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.

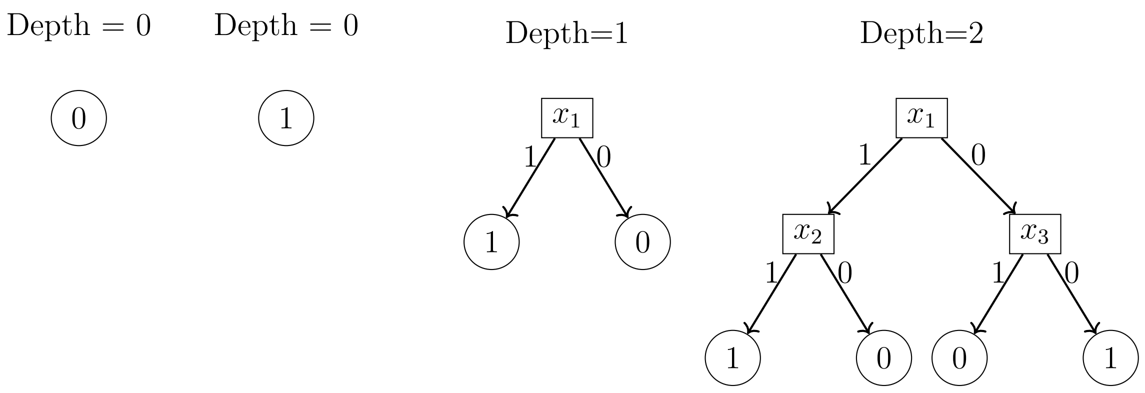
Important. Do not just put down an answer. We want an explanation. No points will be given for just the final answer without an explanation. You will be graded on your reasoning, not just on your final result.

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?
   Ans) There would be 2 decision trees for depth zero. One with label 0 and one with label 1.

   (b) [3 points] What is $S_n(1)$? That is, with n features, how many trees of depth 1 exist?
   Ans) Since there are n features and we need decision tree of depth 1, The root node would have n choices. That means decision tree can be split by n different attributes. Then there would be 2 decisions that can be taken, if we select a feature or we reject a feature. Then all the leaf nodes would have 2 labels, 0 and 1 since it is binary classification.
   Therefore $S_n(1) = 4 * n * n$

   (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) To derive a recursive definition for $S_n(i+1)$ in terms of n and $S_n(i)$, let's consider the expansion of decision trees when increasing their depth.
   Given that each node in a decision tree has two children (as per the assumption of complete trees), when we add one more level of depth, each existing node can have two additional children. Therefore, the number of trees of depth $i+1$ can be expressed as the number of trees of depth i for each node, multiplied by the number of nodes at depth i, which is $S_n(i)$, and then multiplied by the number of ways we can choose one of the n features at each node. So, the correct recursive definition would indeed be:

$$S_n(i+1) = n \times (S_n(i))^2$$

This definition accurately reflects the expansion of the decision tree structure when adding one more level of depth.

(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) To find a closed-form expression representing $\log S_n(k)$ in terms of n and k, we can use the recursive definition we derived earlier:

$$S_n(i+1) = n \times (S_n(i))^2$$

To simplify the expression, let's take the logarithm of both sides:

$$\log S_n(i+1) = \log \left(n \times (S_n(i))^2\right)$$

Using logarithmic properties, we can rewrite this as:

$$\log S_n(i+1) = \log n + 2\log S_n(i)$$

Now, we can apply this recursive relationship repeatedly to find a general formula for $\log S_n(k)$:

$$\begin{aligned}
\log S_n(k) &= \log n + 2\log S_n(k-1) \\
&= \log n + 2\left(\log n + 2\log S_n(k-2)\right) \\
&= \log n + 2\log n + 4\log S_n(k-2) \\
&= \log n + 2\log n + 4\left(\log n + 2\log S_n(k-3)\right) \\
&= \log n + 2\log n + 4\log n + 8\log S_n(k-3) \\
&\vdots \\
&= \log n + 2\log n + 4\log n + \ldots + 2^{k-1}\log n + 2^k \log S_n(0) \\
&= \log n \left(1 + 2 + 4 + \ldots + 2^{k-1}\right) + 2^k \log S_n(0) \\
&= \log n \left(2^k - 1\right) + 2^k \log S_n(0)
\end{aligned}$$

Where $S_n(0)$ represents the number of decision trees with depth 0, which is 1 since there's only one decision tree with no splits. Therefore, $\log S_n(0) = \log 1 = 0$.

