-1	-1	-1(**TERMINAL**)
START									
-50	+1	+1	+1	+1	…	+1	+1	+1	+1(**TERMINAL**)

2.1. [10 pts] Compute the utility of each action as a function of γ.

Util(up) = 50 - $\sum_{i=1}^{100} \gamma^i$

Util (down) = -50 + $\sum_{i=1}^{100} \gamma^i$

2.2. [10pts] Assuming a discounted reward function, for what values of the discount factor γ should the agent choose Up as the initial action?

50 - $\sum_{i=1}^{100} \gamma^i$ = -50 + $\sum_{i=1}^{100} \gamma^i$

100 = $2\sum_{i=1}^{100} \gamma^i$

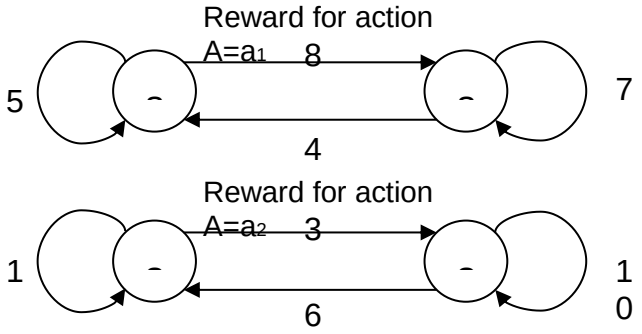
50 = $\sum_{i=1}^{100} \gamma^i$

$\gamma = 0.984$

So when $0 <= \gamma < 0.9844$ it will go up.

Problem 3. Q-Learning (Reinforcement learning/MDPs) [15 pts]

Let's run Q-learning on the following problem with the state space $S = \{s_1, s_2\}$ and the action space $A = \{a_1, a_2\}$. The following state machines show rewards for each state transition and action:



The following table shows the initial Q-values each state and action:

Q-Values			
		Action	
		a_1	a_2
State	s_1	0	0
	s_2	1	1

Table 1. Initial Q-Values

Then, we observe the two following episodes:

Episode 1:	*Episode* 2:
$(previous = s_1, action = a_1, next = s_2)$	$(previous = s_2, action = a_2, next = s_2)$
$s_1, a_1, s_2, 8$	$s_2, a_2, s_2, 10$

We will conduct Q-learning with discount factor γ = 0.5 and learning rate α = 0.1.

3.1. [7 pts] From the initial Q-value table in Table 1, update Q-values given the first episode. What is the updated Q-value table?

		Action	
		a_1	a_2
State	s_1	0.85	0
	s_2	1	1

3.2. [8 pts] **From the Q-value table updated in 3.1**, update Q-values with the second episode. What is the updated Q-value table?

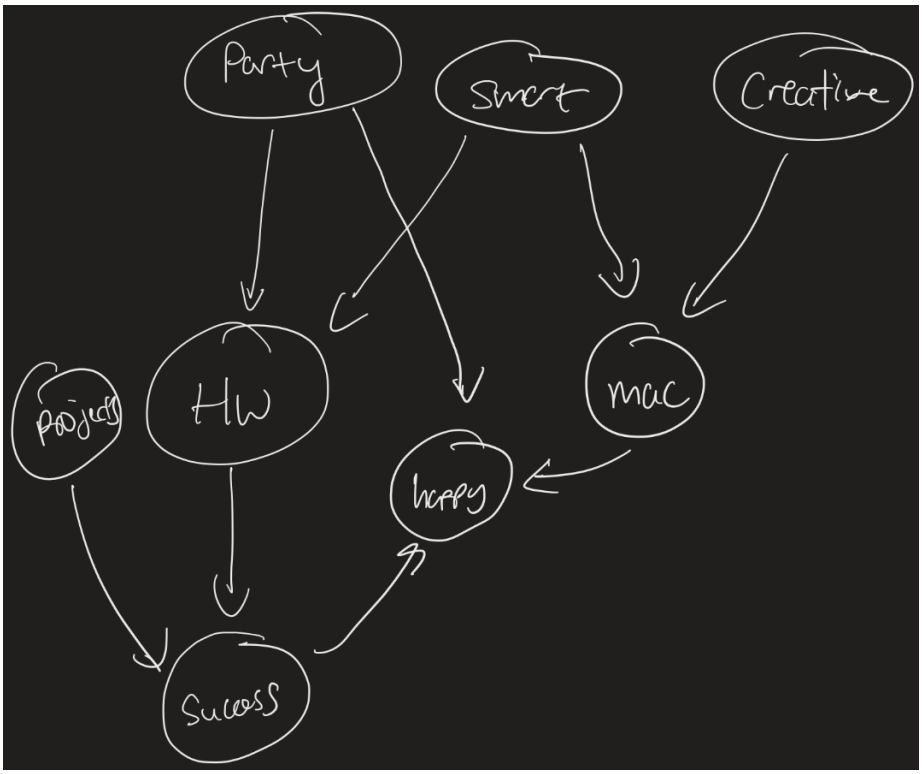
		Action	
		a_1	a_2
State	s_1	0.85	0
	s_2	1	1.95

