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

Homework 4

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

General Instructions

- You are welcome to talk to other members of the class about the homework. I am more concerned that you understand the underlying concepts. However, you should write down your own solution. Please keep the class collaboration policy in mind.
- Feel free discuss the homework with the instructor or the TAs.
- Your written solutions should be brief and clear. You need to show your work, not just the final answer, but you do *not* need to write it in gory detail. Your assignment should be **no more than 10 pages**. Every extra page will cost a point.
- Handwritten solutions will not be accepted.
- The homework is due by midnight of the due date. Please submit the homework on Canvas. You should upload one file: a PDF report with answers to the questions below.
- Some questions are marked **For 6350 students**. Students who are registered for CS 6350 should do these questions. Of course, if you are registered for CS 5350 or DS 4350, you are welcome to do the question too, but you will not get any credit for it.

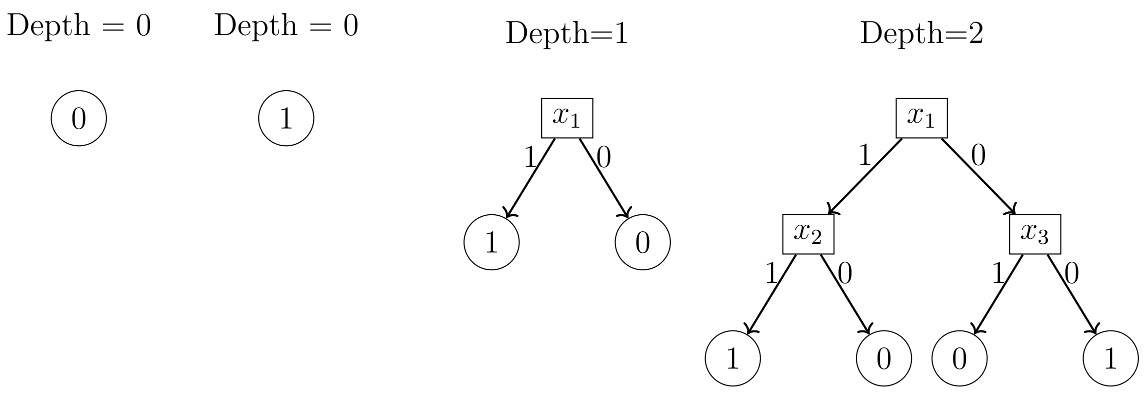
Important. Do not just put down an answer. We want an explanation. No points will be given for just the final answer without an explanation. You will be graded on your reasoning, not just on your final result.

Please follow good proof technique; what this means is if you make assumptions, state them. If what you do between one step and the next is not trivial or obvious, then state how and why you are doing what you are doing. A good rule of thumb is if you have to ask yourself whether what you are doing is obvious, then it is probably not obvious. Try to make the proof clean and easy to follow.

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

In this question, you will be showing that depth limited decision trees are PAC learnable.

Suppose we have a binary classification problem with n Boolean features that we seek to solve using decision trees of depth k. For this question assume trees are complete, meaning each node (other than the leaf nodes) has exactly two children. The figure below shows some examples of such trees and their depths.



1. Since decision trees represent a finite hypothesis class, the quantity of interest is the number of such trees—i.e., trees with depth k over n features. Suppose we use $S_n(k)$ to denote the number of the number of trees with depth exactly k if we have n features.

   The following questions guide you through this counting process. Recall that each answer should be accompanied with an explanation.*If you simply write the final answer, you will not get any points.* (**Please see the note at the end of this questions for further clarification**3).

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

   2, if we have a depth 0 tree, that means there exists only a single node. That node can take 2 values, either 1 or 0.

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

   n*2^2, This is because we only have 1 decision node, we can pick any of the n values for that decision node. This tree has 2 leaf nodes, each leaf node can 1 of 2 values, so that's 2^2.

   (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.

   so our base case in this situation is 2, because a tree with depth 0 (simplest tree) has 2 options. Next we notice that each i+1 tree is the i tree twice with a root node (attribute box) on top connecting them. The root node adds n times extra variants to the tree, doubling the tree adds $S_n(i)$ times extra variance as well. So $S_n(i+1) = S_n(i)^2 * n$

(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.

We first start by writing a sequence of examples where k goes from 0 to 4 and look for patterns
$S_n(0) = n^0 * 2$
$S_n(1) = n^1 * 2^2$
$S_n(2) = n^3 * 4^2$
$S_n(3) = n^7 * 8^2$
$S_n(4) = n^{15} * 16^2$
we'll split the n terms from the powers of 2, and well first start with the powers of 2. we notice that with each iteration the base of the equation increases by 2 and the power stays the same. Its simple to see that the equation is $(2^k)^2$

Next we look at the n terms. each iteration the power that n is raised to increases as $2^k - 1$ this means we can write this portion of the equation as n^{2^k-1}

Putting this all together we get O$(n^{2^k-1} * (2^k)^2)$ as the closed form solution.