So, the final expression for $\log S_n(k)$ is:

$$\log S_n(k) = \log n \left(2^k - 1\right)$$

This expression gives us the order of magnitude of $\log S_n(k)$ in terms of n and k. Therefore, in big O notation, we can express it as:

$$\log S_n(k) = O(k \log n)$$

This indicates that the logarithm of the size of the hypothesis class grows linearly with k and logarithmically with n.

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?

Ans) For the situation where we are given a dataset and asked to use depth-k decision trees as our hypothesis class, the second Occam's razor result, which is for the agnostic setting, is more appropriate.

In the agnostic setting, we are not assuming that the true concept belongs to the hypothesis class under consideration. This is suitable for decision trees because decision trees are typically used as general-purpose models without strong assumptions about the underlying data distribution. In other words, we don't assume that the true concept can be perfectly captured by a decision tree of depth k. Therefore, the agnostic setting allows for a more realistic and flexible approach to learning with decision trees.

(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) To write the sample complexity bound, we use the VC-dimension and the PAC learning framework.

The VC-dimension of a hypothesis class gives us a measure of its complexity and capacity to shatter points. For decision trees of depth k, the VC-dimension is $O(k \log n)$, as we've derived earlier.

Now, using the PAC learning framework, 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 is given by:

$$m = O\left(\frac{1}{\varepsilon}\left(k \log n + \log \frac{1}{\delta}\right)\right)$$

This bound ensures that with high probability, we will be able to find a depth-k decision tree that generalizes well from the training data to unseen data, with the generalization error bounded by ε.

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

   Ans) Yes, the set of depth-k decision trees is PAC learnable, but whether it is efficiently PAC learnable depends on the specific context and the efficiency of the algorithms used for learning.

   (a) **PAC Learnability:** The set of depth-k decision trees is PAC learnable because it satisfies the PAC learning criteria. Given a finite hypothesis class (the set of all possible decision trees of depth k), the PAC learning framework provides bounds on the sample complexity required to learn a hypothesis that generalizes well to unseen data with high probability. As we've derived earlier, there exists a sample complexity bound for depth-k decision trees, ensuring that with enough samples, we can find a hypothesis (decision tree) that approximates the true concept well.

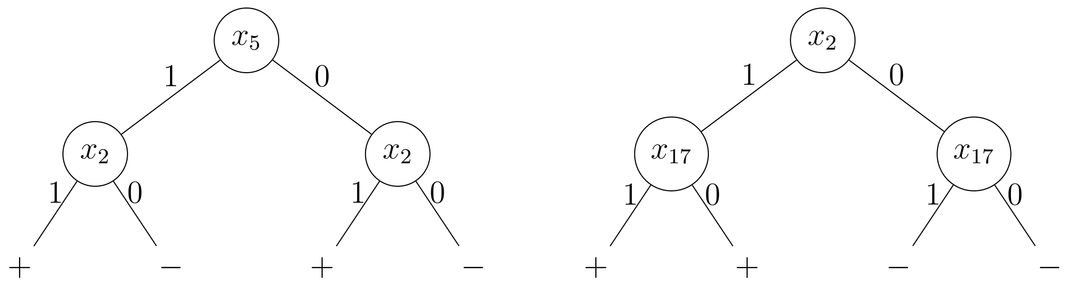
   (b) **Efficient PAC Learnability:** Whether the set of depth-k decision trees is efficiently PAC learnable depends on the computational resources required to implement the learning algorithms. While the existence of a sample complexity bound indicates that the concept is learnable in principle, the efficiency of the learning process also matters in practice. Efficient PAC learnability implies that the computational complexity of the learning algorithm is polynomial in the relevant parameters such as the size of the hypothesis class, the sample size, and the accuracy and confidence parameters.

