Introduction to Machine Learning - COS 324

Homework Assignment 3 Solutions

1. State and re-prove the fundamental theorem for PAC learnability for finite hypothesis classes which we learned and proved in class. Explain and justify each transition in the proof from elementary facts in probability theory and combinatorics. You are welcome to use the handouts.

   Solution:

   Please refer to slides 8-11 in lect5.pdf for notations. H_B is the set of hypothesis which have error greater than ϵ on the distribution, i.e. $H_B = \{h \in H : \mathcal{L}_D(h) > \epsilon\}$. We wish to show that the probability of picking a set S of m points for which ERM returns a bad hypothesis is less than δ. The following are the crucial steps in the proof:

   (a) $\{S : \mathcal{L}_D(ERM(S,H)) > \epsilon\} \subseteq M = \{S : \exists h \in H_B, \mathcal{L}_S(h) = 0\}$.
   Proof: $h_S := ERM(S,H)$ satisfies $\mathcal{L}_S(h_S) = \min_{h \in H} \mathcal{L}_S(h) = 0$, since this is the realizable setting. So for a set S, the event $\mathcal{L}_D(h_S) = \mathcal{L}_D(ERM(S,H)) > \epsilon$ is the same as saying $h_S \in H_B$ which implies $\exists h \in H_B, \mathcal{L}_S(h) = 0$.

   (b) Bounding probability using Union bound

   $$\begin{aligned} D(\{S : \mathcal{L}_D(ERM(S,H)) > \epsilon\}) &= D(\{S : \exists h \in H_B, \mathcal{L}_S(h) = 0\}) \\ &= D(\cup_{h \in H_B} \{S : \mathcal{L}_S(h) = 0\}) \\ &\leq \sum_{h \in H_B} D(\{S : \mathcal{L}_S(h) = 0\}) \end{aligned}$$

   (c) For $h \in H_B$, $\mathcal{D}(\{S : \mathcal{L}_S(h) = 0\}) = (1 - \mathcal{L}_D(h))^m$
   Proof: Observe that $1 - \mathcal{L}_D(h)$ is simply the probability that h labels a point sampled randomly from the distribution D correctly. The m points in S are sampled independently from the distribution D. Using the fact the probability of independent events is the product of probabilities of the events, we get that the probability that h labels all m points correctly is $(1 - \mathcal{L}_D(h))^m$.

   (d) Combining (c) and (d), we get $D(\{S : \mathcal{L}_D(ERM(S,H)) > \epsilon\}) \leq |H_B|(1 - \mathcal{L}_D(h))^m \leq |H|(1 - \mathcal{L}_D(h))^m$. Choosing $m > \frac{\log |H|/\delta}{\epsilon}$ will make this quantity less than δ

2. Consider the following dataset:

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	y
1	1	0	0	0	1	0	1	1
1	0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	0	0
0	1	0	0	1	0	0	0	0
1	0	0	0	0	1	0	1	0
0	1	1	0	1	1	0	1	1
1	1	0	1	0	1	0	1	1
0	1	0	1	0	1	0	0	1
0	0	0	0	0	0	1	1	0

In this formulation, there are eight attributes (or features or dimensions), $x_1, \ldots, x_8$, each taking the values 0 or 1. The label (or class) is given in the last column denoted y and also takes the values 0 or 1. Notice that the label y is 1 if and only if x_2 and x_6 are both equal to 1. Since attributes and labels are $\{0, 1\}$-valued, we can write this rule succinctly as $y = x_2 x_6$. In general, such a product of any number of attributes is called a *monomial*. (This includes the "empty" monomial, which, being a product of no variables, is always equal to 1.)

Throughout the question, you may assume that the attributes and labels are all $\{0, 1\}$-valued. Also, let n be the number of attributes. Let m be the number of examples. For instance, $n = 8$ and $m = 9$ in the table above. Assume, as usual, that training and test examples are generated independently at random according to some unknown distribution.

(a) What is the total number of monomials that can be defined on n attributes?
Answer: 2^n. Observe that there is a bijection between monomials and subsets of the set $\{1, \ldots, n\}$. In other words, every subset of $\{1, \ldots, n\}$ corresponds to exactly one monomial.

(b) Describe a simple algorithm that, given a dataset, efficiently (in time which is polynomial in n and m) finds a consistent monomial, assuming that one exists.
Algorithm For any data point, the set of "on" variables is the set of all variables which are 1. Take the intersection of "on" variables for every data point with $y = 1$. A consistent monomial is the product of all these variables (lets call this m_S).
Proof of consistency: Clearly by construction m_S is consistent on all data points with $y = 1$. Any other consistent monomial m' can not have more variables than m_S, otherwise it won't be consistent on $y = 1$ points. So the value of monomial m' will be greater than the value of m_S. So if m_S is not consistent on $y = 0$ points, then m' can not be consistent on these points as well, which contradicts the realizable setting.

(c) Suppose you applied your algorithm to the dataset above, and that a consistent monomial was found. Use the bound derived in class to compute an upper bound

on the generalization error ϵ of this monomial. Derive a bound that holds with 95% confidence (so that $\delta = 0.05$).

Answer: We know that in the realizable setting if $m \geq \frac{\log |H|/\delta}{\epsilon}$, then generalization error is less than ϵ with probability at least $1 - \delta$. So this is also true for $m = \frac{\log |H|/\delta}{\epsilon}$. So generalization error $\leq \epsilon = \frac{\log |H|/\delta}{m}$. Setting $m = 9$, $|H| = 2^8$ and $\delta = 0.05$, we get an upper bound of 0.95 on the generalization error.

(d) Continuing the last question in which your algorithm is applied to data with $n = 8$ attributes, how many training examples would be needed to make sure that the generalization error of a consistent monomial is at most 10% with 95% confidence?

Answer: We need $m \geq \frac{\log |H|/\delta}{\epsilon}$ to ensure an error less than ϵ with probability at least $1 - \delta$. Plugging in $\epsilon = 0.1$, $\delta = 0.05$ and $|H| = 2^8$, we conclude that we need at least ~ 86 points.