CS/DSC/AI 391L: Machine Learning

Homework 1 - Theory - Solutions

Lecture: Prof. Adam Klivans

Keywords: Boolean functions, mistake bounds, PAC learning

1. Since f, g are $\{-1, 1\}$-valued, $f(x)g(x)$ is $+1$ whenever $f(x) = g(x)$ and -1 whenever $f(x) \neq g(x)$. Thus we can write

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

   using the fact that $\mathbb{P}_{x\sim D}[f(x) = g(x)] = 1 - \mathbb{P}_{x\sim D}[f(x) \neq g(x)]$. Rearranging the above equation proves the result.

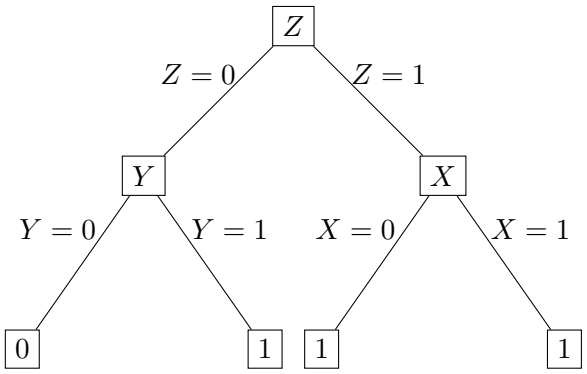
   Notice that we did not need any properties from the domain or the distribution at all for this proof, just that f, g be $\{-1, 1\}$-valued. Thus the statement holds for arbitrary domains.

2. We can write the decision tree as a polynomial by decomposing it in terms of its root-to-leaf paths. Consider an example path where we move along $x_1 = 1$ at the root x_1, along $x_3 = -1$ at x_3, along $x_8 = 1$ at x_8, and then output -1; in other words, if $x_1 = 1$, $x_3 = -1$, and $x_8 = 1$, output -1. We can represent this by the following term:

$$\left(\frac{1 + x_1}{2}\right)\left(\frac{1 - x_3}{2}\right)\left(\frac{1 + x_8}{2}\right)(-1)$$

   Clearly this is nonzero iff $x_1 = 1, x_3 = -1, x_8 = 1$, in which case it evaluates to -1. In this way we can represent any path by a polynomial term. We can now write the whole decision tree f by summing the terms for each of the t root-to-leaf paths. Since any assignment of values to $x_1, \ldots, x_n$ will follow exactly one of the root-to-leaf paths, exactly one of the corresponding terms will be nonzero, and the overall sum will be $+1$ or -1 according to the value at the leaf. Thus we obtain a polynomial p such that $f(x) = p(x)$ for all $x \in \{-1, 1\}^n$.

3. Overall accuracy is 74%. The decision tree is shown below:



   The calculations are as follows:

Root node: Note that the potential for all the examples is $C(Pr[Pos]) = 2 * \frac{165}{250} * \frac{85}{250} = 0.4488$.

If we choose X to be the root node, then the new expected potential is $Pr[X = 0]C(Pr[Pos|X = 0]) + Pr[X = 1]C(Pr[Pos|X = 1]) = \frac{150}{250} * C(\frac{105}{150}) + \frac{100}{250} * C(\frac{60}{100}) = 0.444$

If we choose Y, it is $\frac{120}{250} * C(\frac{70}{120}) + \frac{130}{250} * C(\frac{95}{130}) = 0.437949$.

If we choose Z it is $\frac{120}{250} * C(\frac{60}{120}) + \frac{130}{250} * C(\frac{105}{130}) = 0.401538$.

So, Z minimizes the new potential, or in other words, it maximizes the information gain which is $0.4488 - 0.401538 = 0.047262$. Thus, we should pick Z to be the root node.

Left node: Since an example only enters the left node when $Z = 0$, we will restrict ourselves to the examples where $Z = 0$.

Note that the total potential in this case is $C(\frac{60}{120}) = 0.5$.