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.

   Ans)

   - If the number of features n and the depth limit k are large, the sample complexity bound suggests that the number of examples needed m will likely be large as well. This is because the sample complexity grows with $k \log n$.
   - Large sample sizes could pose practical challenges in data collection, labeling, and computational resources required for training. It implies that as the complexity of the hypothesis class (determined by n and k) increases, more data is needed to achieve good generalization performance, which might not always be feasible in practice.
   - Additionally, the dependency on ε and δ should also be considered. Tighter requirements on the desired accuracy (ε) and confidence level (δ) would further increase the required sample size m, potentially making the learning process even more resource-intensive.

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.

Ans)

Assumption: set of templates H_n also consist of n bits and can't exceed n bits.

If $n = 1$, then $X_1 = \{0, 1\}$. The corresponding table is as follows:

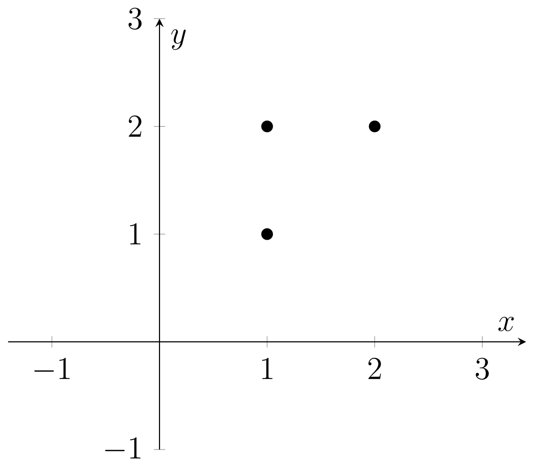
Table 1: Matching Function for Different Labels

Label of 0	Label of 1	Matching function
$+1$	$+1$	$-$
-1	$+1$	1
$+1$	-1	0
-1	-1	Nothing possible

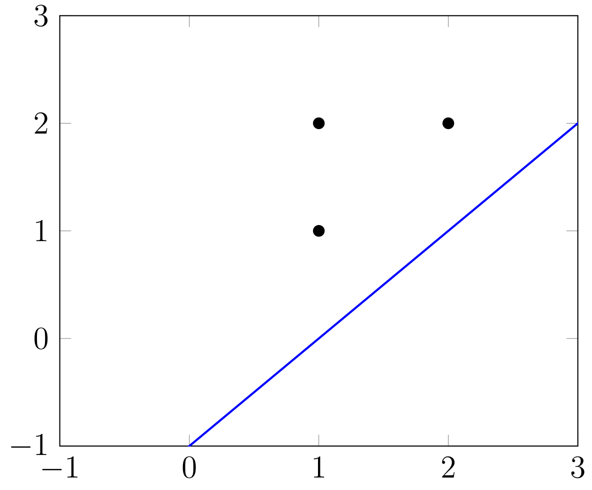
Since H_1 can't shatter X_1 itself, therefore in general H_n can't shatter X_n

