I discussed with my teammate and Jiaqi He
PART-A [5 questions = 40 points]

Question1 [6 points]
Explain overfitting [2 points] and VC dimension [2 points]. Explain the relationship between VC dimension and overfitting. [2 points]

Overfitting is when your model has good performance on your training data, but a bad performance on testing data. Under the same training accuracy, a simpler model is better.
VC dimension is a measure of the complexity of a classification model. It is the maximum number of data points that can be shattered by the model. The VC-dimension of a hypothesis space H is the cardinality of the largest set S that can be shattered by H.
Models with very high VC dimensions are more likely to overfit, because the model will become very complex and learn all the noise from the data. Models with very low VC dimensions are more likely to underfit, because the model did not learn the pattern from the data. Therefore, we should find a balanced VC dimension, so that our model will not overfit or underfit.

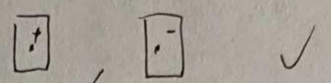
Question2 [4 points]
Can a hypothesis class with a high VC dimension always fit any set of training data perfectly? [2 points] Why or why not? [2 points]

No, because a hypothesis class with a high VC dimension does not mean it takes into account the complexity of the true underlying distribution. Also, as I said in question 1, high VC dimensions models are more likely to overfit. That's why it can not always fit to any set of training data perfectly.

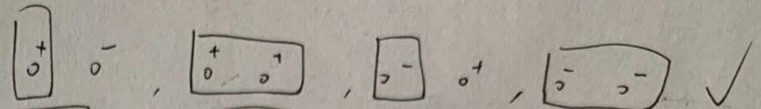
Question3 [10 points]
What is the VC Dimension of a 2D classifier with a rectangle shape? [2 points] Explain with examples (similar to the procedure performed in class). [8 points]

The VC Dimension is 4.

