Question 1.

Let X contain all real numbers from $[0;1]$ which can be represented using 256 bits. Let $\mathcal{H} = X$, and let the decision be given by $H \in \mathcal{H}$ as

$$h(x) = 1 \text{ iff } x > H$$

Determine an m such that with probability at least 0.9, $\text{err}(h) < 0.1$, where h is an arbitrary hypothesis from $\mathcal{H}$ consistent with m i.i.d. examples from X. Estimate it

(a) without using any *textbook* lower bounds

(b) using the lower bound $m > \frac{1}{\epsilon} \ln \frac{|\mathcal{H}|}{\delta}$

(c) using the lower bound $m > \frac{8}{\epsilon} \left(\text{VC}(\mathcal{H}) \cdot \ln \frac{16}{\epsilon} + \ln \frac{2}{\delta}\right)$

Answer:

We have

$$\begin{aligned} \epsilon &= 0.1 \\ \delta &= 1 - 0.9 = 0.1 \\ |\mathcal{H}| &= 2^{256} \end{aligned}$$

(a) For a fixed h, the probability that it is "bad" ($\text{err}(h) > \epsilon$) and still consistent with m i.i.d. observations is at most $(1-\epsilon)^m = 0.9^m$.

For an arbitrary $h \in \mathcal{H}$, we can bound the probability of at least one of them being "bad" by

$$\sum_{h \in \mathcal{H}} (1-\epsilon)^m = |\mathcal{H}|(1-\epsilon)^m = 2^{256} 0.9^m$$

We want this probability to be smaller than δ:

$$\begin{aligned} |\mathcal{H}|(1-\epsilon)^m &< \delta \\ m &\geq \log_{1-\epsilon} \frac{\delta}{|\mathcal{H}|} \end{aligned}$$

i.e.,

$$m > \log_{0.9} \frac{0.1}{2^{256}} \approx 1707 \text{ examples (smallest such } m\text{)} \tag{1}$$

(b)

$$\begin{aligned} m &> \frac{1}{\epsilon} \ln \frac{|\mathcal{H}|}{\delta} \\ m &> \frac{1}{0.1} \ln \frac{2^{256}}{0.1} \approx 1798 \text{ examples (smallest such } m\text{)} \end{aligned}$$

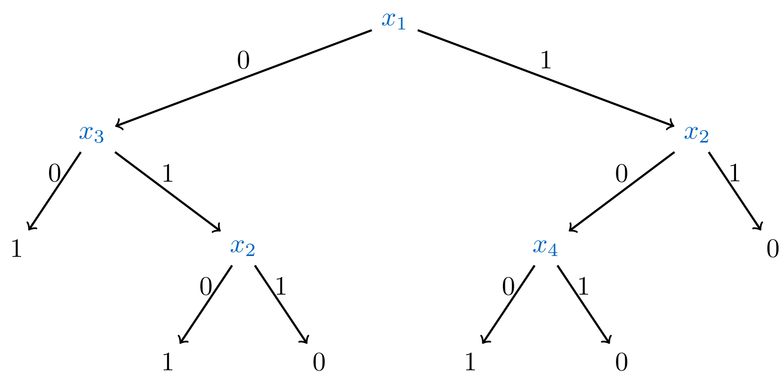
which is slightly greater than before because the upper bound $(1-\epsilon)^m < e^{-\epsilon m} (\epsilon > 0)$ is used in the derivation of the formula.

(c) $\text{VC}(\mathcal{H}) = 1$ because a single number from X can evidently be shattered (classified positively or negatively by hypotheses from $\mathcal{H}$) but two different numbers from X cannot be shattered: the smaller cannot be made positive while the larger is negative.

$$m > \frac{8}{\epsilon} \left(\text{VC}(\mathcal{H}) \cdot \ln \frac{16}{\epsilon} + \ln \frac{2}{\delta} \right) \approx 646 \text{ examples}$$

Question 2.

Consider the following decision tree:



(a) Express the tree as a 3-DNF.

(b) Express the tree as a 3-CNF.

(c) How can we use (modify) the generalization algorithm to learn k-decision trees in the PAC learning model?

Answer:

(a) A 3-DNF is a disjunction of minterms conjoining at most 3 literals.

We construct each minterm by following one path to a positive label. By disjoining all those paths, we get the final DNF tree representation.

$$(\neg x_1 \wedge \neg x_3) \vee (\neg x_1 \wedge x_3 \wedge \neg x_2) \vee (x_1 \wedge \neg x_2 \wedge \neg x_4)$$

(b) A 3-CNF is a conjunction of maxterms (clauses) disjoining at most 3 literals.

We can use the fact that a negation of a DNF is a CNF. We can't negate the DNF constructed above directly, since then, we would consider paths going into the negative labels. However, we can construct a DNF going into the negative labels and negate that, giving us a CNF capturing the paths going into the positive labels.

$$\neg\left((\neg x_1 \wedge x_3 \wedge x_2) \vee (x_1 \wedge \neg x_2 \wedge x_4) \vee (x_1 \wedge x_2)\right)$$

We could also construct the CNF directly. Consider each path to a negative label and "do everything in your power to avoid going down that path". For example, considering the "path" $\langle \neg x_1, x_3, x_2 \rangle$, construct the clause $(x_1 \vee \neg x_3 \vee \neg x_2)$.

Regardless of the strategy used, the final 3-CNF reads as

$$(x_1 \vee \neg x_3 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2)$$

(c) We can use the fact following from the exercises above that k-DT $\subseteq$ k-CNF (k-DNF). Hence, we will use the generalization algorithm for learning $k-$CNFs ($k-$DNFs). Each clause (minterm) will be encoded by a new propositional variable and then we will simply be learning a conjunction (disjuction). There will be $\sum_{i=1}^{k} \binom{n}{i} 2^i \leq \text{poly}(n)$ propositions, hence we will also learn efficiently. We won't be learning properly, though.

Using the adapted algorithm, we will be learning k-DTs in the mistake bound learning model, which implies that we will also be learning them in the PAC model.