3 VC Dimension [45 points]

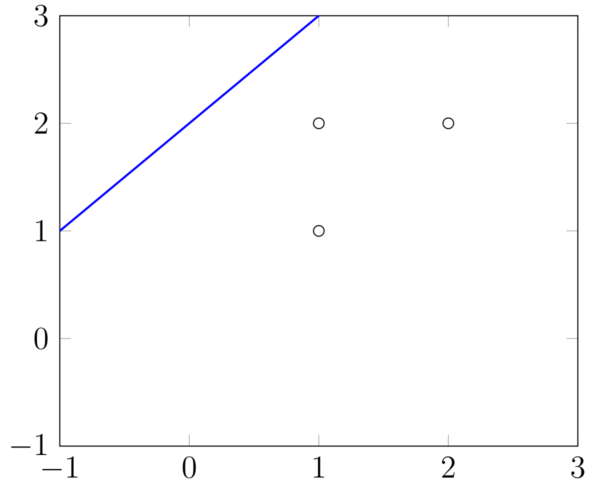
1. [5 points] Assume that the three points below can be labeled in any way. Show with pictures how they can be shattered by a linear classifier. Use filled dots to represent positive classes and unfilled dots to represent negative classes.



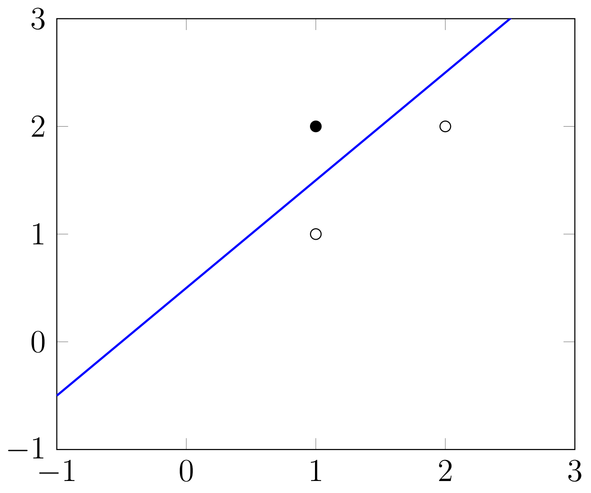
3
2
1
−1
y
x
−1
1
2
3

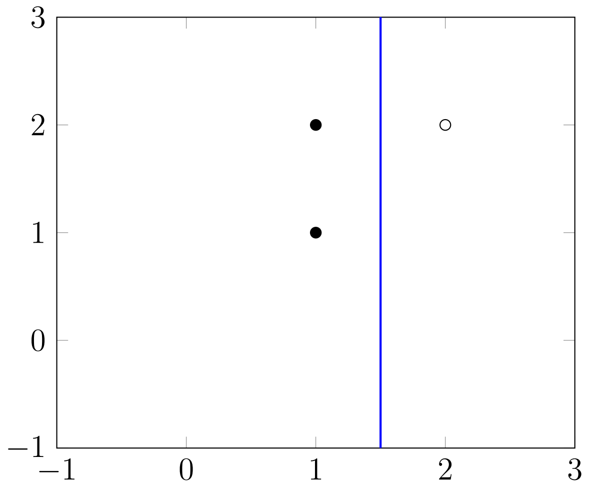


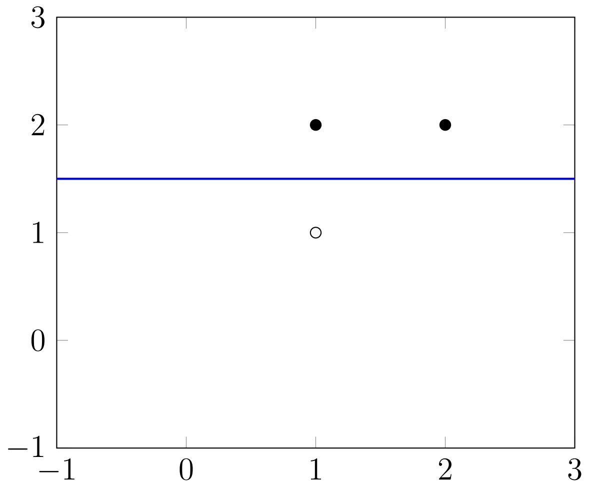
3
2
1
0
−1
−1
0
1
2
3

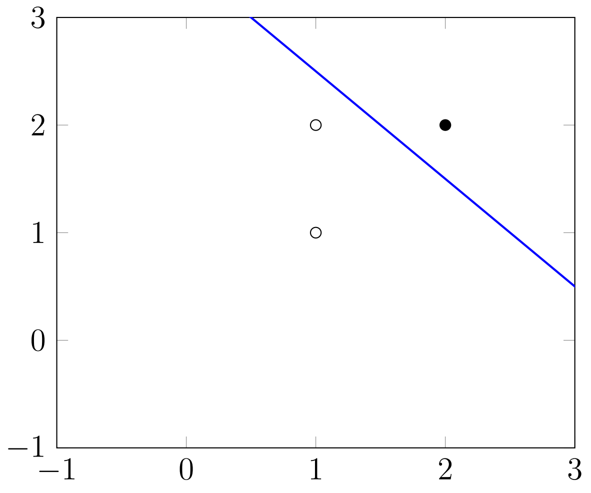