If we choose X to be the left node, then the new expected potential is $\frac{80}{120} * C(\frac{45}{80}) + \frac{40}{120} * C(\frac{15}{40}) = 0.484375$

If we choose Y to be the left node, then the new expected potential is $\frac{50}{120} * C(\frac{15}{50}) + \frac{70}{120} * C(\frac{45}{70}) = 0.442857$

Thus, choosing Y minimizes the new expected potential, or maximizes the information gain which is $0.5 - 0.442857 = 0.057143$. Since in this case, we have more examples labeled negative then positive when $Y = 0$, we will output negative when $Y = 0$. Similarly, we have more examples labeled positive than negative when $Y = 1$, and so when $Y = 1$, we will output positive.

Right node: Since an example only enters the right node when $Z = 1$, we will restrict ourselves to the examples where $Z = 1$.

Note that the total potential in this case is $C(\frac{105}{130}) = 0.31065088757$.

If we choose X to be the right node, then the new expected potential is $\frac{70}{130} * C(\frac{60}{70}) + \frac{60}{130} * C(\frac{45}{60}) = 0.304945$

If we choose Y to be the right node, then the new expected potential is $\frac{70}{130} * C(\frac{55}{70}) + \frac{60}{130} * C(\frac{50}{60}) = 0.309524$

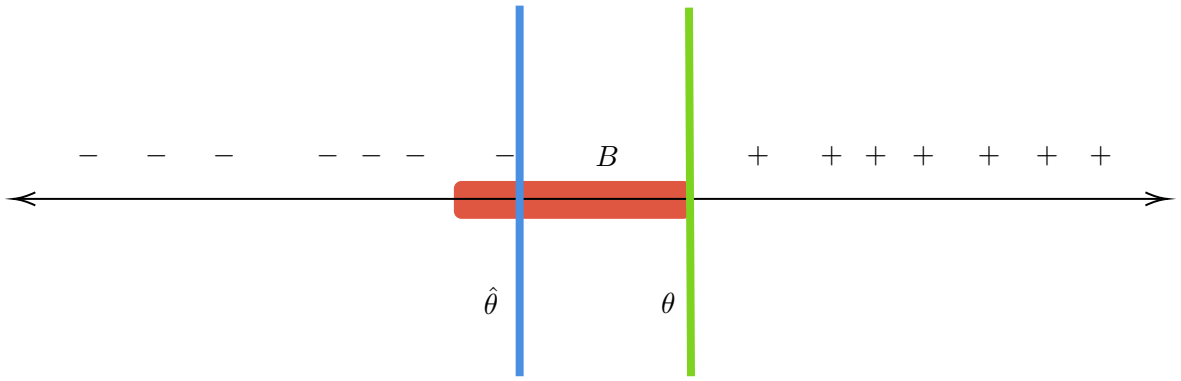
Thus, choosing X minimizes the new expected potential, or maximizes the information gain which is $0.31065088757 - 0.304945 = 0.00570588757$. Since in this case, we have more examples labeled positive then negative when $X = 0$, we will output positive when $X = 0$. Similarly, we have more examples labeled positive than negative when $X = 1$, and so when $X = 1$, we will output positive.

Our accuracy is computed as (number of examples our decision tree labels correctly)/(total number of examples). In this case, this is $\frac{185}{250} = 0.74 = 74\%$.

4. This is actually a strictly simpler, instructive version of the axis-parallel rectangles problem from lecture. As in that problem, the algorithm is very natural: draw a large number (say m) of examples, and then pick the "tightest-fitting" threshold function. More concretely, we can arrange our training data in ascending order in terms of x, and then pick the largest x that is labeled -1 as our threshold. (Another approach would be to pick the smallest x that is labeled $+1$; this also works and has a nearly identical analysis.) This can be done efficiently in $O(m)$ time by going through all m points. (Aside: we cannot use binary search since the points don't arrive in sorted order.) What remains is the analysis of how large m needs to be in order to be have high confidence that our output has low error.

