HW1

Nate Love

Due: 15 June 2025

1

1.1 a)

Prove:

$$\mathbb{P}[f(x) \neq g(x)] = \frac{1 - \mathbb{E}[f(x)g(x)]}{2}$$

Start with:

$$\mathbb{E}_{x\sim D}[f(x)g(x)] = (+1) \cdot \mathbb{P}[f(x) = g(x)] + (-1) \cdot \mathbb{P}[f(x) \neq g(x)]$$

$$= \mathbb{P}[f(x) = g(x)] - \mathbb{P}[f(x) \neq g(x)]$$

We know:

$$\mathbb{P}[f(x) = g(x)] + \mathbb{P}[f(x) \neq g(x)] = 1$$

And so:

$$\mathbb{P}[f(x) = g(x)] = 1 - \mathbb{P}[f(x) \neq g(x)]$$

Substituting in the first equation:

$$\mathbb{E}_{x\sim D}[f(x)g(x)] = (1 - \mathbb{P}[f(x) \neq g(x)]) - \mathbb{P}[f(x) \neq g(x)]$$

$$\mathbb{E}_{x\sim D}[f(x)g(x)] = 1 - 2\mathbb{P}[f(x) \neq g(x)]$$

$$-\mathbb{E}_{x\sim D}[f(x)g(x)]+2\mathbb{P}[f(x) \neq g(x)] = 1-2\mathbb{P}[f(x) \neq g(x)]-\mathbb{E}_{x\sim D}[f(x)g(x)]+2\mathbb{P}[f(x) \neq g(x)]$$

After this:

$$2\mathbb{P}[f(x) \neq g(x)] = 1 - \mathbb{E}_{x\sim D}[f(x)g(x)]$$

Now divide by two:

$$\frac{2\mathbb{P}[f(x) \neq g(x)]}{2} = \frac{1 - \mathbb{E}_{x\sim D}[f(x)g(x)]}{2}$$

Q.E.D:

$$\mathbb{P}[f(x) \neq g(x)] = \frac{1 - \mathbb{E}[f(x)g(x)]}{2}$$

1.2 b)

Would this still be true if the domain were some other domain (such as Rn, where R denotes the real numbers, with say the Gaussian distribution) instead of $\{-1,1\}^n$?

It would remain true. The key relationship of

$$\mathbb{E}_{x\sim D}[f(x)g(x)] = \mathbb{P}[f(x) = g(x)] - \mathbb{P}[f(x) \neq g(x)]$$

remains the same if we add in something like a Gaussian distribution. This relationship only depends on the input and output values, which will remain boolean as $\{-1,1\}$. The distribution could theoretically be changed to anything and as long as this remains boolean the key relationship remains unchanged.

2

You can write f as a multivariate polynomial $p(x_1,...,x_n) \in \{-1,1\}^n$ such that for every input $x \in \{-1,1\}^n, f(x) = p(x)$ as such:

We need to create an indicator function that corresponds to the path of a decision tree. We also want this function to match the outputs of the tree. This will yield a polynomial built on a function that uses the indicator polynomial for each leaf and the label at each leaf. t represents the number of leaves that are in our decision tree:

$$\sum_{l=1}^{t} label_l \cdot I_l(x) = p(x)$$

Our polynomial needs to equal 1 based on the label and the output of 1 or -1. This leads to the equations:

$$\frac{1+x_i}{2}$$

when $x_i = 1$ and

$$\frac{1-x_i}{2}$$

when $x_i = -1$
If the path matches, then these equations will output a 1 with the correct path or a 0 otherwise. For a leaf l, it will have path conditions of $x_{i1} = b_1, ..., x_{in} = b_n$
We can simplify this to write:

$$I_l(x) = \prod_{j=1}^{k} \frac{1+b_j x_i}{2}$$

$$p(x) = \sum_{i=1}^{n} label_n \cdot I(x)$$

And so $f(x) = p(x)$

3

Compute a depth-two decision tree for the training data in table 1 using the Gini function, $C(a) = 2a(1-a)$ as described in class. What is the overall accuracy on the training data of the tree?