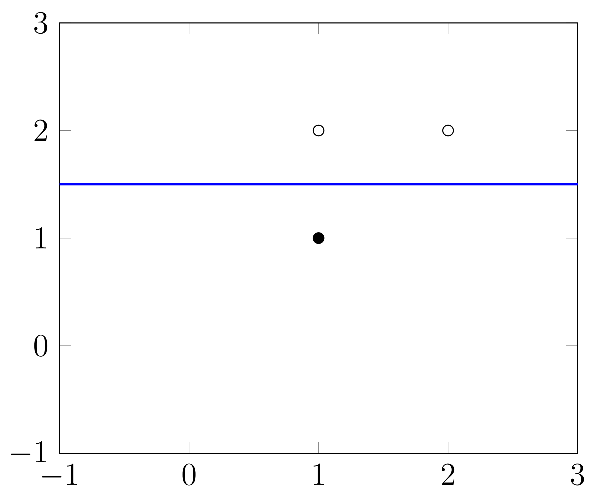
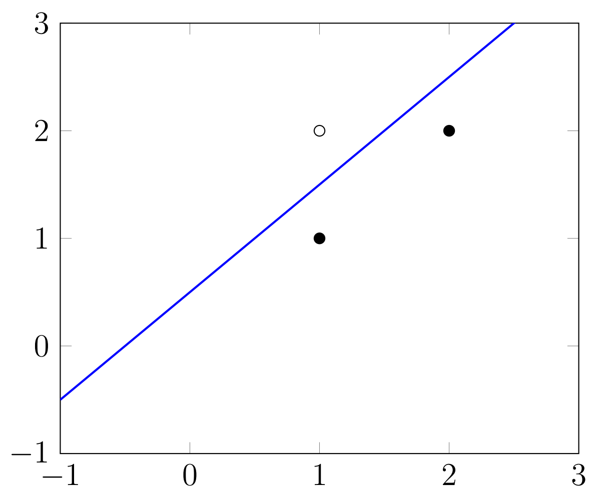
3
2
1
0
−1
−1
0
1
2
3


3
2
1
0
−1
−1
0
1
2
3


3
2
1
0
−1
−1
0
1
2
3


3
2
1
0
−1
−1
0
1
2
3


3
2
1
0
−1
−1
0
1
2
3

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.

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

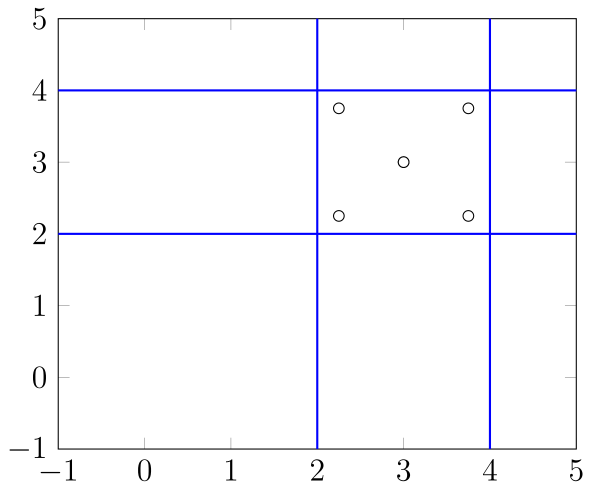
   Ans)
   To show that the VC dimension of H^2_{rec} is 4, we need to demonstrate:

   (a) There exists a set of 4 points that can be shattered by H^2_{rec}.