2. 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?

lets evaluate our choices, first for the consistent learner. A decision tree can also fit the data perfectly, This is how a decision tree could work for a consistent learner. Though this assumes that our features were picked essentially perfectly and that our data has no noise.
On the other hand for agnostic learning this doesn't make the assumption that the true concept is in the hypothesis class which is a more realistic assumption. For that reason I will choose agnostic learning.

(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.

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

3. [4 points] Is the set of depth-k decision trees PAC learnable? Is it efficiently PAC learnable?

   if we rearrange the equation we get

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

   here we see that the complexity of a decision tree is exponential in terms of the depth so it isn't PAC learnable

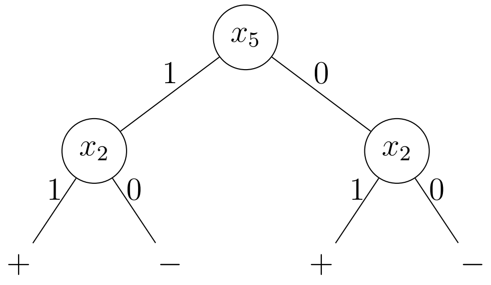
4. [4 points] Suppose the number of features we have is large and the depth limit we are considering is also large (say, in the thousands or more). Will the number of examples we need be small or large? Discuss the implications of the sample complexity bound from above.

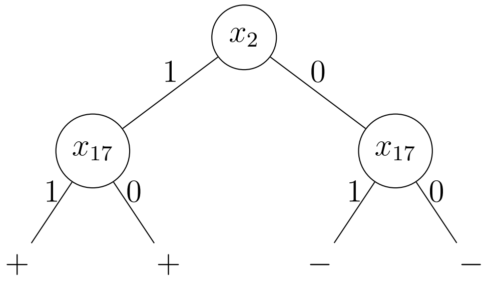
   The number of examples needed will be large
   Considering the sample complexity from above lets explore what a large n and a large k does.

   Large N: the sample complexity above suggests a logarithmic relationship with N. This means that N can get very large and the sample complexity won't increase too much

   Large K: the sample complexity above suggests an exponential relationship with K. This means that if K starts to get large, the number of examples needed goes up very fast.

NOTE: In this question we are counting trees that are structurally different instead of functionally different. As an exercise, you can confirm that the following two trees are structurally different but equal in terms of the label they assign to any example, namely the trees are functionally equivalent.





2 Shattering [15 points, for 6350 students]

Suppose we have a set X_n consists of all binary sequences of a length n. For example, if $n = 3$, the set would consist of the eight elements {`000, 001, 010, 011, 100, 101, 110, 111`}.

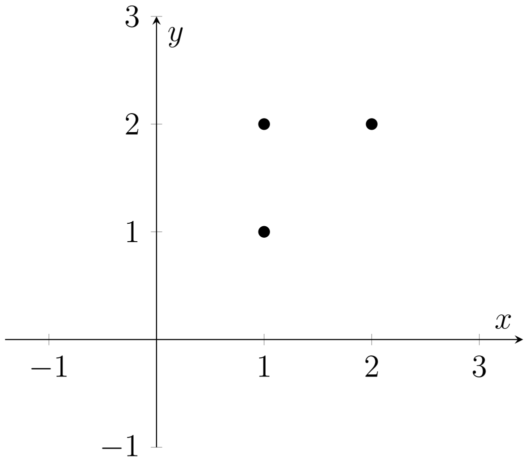
Consider a set of functions H_n that we will call the set of *templates*. Each template is a sequence of length n that is constructed using `0`, `1` or `-` and returns $+1$ for input binary sequences that match it and -1 otherwise. While checking whether a template matches an input, a `-` can match both a `0` and a `1`.