First step: $C(a) = 2a(1-a)$

$$2 \cdot \frac{85}{250} \cdot \frac{165}{250} = .448$$

Now we test X,Y,Z and pick the best one with the best gain to set as the root:

X:

$$Pr(X = 0) \cdot \phi(NegX = 0) + Pr(X = 1) \cdot \phi(NegX = 1)$$
$$\frac{3}{5} \cdot 2 \cdot \frac{3}{10} \cdot \frac{7}{10} + \frac{2}{5} \cdot 2 \cdot \frac{4}{10} \cdot \frac{6}{10} = .444$$

Y:

$$Pr(Y = 0) \cdot \phi(NegY = 0) + Pr(Y = 1) \cdot \phi(NegY = 1)$$
$$\frac{12}{25} \cdot 2 \cdot \frac{50}{120} \cdot \frac{70}{120} + \frac{13}{25} \cdot 2 \cdot \frac{35}{130} \cdot \frac{95}{130} = .438$$

Z:

$$Pr(Z = 0) \cdot \phi(NegZ = 0) + Pr(Z = 1) \cdot \phi(NegZ = 1)$$
$$\frac{12}{25} \cdot 2 \cdot \frac{1}{2} \cdot \frac{1}{2} + \frac{13}{25} \cdot 2 \cdot \frac{5}{26} \cdot \frac{21}{26} = .401$$

Now calculate gains:

X:

$$.448 - .444 = .004$$

Y:

$$.448 - .438 = .010$$

Z:

$$.448 - .401 = .047$$

Our root will be Z. Now we will do it again for the next depth.

$$\Pr(Y = 0 \mid Z = 0) \cdot \phi_{Y=0 \cap Z=0} + \Pr(Y = 1 \mid Z = 0) \cdot \phi_{Y=1 \cap Z=0}$$

$$\frac{50}{120} \cdot 2 \cdot \frac{15}{50} \cdot \frac{35}{50} + \frac{70}{120} \cdot 2 \cdot \frac{45}{70} \cdot \frac{25}{70} = .442$$

$$\Pr(Y = 0 \mid Z = 1) \cdot \phi_{Y=0 \cap Z=0} + \Pr(Y = 1 \mid Z = 1) \cdot \phi_{Y=1 \cap Z=0}$$

$$\frac{70}{130} \cdot 2 \cdot \frac{55}{70} \cdot \frac{15}{70} + \frac{60}{130} \cdot 2 \cdot \frac{50}{60} \cdot \frac{10}{60} = .309$$

Now combine and find total score:

$$\Pr(Z = 0) \cdot C(a)_{Z=0} + \Pr(Z = 1) \cdot C(a)_{Z=1}$$

$$\frac{120}{250} \cdot .442 + \frac{130}{250} \cdot .309 = .372$$

$$\Pr(X = 0 \mid Z = 0) \cdot \phi_{X=0 \cap Z=0} + \Pr(X = 1 \mid Z = 0) \cdot \phi_{X=1 \cap Z=0}$$

$$\frac{80}{120} \cdot 2 \cdot \frac{45}{80} \cdot \frac{35}{80} + \frac{40}{120} \cdot 2 \cdot \frac{15}{40} \cdot \frac{25}{40} = .484$$

$$\Pr(X = 0 \mid Z = 1) \cdot \phi_{X=0 \cap Z=0} + \Pr(X = 1 \mid Z = 1) \cdot \phi_{X=1 \cap Z=0}$$

$$\frac{70}{130} \cdot 2 \cdot \frac{60}{70} \cdot \frac{10}{70} + \frac{60}{130} \cdot 2 \cdot \frac{45}{60} \cdot \frac{15}{60} = .305$$

Now combine and find total score:

$$\Pr(Z = 0) \cdot C(a)_{Z=0} + \Pr(Z = 1) \cdot C(a)_{Z=1}$$

$$\frac{120}{250} \cdot .484 + \frac{130}{250} \cdot .305 = .391$$

Compare Gains:
Y:

$$.448 - .372 = .076$$

X:

$$.448 - .391 = .057$$

Based on this, we chose Y as the next split.
From here we can calculate our total accuracy.

