CS 5350/6350, DS 4350: Machine Learning Spring 2024

Homework 4

Handed out: March 15, 2020
Due date: March 29, 2020

1 PAC Learnability of Depth Limited Decision Trees [30 points]

amsmath

(a) [2 points] What is $S_n(0)$? That is how many trees of depth 0 exist?

Ans: The number of trees with depth 0 is two, i.e., trees with all possible label values. In this case, there are two labels, 0 1.

(b) [3 points] What is $S_n(1)$? That is, with n features, how many trees of depth 1 exist?

Ans: The number of trees with depth 1 having n features is $4n$.

A tree with $depth = 1$ will have only one node, and there are n features. Thus, there are possible n ways to select the root node for the tree. Likewise, there will be two labels for the root node, each leaf with two possible values.

Total possible tree = # of nodes $*$ $\prod$ possible leaf values $= n * 2 * 2 = 4n$

(c) [4 points] Suppose you know the number of trees with depth i, for some i. This quantity would be $S_n(i)$ using our notation. Write down a recursive definition for $S_n(i+1)$ in terms of n and $S_n(i)$.
For this expression, you can assume that we are allowed to the use same feature any number of times when the tree is constructed.

Ans: Assumed same feature can be reused for any node. This avoids selection of node and order of selection.

# of possible trees = # possible of nodes $* \prod$ possible values at each leaf

# of nodes with depth $i = 2^i - 1$
# of possible nodes with depth $i = n^{2^i-1}$
# of leaf nodes with depth $i = 2^i$
# of possible values for a leaf = 2. (0 or 1)
# of possible values for all leaf = 2^{2^i}

$S_n(i)$: # of possible trees = $n^{2^i-1} * 2^{2^i}$

Similarly, $S_n(i+1) = n^{2^{i+1}-1} * 2^{2^{i+1}}$

$\frac{S_n(i+1)}{S_n(i)} = \frac{n^{2^{i+1}-1} * 2^{2^{i+1}}}{n^{2^i-1} * 2^{2^i}} = \frac{n^{2.2^i-1} * 2^{2^{i+1}-2^i}}{n^{2^i-1}} = \frac{n^{2^i-1}.n^{2^i} * 2^{2^i}}{n^{2^i-1}} = \frac{n.n^{2^i} * 2^{2^i}}{n} = n.n^{2^i-1} * 2^{2^i}$

$S_n(i+1) = n * S_n(i)^2$

(d) [6 points] Recall that the quantity of interest for PAC bounds is the log of the size of the hypothesis class. Using your answer for the previous questions, find a closed form expression representing $\log S_n(k)$ in terms of n and k. Since we are not looking for an exact expression, but just an order of magnitude, so you can write your answer in the big O notation.

Ans:

$S_n(i) = n^{2^i-1} * 2^{2^i}$

$log(|H|) = log(S_n(i)) = log(n^{2^i-1} * 2^{2^i})$

$log(|H|) = log(n^{2^i-1}) + log(2^{2^i})$

$log(|H|) = (2^i - 1) * log(n) + log(2^{2^i})$

$log(|H|) = \mathcal{O}(\log n)$

1. Next, you will use your final answer from the previous question to state a sample complexity bound for decision trees of depth k.

   (a) [3 points] With finite hypothesis classes, we saw two Occam's razor results. The first one was for the case of a consistent learner and the second one was for the agnostic setting. For the situation where we are given a dataset and asked to use depth-k decision trees as our hypothesis class, which of these two settings is more

appropriate? Why?

Ans: Agnostic Learning

We know that the size of hypothesis classes for all boolean decision trees is larger than the depth limited decision trees. Thus, by increasing our hypothesis space, we have a higher chance of generalization. In other words, there exists a hypothesis in H whose training error will be ϵ away from the true error.

(b) [4 points] Using your answers from questions so far, write the sample complexity bound for the number of examples m needed to guarantee that with probability more than $1 - \delta$, we will find a depth-k decision tree whose generalization error is no more than ϵ away from its training error.