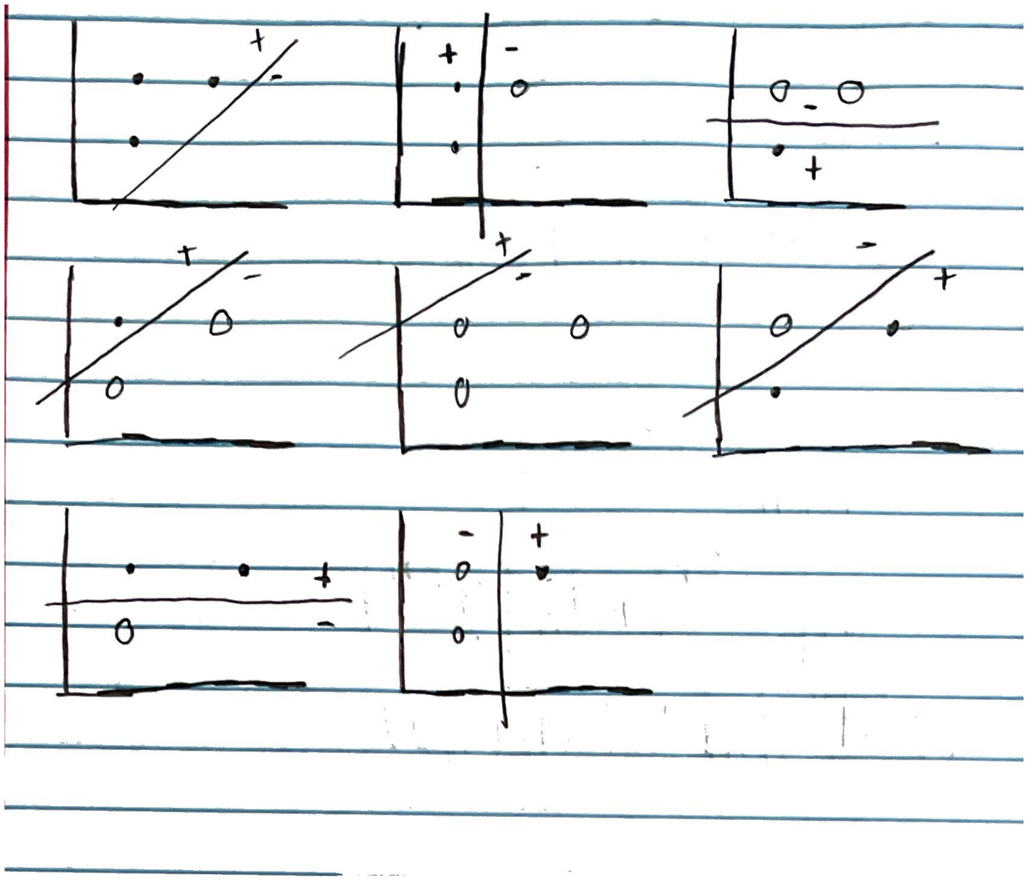
For example, the template `-10` matches the binary strings `010` and `110`, while `-1-` matches all strings that have a `1` in the middle position, namely `010`, `011`, `110` and `111`.

Does the set of templates H_n shatter the set X_n? Prove your answer.

3 VC Dimension [45 points]

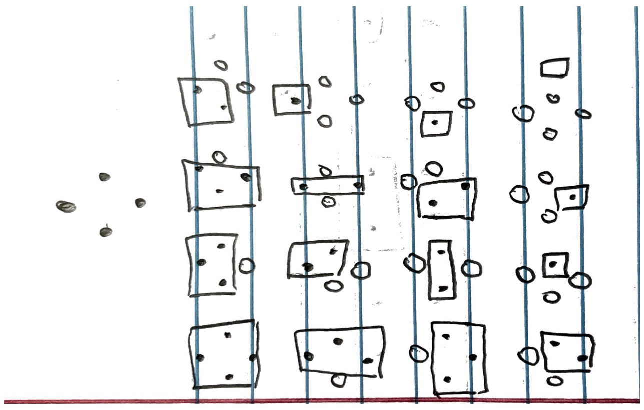
1. [5 points] Assume that the three points below can be labeled in any way. Show with pictures how they can be shattered by a linear classifier. Use filled dots to represent positive classes and unfilled dots to represent negative classes.





2. **VC-dimension of axis aligned rectangles in $\mathbb{R}^d$:** Let H^d_{rec} be the class of axis-aligned rectangles in $\mathbb{R}^d$. When $d = 2$, this class simply consists of rectangles on the plane, and labels all points strictly outside the rectangle as negative and all points on or inside the rectangle as positive. In higher dimensions, this generalizes to d-dimensional boxes, with points outside the box labeled negative.

   (a) [10 points] Show that the VC dimension of H^2_{rec} is 4.



Here we see that a set of 4 points can be shattered if the points are placed in a rectangle that is 90 degrees grom the axis's, this allows for us to pick of any points using the rectangle. This is because there are two directions for each axis and each point is either the farthest point - or + along an axis. The left point is the farthest along the -x axis, the right point the +x axis, the upper point the +y axis and the bottom point the -y axis. This means that we are always able to pick off any number of points using an axis aligned rectangle. We just see which points that we want to include in the rectangle and set the edges of the rectangle such that they are inside the rectangle and the points that we want to exclude we set those outside the rectangle. Again this is possible due to each point being at the farthest extreme along one the axis.

if we introduce a fifth point this would no longer be true. One of the points can never be the farthest along in one of the axis, because there are only 2 axis and 2 directions on each axis that means at most there can only be $2 * 2 = 4$ such points. This cannot set the edges of the rectangle equal to any number of points because if one point isn't the farthest along in 1 of the axis that means its sandwich between two points along that axis. We can simply label the outer points as +'s and the inner point as a -, the rectangle cannot correctly classify these points.