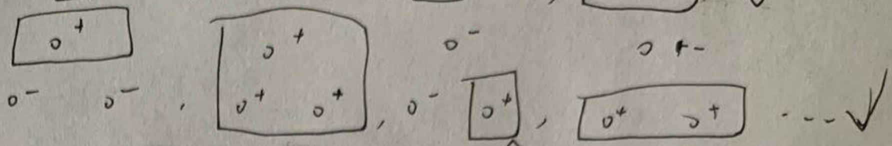
VC = 1



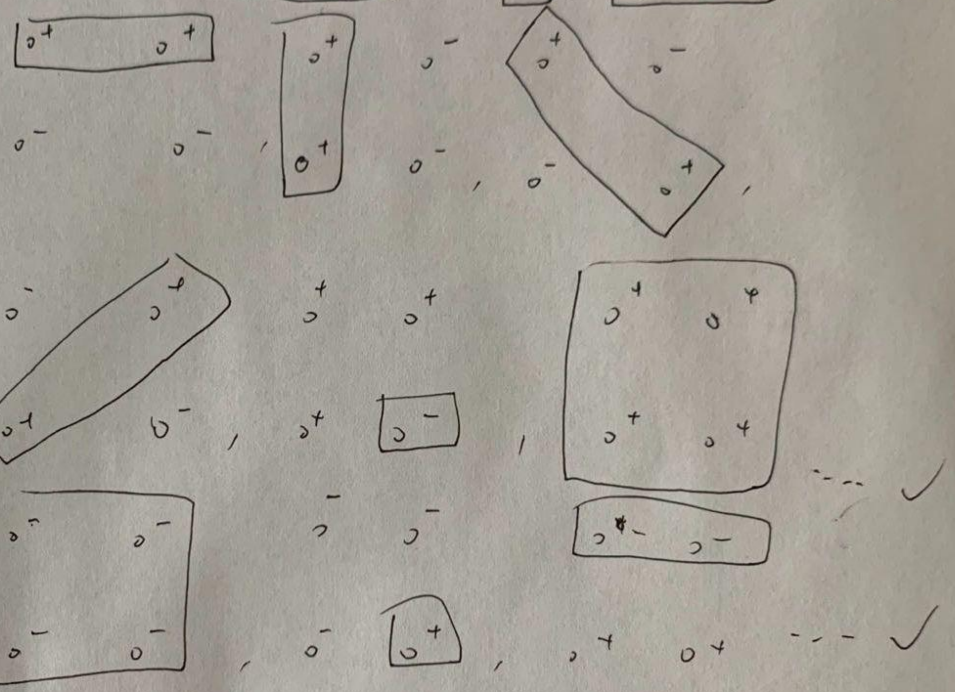
VC = 2



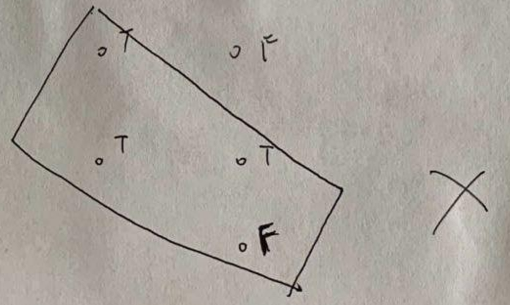
VC = 3



VC = 4

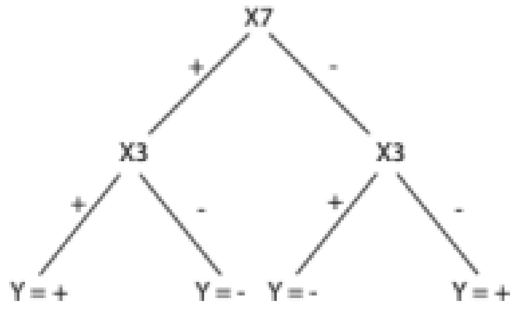


VC = 5

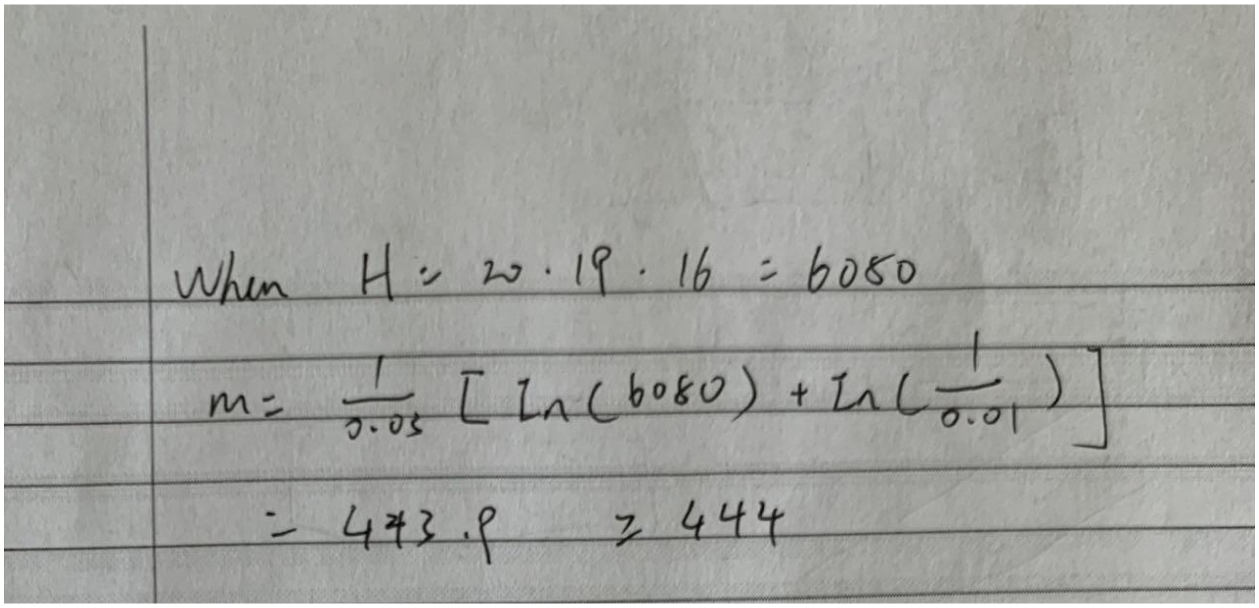


Question4 [10 points]

For the following problem consider a decision tree with boolean inputs ($X_1, \ldots, X_n$) and boolean outputs ($Y \in \{+, -\}$). Consider a regular depth-2 decision tree (4 leaf nodes in total) in which the left and right child of the root are required to test the same attribute. For example:



Consider the training data to be noise-free for target concept (c) which is described by a regular, depth-2 decision tree. How many training examples must you provide the learning algorithm in order to assure that with probability 0.99 the learner will output a tree whose true accuracy is at least 0.97? Assume you have data with 20 features in total (although you believe only two of these twenty will be needed to describe the correct tree).