Ans:

$m \geq \frac{1}{2.\epsilon^2}[log(|H|) + log(\frac{1}{\delta})]$

$m \geq \frac{1}{2.\epsilon^2}[log(n^{2^k-1} * 2^{2^k}) + log(\frac{1}{\delta})]$

2. [4 points] Is the set of depth-k decision trees PAC learnable? Is it efficiently PAC learnable?

Ans: Yes, PAC Learnable. But it is not an efficient PAC learnable because there is compute problem and finding a decision tree is an NP-Hard problem as the leaf nodes are exponentional.

3. [4 points] Suppose the number of features we have is large and the depth limit we are considering is also large (say, in the thousands or more). Will the number of examples we need be small or large? Discuss the implications of the sample complexity bound from above.

Ans: The number of examples required when the number of features and depth limits are also large is very high. The required examples grow exponentially with depth, and the computation required to find a true decision tree is very high. Finding a decision tree is an NP-Hard problem as the leaf nodes are exponential.

2 Shattering [15 points, for 6350 students]

Ans:

Shattering is defined as: A set S of examples is shattered by a set of functions H if for every partition of the examples in S into positive and negative examples there is **a function in H** that gives exactly these labels to the examples.

Let function from H_n be $-$ of length n.

For $n = 1$, the X_1 and H_n would be $X_1 = 0, 1$, and $H_n = -$,

X_1	Label	$H_n = -$
0	+	0
0	-	1
1	+	1
1	-	0

For $n = 2$, the X_1 and H_n would be $X_1 = 00, 01, 10, 11$, and $H_n = --$,

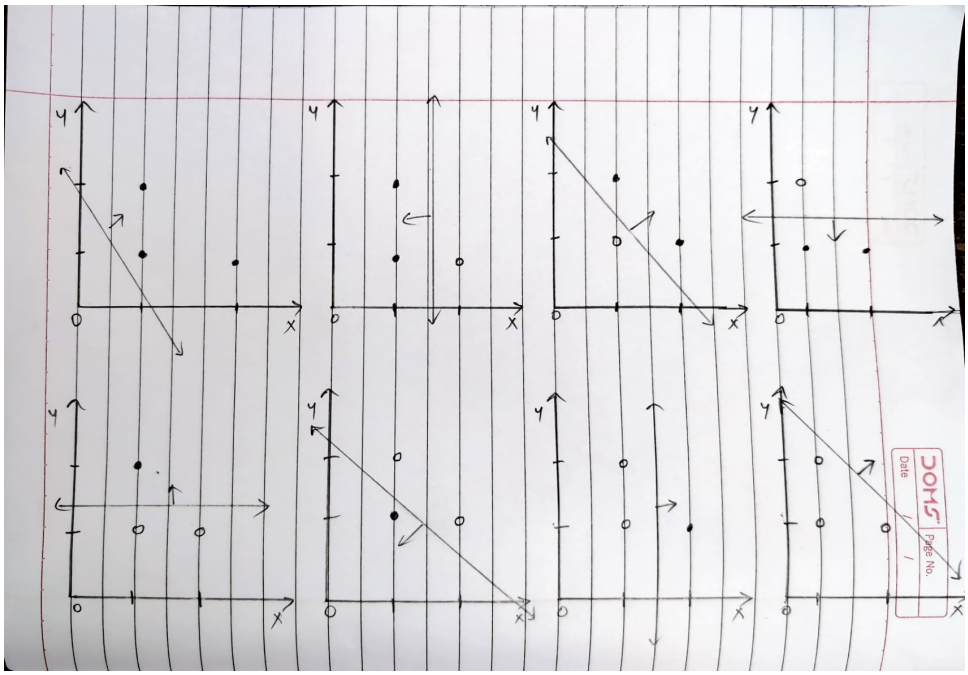
X_1	Label	$H_n = --$
00	+	00
01	+	01
10	+	10
11	+	11
00	-	11
01	-	10
10	-	01
11	-	00

Thus, H_n clearly shatters X_n.