Problem 4. Bayes Nets [20 pts]

As part of a comprehensive study of the role of 10-601 on people's happiness we have been collecting important data from graduating students. In an entirely optional survey that all students are required to complete, where they were asked about whether they partied, whether they did well on their homeworks and projects, whether they used a Mac, and whether they thought of themselves as smart, creative, happy and successful. After consulting a behavioral psychologist, the following complete set of conditional relationships between the eight binary variables *Party, Smart, Creative, HW, Mac, Project, Success,* and *Happy*:

- Students' homework performance depends only on their partying habits and how smart they are.
- Students' Mac use is related to how smart and creative they are.
- Students' project outcomes depend only on how smart and creative they are.
- A student's success depends on their performance on homeworks and projects alone.
- Finally, students are happy depending on their degree of success, their partying habits, and if they use a Mac!

4.1 [6 pts] Draw a reasonable Bayes Network for the relationships between these variables.

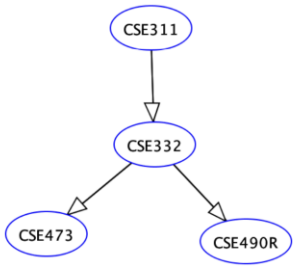


4.2 [6 pts] Write joint distribution as a product of conditional probabilities for the above Bayesian network.

P(creative, smart, party, project, mac, hw, success, happy) =

P(creative) P(smart) P(party) P(project | creative, smart) P(success | hw, project) P(mac | creative, smart) P(hw | smart, party) P(success | project, hw) P(happy | success, mac, party)

4.3 [8 pts] Consider the following Bayesian Network about class prerequisites in UW.



Give reasonable probability tables for all nodes in this subnetwork.

P(CSE 311)

+CSE311	**0.999**
-CSE311	**0.001**

P(CSE 332)

CSE 311	CSE 332	P(CSE 332 | CSE 311)
+311	**+332**	**0.95**
+311	**-332**	**0.05**
-311	**+332**	**0.4**
-311	**-332**	**0.6**

P(CSE 473)

CSE 332	CSE 473	P(CSE 473 | CSE 332)
+332	**+473**	**0.9**
+332	**-473**	**0.1**
-332	**+473**	**0.7**
-332	**-473**	**0.3**

P(CSE 490R)

CSE 332	CSE 490	P(CSE 490 | CSE 332)
+332	**+473**	**0.7**
+332	**-473**	**0.3**
-332	**+473**	**0.5**
-332	**-473**	**0.5**

Problem 5. Probability [12 pts]

From patients admitted to the emergency room, let's assume 6% of patients were admitted due to the infectious disease A. According to the clinical test for the infectious disease A, 85% of infected patients are test-positive while 3% of non-infected patients are test-positive.

5.1. [4 pts] What is the **joint** probability of the clinical test being positive and a patient was infected?

P(pos, A) = P(pos) * P(A) = P(positive | A) * P(A) = (0.85)*0.06 = **0.051**

5.2. [4 pts] What is the conditional probability that the patient was infected if the clinical test was positive?

P(A | pos) = $\frac{\mathrm{P(pos \mid A) * P(A)}}{P(pos)}$ = $\frac{0.85*0.06}{0.85*0.06+0.03*0.94}$ = **0.644**

5.3. [4 pts] What is the conditional probability that the clinical test was not positive but the patient was infected?