There is also the case where the fifth point is placed on the same x or y axis as another point. In this case if the two points on the same line were to have a different labeling the axis aligned rectangle wouldn't be able to shatter the two.

(b) [10 points] Generalize your argument from the previous proof to show that for d dimensions, the VC dimension of H^d_{rec} is $2d$.

Expanding on the previous argument we can clearly see that the number of points that can be shattered by an axis aligned rectangle in n dimensions is directly proportional to the number of dimensions. Its the number of dimensions * 2 because for each dimension a point can either be the farthest in the - or + direction along that dimension. If we were to introduce a point so that there is $2d+1$ points then there must be a point that isn't the farthest along in any dimension. The same sandwiching argument applies.

There is also the case where there is a pair of points that are on a line lined up along one of the axises. Those points on a line wouldn't be able to be shattered by the axis aligned rectangle in the case where they had different labeling.

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!)
   assume that a finite hypothesis class can classify more than $log_2(C)$ points.

   for a finite hypothesis class there are C functions (each with a possible mapping to + or -) and $log_2(C)$ functions before a labeling is chosen.

   now consider any set of points that is greater than $log_2(C)$. Our hypothesis class will classify all the points up till $log_2(C)$. After that there are no more functions to consider for the extra points because our hypothesis class simply isn't large enough. Therefore we cannot classify more than $log_2(C)$ points and at most we can classify $log_2(C)$

(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.

my classifier is a point on the axis and any point on it will be classified as a + and anywhere else will be classified as a -.

for any one point, if the labeling is 1, we place the classifier on top of it. if the labeling is 0 we place the classifier anywhere else

for two distinct points our classifier doesn't work. We will prove this by considering the following labeling, [+,+] that is a + label close to 0 and a + label closer to 1. Our classifier cannot classify both these points because it cannot lie on both +'s.

for two points that have the same coordinates, consider the following labeling [-, +], that is both points with different labeling lie on the same coordiantes. Our classifier cannot distinguish between these two points.

(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 $\log_2(|\mathcal{C}|)$.

a finite hypothesis class of size C whose VC dimention is $log_2(C)$ would be to split the number line between [0, 1] into $log_2(C)$ evenly distributed intervals with each interval having a labeling of + or -. The left most interval on the number line would include both its end points and each interval after would only include its right end point.

Consider the case where there are $log_2(C)$ points between [0, 1], each inside an interval. Our classifier would be able to correctly classify these points. Just have each interval have the same labeling as the point that resides inside it.

Now consider the case where there are strictly more than $log_2(C)$. In this case there would have to be at least one case where 2 or more points that reside within the same interval. If we were to give these points a different labeling then there would be no way for our classifier to correctly label these points.

Putting those two together can can see that our finite class has a VC dimension of $log_2(C)$.

4 Extra Credit - Decision Lists [25 points]

In this problem, we are going to learn the class of k-decision lists. A decision list is an ordered sequence of if-then-else statements. The sequence of if-then-else conditions are tested in order, and the answer associated to the first satisfied condition is output. See Figure 1 for an example of a 2-decision list.

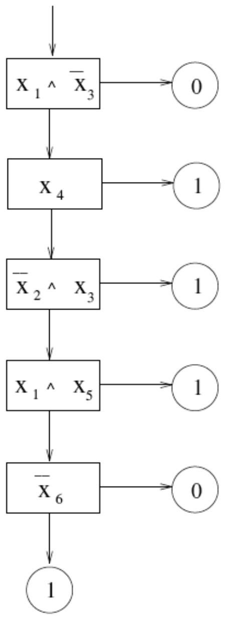


Figure 1: A 2-decision list.

A *k-decision list* over the variables $x_1, \ldots, x_n$ is an ordered sequence $L = (c_1, b_1), \ldots, (c_l, b_l)$ and a bit b, in which each c_i is a conjunction of at most k literals over $x_1, \ldots, x_n$. The bit b is called the *default* value, and b_i is referred to as the bit *associated* with condition c_i. For any input $x \in \{0, 1\}^n$, $L(x)$ is defined to be the bit b_j, where j is the smallest index satisfying $c_j(x) = 1$; if no such index exists, then $L(x) = b$.

We denote by *k-DL* the class of concepts that can be represented by a k-decision list.

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 $c = \{(c_1, b_1), \ldots, (c_l, b_l), b)$.

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