3 VC Dimension [45 points]

1. [5 points] Assume that the three points below can be labeled in any way. Show with pictures how they can be shattered by a linear classifier. Use filled dots to represent positive classes and unfilled dots to represent negative classes.

   Ans: There are 8 cases with 3 dots, which can be either positive or negative, and all 8 cases can be perfectly shattered by a linear classifier, as shown in the figure below.



2. **VC-dimension of axis aligned rectangles in $\mathbb{R}^d$:**

   (a) [10 points] Show that the VC dimension of H^2_{rec} is 4.
   Ans:
   Definition: The VC dimension of hypothesis space H over instance space X is the size of the largest finite subset of X that is shattered by H.

   • If there exists any subset of size d that can be shattered, $VC(H) \geq d$.
   – Even one subset will do.

   • If no subset of size d can be shattered, then $VC(H) < d$.

   Assumed, the subset of the samples to be perfectly axis aligned points. An example of this subset is set of points of same label i.e., all positive or all negative. Such a subset is said to be perfectly shattered when we can draw a box when all points are positive and an empty box when all points are negative.

   Imagine, 4 points in a $2D$ space, aligned to X and Y axis. As mentioned earlier, the set of 4 points belong to sample space when the 4 points are axis aligned and all 4 points are either positive or negative. We can shatter this example by drawing box containing 4 axis aligned points when positive and by drawing box not containing 4 axis aligned points when negative.

   5 or 6 or 7 points, will not fit into our samples space. Consider 8 points, with 4 points +ve and another 4 points -ve. We cannot draw a axis aligned rectangle

box to shatter the 8 points.

Thus, VC dimension of H^2_{rec} is 4.

(b) [10 points] Generalize your argument from the previous proof to show that for d dimensions, the VC dimension of H^d_{rec} is $2d$.
Ans:
Let there be an arrangement of $2d$ points that perfectly shattered by H^d_{rec}. There will be hyperplane rectangle with d dimensions to cover all the labelling of these points.

When there are $2d+1$ points, let the additional point be present at the center of hyperplane rectangle with an opposite label. We will not be able draw any plane that fits all $2d$ points and this center points. $2d+1$ makes the hyperplane rectangle unshatterable.

Thus, VC dimension of H^d_{rec} is $2d$.

3. In the lectures, we considered the VC dimensions of infinite concept classes. However, the same argument can be applied to finite concept classes too. In this question, we will explore this setting.

(a) [10 points] Show that for a finite hypothesis class $\mathcal{C}$, its VC dimension can be at most $\log_2(|\mathcal{C}|)$. (Hint: You can use contradiction for this proof. But not necessarily!)
Ans:
Occam's razor when finite samples space:
$m_1 \geq \frac{1}{\epsilon}[log(|C|) + log(\frac{1}{\delta})]$

Occam's razor when infinite samples space with VC dimension:
$m_2 \geq \frac{1}{\epsilon}[8 * VC(H)log(\frac{13}{\epsilon}) + 4 * log(\frac{2}{\delta})]$

Assumed, since the number of examples in infinite samples space to be finite space. Then let m_1 be highest possible samples space such that $m_2 \leq m_1$.

$8 * VC(H)log(\frac{13}{\epsilon}) + 4 * log(\frac{2}{\delta})] \leq [log(|C|) + log(\frac{1}{\delta})]$

Removing the constants and simplifying,

$VC(H) \leq log(|C|)$.

Thus, VC dimension can be at most $\log_2(|\mathcal{C}|)$.

(b) [5 points] Find an example of a class $\mathcal{C}$ of functions over the real interval $X = [0, 1]$ such that $\mathcal{C}$ is an **infinite** set, while its VC dimension is exactly one.

Ans:

Consider class of functions to be left-bound function $X = a$, where $a = [0, 1]$.