Accuracy of Z=0 and Y=0 leaf =

$$\frac{35}{50}$$

Accuracy of Z=0 and Y=1 leaf =

$$\frac{45}{70}$$

Accuracy of Z=1 and Y=0 leaf =

$$\frac{55}{70}$$

Accuracy of Z=1 and Y=1 leaf =

$$\frac{50}{60}$$

Total Accuracy =

$$\frac{185}{250} = .74$$

4

For our algorithm, we will take the input of m labeled training examples. We will then sort the examples by their $x-values$. Then we will find the $maximum$ and $minimum$ for the largest x_i with label -1 and the smallest x_i with label $+1$. Then we will set our $h*$ to $\frac{x^- + x^+}{2}$. Then we output h.

The worst case for our algorithm is that when the number line is split at h* all -1 labels and all +1 labels are to the far left and far right of h*. The probability that this occurs is $2 \cdot (1-\epsilon)^m \leq \delta$. Using this we can confirm our algorithm is within the given Big -O bounds.

$$(1-\epsilon)^m \leq \frac{\delta}{2}$$

$$e^{-\epsilon \frac{m}{2}} \leq \frac{\delta}{2}$$

$$\frac{-\epsilon^m}{2} \leq \lg(\frac{\delta}{2})$$

$$m \leq \frac{2 \cdot \lg(\frac{\delta}{2})}{\epsilon}$$

$$m \leq \frac{2 \cdot \lg(\frac{\delta}{2})}{\epsilon}$$

For Big O notation we can drop constants resulting in $m = O(\frac{1}{\epsilon} \lg \frac{1}{\delta})$ with error at most ϵ with probability at least $1-\delta$. The sorting part of our algorithm will only be $O(m \lg n)$ and as such keeps us within our bounds of $m = O(\frac{1}{\epsilon} \lg \frac{1}{\delta})$

5

5.1 a)

This question is much like question 4. The worst case for our hypothesis is: $err(h^{'}) > \epsilon$. The probability that a sample of K blocks is in this case is $(1-\epsilon)^k$. This needs to happen with probability at most $\delta^{'}$

So we get:

$$(1-\epsilon)^k \leq \delta^{'}$$

$$k \leq \frac{1}{\epsilon} \lg \frac{1}{\delta'}$$

5.2 b)

Using part a, calculate m samples.

$$k \leq \frac{1}{\epsilon} \lg \frac{1}{\delta'}$$

In this part, we union bound hypotheses, resulting in:

$$P[\cup err(h_1) > \epsilon] \leq \sum_{i=1}^{t} P[err(h_i) > \epsilon] \leq t \cdot \delta^{'}$$

This demonstrates the probability that any one hypothesis has an error greater than ϵ is at most the sum of all individual hypotheses. As such, the probability of that is $t \cdot \delta^{'}$ From this we can use our answer in part a to find how large m needs to be.

$$t \cdot \delta^{'} \leq \delta$$

$$\delta^{'} \leq \frac{\delta}{t}$$

Now plug this into the equation from part a.

$$k \leq \frac{1}{\epsilon} \lg \frac{t}{\delta}$$

$$m = \frac{1}{\epsilon} \lg \frac{t}{\delta}$$

5.3 c)

To run this algorithm, we would take m examples from distribution D. Then we would take these examples and send them to learner A. If there is a mistake, A will update its hypothesis. We will continue to do this for all samples and A will return a hypothesis h. h will then be returned as a hypothesis for the PAC.