   (b) No set of 5 points can be shattered by H^2_{rec}.

   For the first part, consider 4 points forming a convex quadrilateral. We can find a rectangle that separates positive and negative labels for all possible labelings of the 4 points. This proves that H^2_{rec} can shatter 4 points.

   For the second part, let's consider a set of 5 points forming a convex pentagon. No matter how we label the points, we cannot find a rectangle that separates positive and negative labels for all possible labelings of these 5 points. This demonstrates that H^2_{rec} cannot shatter 5 points.



Therefore, the VC dimension of H^2_{rec} is 4.
As you can see the four points form a rectangle. If we add another point, shattering won't work.

(b) Generalization to Higher Dimensions

To generalize to higher dimensions, let's consider d dimensions.

For $d = 1$, the class H^1_{rec} corresponds to intervals on the real line, which can shatter 2 points (one positive, one negative).

Now, let's consider d dimensions. For each dimension, we can independently choose an interval (or "side") to include or exclude a point. Therefore, for each dimension, we can shatter 2 points (one positive, one negative). Since there are d dimensions, the total VC dimension is $2d$.

Therefore, the VC dimension of H^d_{rec} in d dimensions 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!)

   (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.

   (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) (a) Showing the VC Dimension Bound for Finite Hypothesis Class C

To prove that for a finite hypothesis class C, its VC dimension can be at most $\log_2(|C|)$, we can use the following reasoning:

Assume for contradiction that there exists a finite hypothesis class C with VC dimension greater than $\log_2(|C|)$.

Let the VC dimension of C be $d > \log_2(|C|)$. This means that there exists a set of d points that can be shattered by C.

However, C is a finite class with $|C|$ hypotheses. By the definition of VC dimension, the largest set that can be shattered by C cannot have more than $|C|$ points.

But we have assumed that there exists a set of d points that can be shattered by C, where $d > \log_2(|C|)$. This contradicts the definition of VC dimension, as d exceeds the maximum number of points that can be shattered by C.

Therefore, the VC dimension of a finite hypothesis class C must be at most $\log_2(|C|)$.

(b) Example of an Infinite Class with VC Dimension 1

An example of an infinite class C of functions over the real interval $X = [0, 1]$ with VC dimension exactly 1 is the class of threshold functions.

The class C consists of functions that output 1 if the input is greater than or equal to a certain threshold, and 0 otherwise. Formally, for a threshold θ in the interval $[0, 1]$, the function $f_\theta(x)$ is defined as:

$$f_\theta(x) = \begin{cases} 1 & \text{if } x \geq \theta \\ 0 & \text{otherwise} \end{cases}$$

The VC dimension of this class is 1 because any set of 2 distinct points can be shattered by threshold functions. Specifically, for any two points $x_1 < x_2$ in X, we can choose a threshold θ such that one point is labeled 0 and the other is labeled 1.

(c) Example of a Finite Class with VC Dimension $\log_2(|C|)$

Consider the class C of all axis-aligned rectangles in the interval $X = [0, 1]$ with integer endpoints. Each rectangle in C can be uniquely identified by its left and right endpoints.

Let $|C| = n$ be the number of rectangles in C. Since each rectangle corresponds to a unique pair of integer endpoints, we can represent each rectangle by an integer index from 1 to n.

The VC dimension of this class is $\log_2(|C|)$ because any set of $\log_2(|C|)$ distinct points can be shattered by C. Specifically, for any set of $\log_2(|C|)$ distinct points, we can choose a rectangle that covers one point and excludes the others, and since there are $|C|$ rectangles in total, all possible labelings of $\log_2(|C|)$ points can be realized.