Clearly there are infinite possible real values for a, and thereby infinite class of left-bound functions.

Consider 1 point; there is at least one function that shatters all possible labelings of these points. As seen in the class.

When there are 2 points of opposite labels, a left bound function cannot shatter these 2 points.

Thus, class of functions is infinite but VC dimension is exactly one.

(c) [5 points] Give an example of a **finite** class $\mathcal{C}$ of functions over the same domain $X = [0, 1]$ whose VC dimension is exactly $\log_2(|\mathcal{C}|)$.

Ans:

Consider class of functions to be all possible binary decision trees. Then # of inputs from domain $X = [0, 1]$ is 2^n.

From Occam's razor, the number of samples required is $log(|C|)$.

We know that, there is a decision tree that can fully shatter possible labelings in this domain.

Since the minimum samples required in the space is $log(|C|)$. Then, there must be at least $log(|C|)$ points that will be shattered by a decision tree.

Thus, class of finite functions has VC dimension of $log(|C|)$.

4 Extra Credit - Decision Lists [25 points]

1. [8 points] Show that if a concept c can be represented as a k-decision list so can its complement, $\neg c$. You can show this by providing a k-decision list that represents $\neg c$, given $c = \{(c_1, b_1), \ldots, (c_l, b_l), b)$.

   Ans:

   Concept class c_1:

A *k-decision list* over the variables $x_1, \ldots, x_n$ with an ordered sequence $L = (c_1, b_1), \ldots, (c_l, b_l)$ and a bit b, can be represented as $c_1 = \{(c_1, b_1), \ldots, (c_l, b_l), b)$.

Concept class c_2:

A *k-decision list* over the variables $x_1, \ldots, x_n$ with the same ordered sequence $L = (c_1, \neg b_1), \ldots, (c_l, \neg b_l)$ and a bit , can be represented as $c_2 = \{(c_1, \neg b_1), \ldots, (c_l, \neg b_l), \neg b)$.

Clearly, c_1 & c_2 represent *k-decision list* with bits b complemented at each sequence L.

Thus, $c_1 = \neg c_2$. Making c_2 as complement of c_1 in the same space.

2. [9 points] Use Occam's Razor to show:
For any constant $k \geq 1$, the class of k-decision lists is PAC-learnable.

Ans:

Assume the same sequence can be reused for any node. This avoids the sequence selection to the node and the order of sequence selection. But it should not impact the output as it would return b when the first sequence is hit.

Let d be arbitrary depth of the decision list.

# of possible values of each node = 3 (x, $\neg x$, and 1)

# of conjunctions with n variables = 3^n

# of possible ways to pick *k-decision list* = 3^{n^k}

# of possible ways to pick *k-decision list* at depth $d = d * 3^{n^k}$

# of possible values at leaf = 2

# of possible leaf values at depth $d = 2^d$

Total possible k-lists at depth d = # of nodes $*$ possible leaf values $= d * 3^{n^k} * 2^d$

Occam's razor: $m \geq \frac{1}{\epsilon}[log(|H|) + log(\frac{1}{\delta})]$

Solving only for $log|H|$,

$log(|H|) = log(d * 3^{n^k} * 2^d)$

$log(|H|) = log(3^{n^k}) + log(d * 2^d)$

$log(|H|) = \mathcal{O}(n^k)$

Polynomial and PAC Learnable.

3. [8 points] Show that 1-decision lists are a linearly separable functions. (Hint: Find a weight vector that will make the same predictions a given 1-decision list.)

   Ans:

   A 1-*decision list* can be represented as $c = \{(c_1, b_1), b)\}$.

   A linearly separable function $y = w_1 * x_1 + b$.

   Let the linear function be $y = x_1$.

   The 1-*decision list* would be $c = \{(c_1, 1), 0\}$

Input x_1 or c_1	y: Linear function	c: Decision List
1	1	1 c_1 *bit*
0	0	0 *default bit*

   Thus, decision-list are linearly separable functions.