To do this, suppose h_θ, for some $\theta \in \mathbb{R}$, is the true threshold function that is labeling the data. Suppose that the threshold we obtain (by picking the largest x labeled -1 in our dataset) is $\hat{\theta}$. Notice that we will necessarily have $\hat{\theta} \leq \theta$, and the only area where $h_{\hat{\theta}}$ differs from the true h_θ is the interval $[\hat{\theta}, \theta]$. This is where our classifier errs, and the error is precisely the probability mass of this interval.

Define B to be the interval immediately to the left of θ that has probability mass ϵ. Observe that if we get even one training point in B, then we are guaranteed that $[\hat{\theta}, \theta]$ lies within B, and so our error will be at most ϵ. Thus, our bad event is that none of our m training points fall in B. The probability of this happening is $(1-\epsilon)^m \leq e^{-\epsilon m}$ (using the fact that $1+x \leq e^x$ for all x), which can be made at most δ by picking $m = \frac{1}{\epsilon} \log \frac{1}{\delta}$. This completes the analysis.



5. (a) If $\text{err}(h) > \epsilon$, then the probability that h labels a single randomly drawn example correctly is less than $1-\epsilon$. The probability of getting k independent examples right is less than $(1-\epsilon)^k \leq e^{-\epsilon k}$ (using the fact that $1+x \leq e^x$ for all x). By picking $k = \frac{1}{\epsilon} \log \frac{1}{\delta'}$, this is at most δ' (which, as we'll see, will be picked based on our final desired δ).

   (b) Since A has a mistake bound of t, and it only updates when it makes a mistake, it can go through at most t hypotheses. This means that if we view our examples as consisting of $t+1$ blocks of size k, then we must have a block where we make no mistakes.

   (c) First we state the algorithm, then give the analysis. Our overall PAC learner works as follows:

      i. The learner draws a block of k random examples from D.

      ii. At the start of the block, we will assume that algorithm A, with its current state/hypothesis h_i, satisfies $\text{err}(h_i) \leq \epsilon$, and will use the examples in the block to test if this is indeed the case. With the block of examples, the learner feeds examples one by one to A. If A makes a mistake labeling one of the examples in the block, we start again at 5(c)i. If it does not, we stop and output h_i.

   Let h_i denote A's hypothesis (or state) after i mistakes. Based on our steps so far, the most natural idea for when we decide to stop and output h_i is to do so when h_i has gotten its block of k examples right.[1] So the hypothesis our PAC learner eventually

[1] Important pedagogical note: when we talk of the event E_i, we are saying something about the *examples*, not the hypothesis per se — we are saying that the ith block is misleading in the technical sense that all k examples happen to fall into h_i's "good area". That is, the function is fixed, and it is the examples that are random. It's important to understand this fact in this kind of analysis. If you like, the sample space consists of realizations of our random draws of examples from D.

outputs will be one of the h_i that A goes through. Define the event E_i to be the event that we output h_i such that $\text{err}(h_i) > \epsilon$. The PAC learner's failure event is precisely that it outputs a hypothesis with error greater than ϵ, and is thus described by $E = \cup_i E_i$. Here the union is taken over all the blocks we go through, which number at most $t + 1$. We want to ensure that $\mathbb{P}[E] \leq \delta$. By the union bound, we can say that $\mathbb{P}[E] \leq \sum_i \mathbb{P}[E_i]$. Recall that we picked k such that $\mathbb{P}[E_i] \leq \delta'$. Since there are at most $t + 1$ events E_i, picking $\delta' = \delta/(t+1)$, we have $\mathbb{P}[E] \leq (t+1)\delta' \leq \delta$, as desired. This proves the correctness of our PAC learner. The total number of examples used is $(t + 1)k = \frac{t+1}{\epsilon} \log \frac{t+1}{\delta}$.