When $H = 20 \cdot 19 \cdot 16 = 6080$

$$m = \frac{1}{0.05}\left[\ln(6080) + \ln\left(\frac{1}{0.01}\right)\right]$$

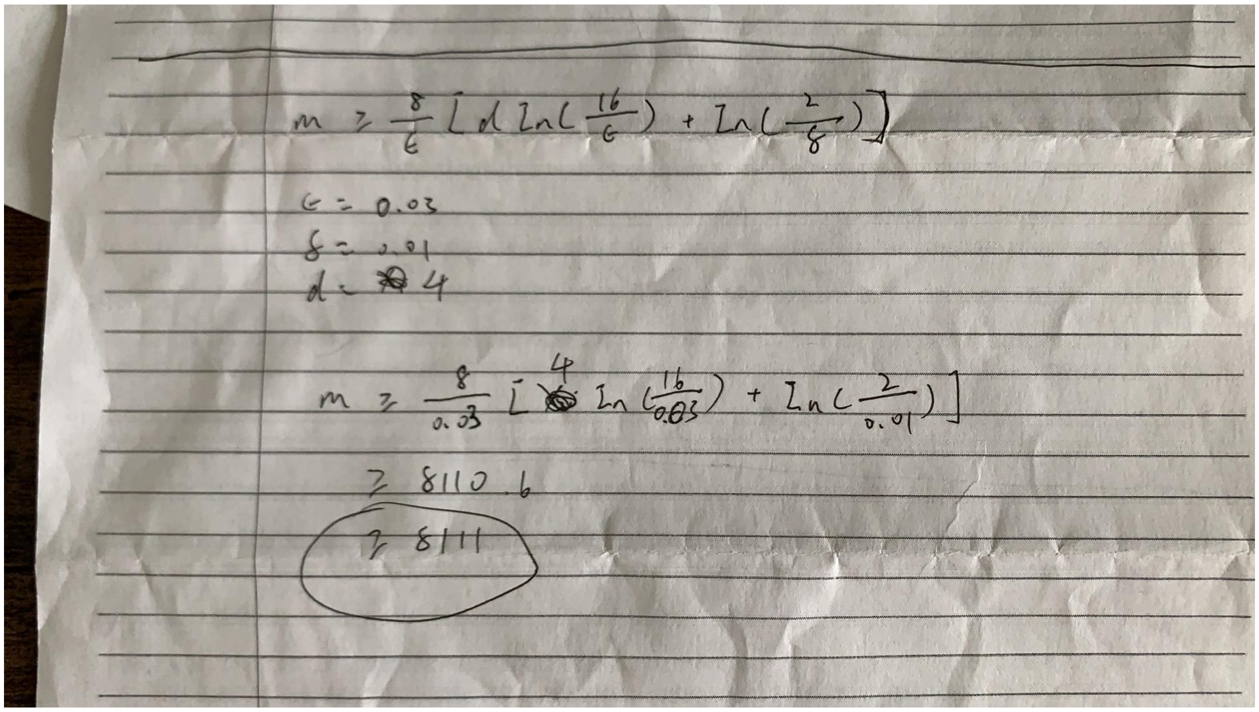
$$= 443.8 \approx 444$$

But now our H is the upper bound, because if we split X7 first and the X3 is the same as if we first split X3 and then X7. Therefore, the H should be smaller than 6080

Question5 [10 points]

Consider the same setup as Question 5 and you modify the algorithm to handle instances that have real-valued attributes instead of boolean attributes (So you allow each decision tree node to perform a Boolean threshold test of the form $X_i > a$ where a is allowed to take on any real value). The tree is similarly constrained such that the two second-level nodes, must both test the same attribute and use the same threshold.

For this modification, How many training examples must you provide the learning algorithm in order to assure that with probability 0.99 the learner will output a tree whose true accuracy is at least 0.97? In this case, assume that each example has only two attributes, so the tree will end up using both. You can still assume that the target concept (c) is in the new hypothesis space.



$$m \geq \frac{8}{\epsilon}\left[d \ln\left(\frac{16}{\epsilon}\right) + \ln\left(\frac{2}{\delta}\right)\right]$$

$\epsilon = 0.03$

$\delta = 0.01$

$d = 4$

$$m \geq \frac{8}{0.03}\left[4 \ln\left(\frac{16}{0.03}\right) + \ln\left(\frac{2}{0.01}\right)\right]$$

≥ 8110.6

$\boxed{\geq 8111}$

PART-B [3 questions = 10 points]

Question1 [6 points].

How does the value iteration algorithm work [3 points], and what are its basic steps [3 points]?

The value iteration is used to calculate the optimal value function for MDP. Given a policy π: S -> A, define $V^\pi(s) = E[\sum_{t=0}^{\infty} \gamma^t r_t]$ assuming action sequence chosen according to π, starting at state s. Our goal is to find the optimal policy. For any MDP, that optimal policy exists. In the end, our value function satisfies Bellman equation:

$$V^\pi(s) = R(s) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V^\pi(s').$$

$$V^*(s) = R(s) + \max_{a \in A} \gamma \sum_{s' \in S} P_{sa}(s') V^*(s').$$

The basic steps are: initialization which initializes V(s) := 0. Bellman update which uses Bellman equation to update the value function for each state in MDP. Repeat until convergence.

Question2 [2 points]

How do you choose the initial values for the state value function in the value iteration algorithm?

We can set our initial value for state value to 0, then we will gradually calculate the reward for each state.

Question3 [2 points]
What are some limitations of value iteration in reinforcement learning?

Value iteration required to know p(s'|s,a), transition probabilities. However, in some real life applications, we do not know transition probabilities. Then we will use action-value function Q(s,a)

PART-C [9 questions = 30 points]

Refer to the Jupyter Notebook (IPYNB file) for further instructions on Deep-Q-learning.

You are expected to work through it and understand the code first (more like a code play-around notebook). This assignment requires no code parts to be filled.