Therefore, the class C of axis-aligned rectangles in $X = [0, 1]$ with integer endpoints is an example of a finite class with VC dimension $\log_2(|C|)$.

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

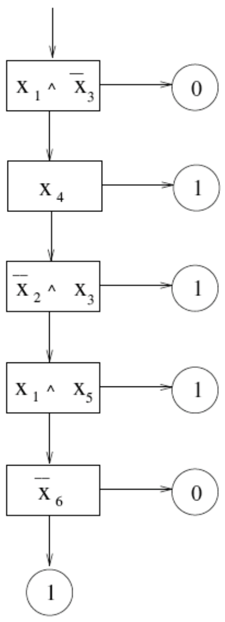


Figure 1: A 2-decision list.

order, and the answer associated to the first satisfied condition is output. See Figure 1 for an example of 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.)

Ans)

1. Representing the Complement $\neg c$ with a k-decision list:

Given a concept c represented by a k-decision list $L = ((c_1, b_1), \dots, (c_l, b_l), b)$, where each c_i is a conjunction of at most k literals over variables $x_1, \dots, x_n$, and b is the default bit, we can represent its complement $\neg c$ using the following procedure:

1. Negate each condition c_i in L to $\neg c_i$.
2. Keep the associated bits b_i unchanged.
3. Negate the default bit b to $\neg b$.

The resulting k-decision list L' now represents $\neg c$. Here's why:

- For any input x, if $c_i(x) = 1$ for some condition c_i in the original list L, then $\neg c_i(x) = 0$ in the new list L', so the associated bit b_i remains unchanged. Therefore, the output of L' for x is the complement of the output of L.
- If none of the conditions in L are satisfied by x, then b is the default output of L, and $\neg b$ is the default output of L', which are opposite bits. So, the output of L' for x is the complement of the output of L.

Therefore, the complement $\neg c$ can be represented by a k-decision list.

2. Occam's Razor and PAC-Learnability:

Occam's Razor states that among hypotheses that fit the training data, simpler hypotheses are more likely to generalize well to unseen data.

For any constant $k \geq 1$, consider a sample complexity bound m such that with probability at least $1 - \delta$, the following holds:

$$m \geq \frac{1}{\varepsilon}\left(2k \log(n) + \log \frac{1}{\delta}\right)$$

where n is the number of variables and ε is the error parameter.

This bound guarantees that with high probability, we can find a k-decision list that approximates the target concept well within error ε using a polynomial number of samples.

Therefore, the class of k-decision lists is PAC-learnable.

3. 1-Decision Lists as Linearly Separable Functions:

A 1-decision list is a list with each condition being a conjunction of at most 1 literal, i.e., a single variable or its negation. Let's denote a 1-decision list as $L = ((x_i, b_i), \dots, (x_j, b_j), b)$.

To show that 1-decision lists are linearly separable functions, we can construct a weight vector $\mathbf{w}$ such that it makes the same predictions as the given 1-decision list.

For each condition (x_i, b_i) in the 1-decision list L, if $b_i = 1$, we set $w_i = 1$ if x_i is positively associated and $w_i = -1$ if x_i is negatively associated. If $b_i = 0$, we set $w_i = 0$.

Let b' be the default bit in the 1-decision list. If $b' = 1$, we set $w_0 = -1$, and if $b' = 0$, we set $w_0 = 1$.

Now, for any input x, the sign of $\mathbf{w} \cdot \mathbf{x}$ (where $\mathbf{x}$ is the input vector corresponding to x) will be the same as the output of the 1-decision list L. This is because $\mathbf{w} \cdot \mathbf{x}$ is positive if and only if the positively associated variables satisfy their conditions and the negatively associated variables do not satisfy their conditions, which is exactly the condition for the output of L to be 1.

Therefore, 1-decision lists can be represented as linearly separable functions.