P(A | not pos) = $\frac{\mathrm{P(not\ pos \mid A) * P(A)}}{P(not\ pos)}$ = $\frac{.15*0.06}{1-P(pos)}$ = $\frac{.15*0.06}{0.9208}$ = **0.00977**

Problem 6. Machine Learning (Perceptrons/Naive Bayes) [23 pts]

6.1 [12pts] Naive Bayes

In this problem, you will build a Naive Bayes classifier for a Professor's email spam filter at UW - a machine learning algorithm to determine whether they should read a mail or not. To train this classifier model, the following data set of binary-valued features about each email is available, including whether they know the author or not, whether the email is long or short, and whether it has any of several key words, along with a final decision about whether to read it (y = +1 for "read", y = −1 for "discard").

x1 know author?	x2 Is long?	x3 has "research"?	x4 has "grade"?	x5 has "lottery"?	y read?
0	0	1	1	0	-1
1	1	0	1	0	-1
0	1	1	1	1	-1
1	1	1	1	0	-1
0	1	0	0	0	-1
1	0	1	1	1	1
0	0	1	0	0	1
1	0	0	0	0	1
1	0	1	1	0	1
1	1	1	1	1	-1

Note: In case of ties, the email is generally read (+1).

i. [8pt] Compute all the probabilities necessary for a naïve Bayes classifier, i.e., the class probability $p(y)$ and all the individual feature probabilities $p(x_i|y)$, for each class y and feature x_i.

Y	**P(Y)**
+1	2/5
-1	3/5

Y	**P(x1=1|Y)**	**P(x2=1|Y)**	**P(x3=1|Y)**	**P(x4=1|Y)**	**P(x5=1|Y)**
+1	3/4	0	3/4	1/2	1/4
-1	1/2	5/6	2/3	5/6	1/3
Y	**P(x1=0|Y)**	**P(x2=0|Y)**	**P(x3=0|Y)**	**P(x4=0|Y)**	**P(x5=0|Y)**
+1	1/4	1	1/4	1/2	3/4
-1	1/2	1/6	1/3	1/6	2/3

ii. [4pt] Which class would be predicted for x = (0 0 0 0 0)? What about for x = (1 1 0 1 0)?

x = (00000) would give you 1

x = (11010) would give you -1

6.2 [11 pts] Consider using the perceptron algorithm to learn the logical OR function with the training set:

(x_1, x_2, b)	$y*$
(-1,-1,1)	-1
(-1,1,1)	1
(1,-1,1)	1
(1,1,1)	1

Assume the following perceptron definition: $f(x) = sign(w \cdot x)$ where

$$\text{sign}(x) = \begin{cases} +1 & x \geq 0 \\ -1 & x < 0 \end{cases}$$

Updating if we guess incorrectly, i.e., $f(x) \neq y^*$, using the rule:

$$w \leftarrow w + y^* \cdot x$$

(i) [6 pts] Fill out the below table:

Iteration	x	w	$f(x)$	y^*
1	(-1,-1,1)	(0,0,0)	1	-1
2	(-1,1,1)	(1,1,-1)	-1	1
3	(1,-1,1)	(2,0,0)	1	1
4	(1,1,1)	(2,0,0)	1	1
5	(-1,-1,1)	(2,0,0)	-1	-1
6	(-1,1,1)	(2,0,0)	-1	1
7	(1,-1,1)	(3,-1,1)	1	1
8	(1,1,1)	(3,-1,1)	1	1

(ii) [3 pts] Has training converged? Why or why not?

Iteration	x	w	$f(x)$	y^*
9	(-1,-1,1)	(3,-1,1)	-1	-1
10	(-1,1,1)	(3,-1,1)	-1	1
11	(1,-1,1)	(4,-2,2)	1	1
12	(1,1,1)	(4,-2,2)	1	1
13	(-1,-1,1)	(4,-2,2)	1	-1

After calculating 5 iterations, it seems that the weight is still changing and aren't fixed. This would indicate that the training has not converged.

(iii) [2 pts] If we initialized w to a non-zero weight vector would we necessarily converge to the same final weight vector? Why or why not?

No, the reason is because initializing weights to zero is not recommended. When doing so, the derivatives will remain the same for all w's, resulting in neurons to learn the same features for every iteration. If we initialize w to a non-zero weight vector, it might converge to the same final weight vector depending on the points, but in this case it simply would not.