MSRI Summer School on Mathematics of Machine Learning

Problem Set #1 Monday, July 29, 2019

Problem 1-1

Let the domain $X = \{0, 1\}^n$ so that each example $\mathbf{x}$ is a vector $(x_1, \ldots, x_n)$ of n boolean variables x_j (where, as usual, 1 is identified with TRUE, and 0 with FALSE). A monotone monomial is the conjunction (logical AND) of a subset of these variables; it is monotone because the variables appearing in the monomial cannot be negated. For instance,

$$c(\mathbf{x}) = x_2 \wedge x_7 \wedge x_{11}$$

is a concept (boolean function) defined by such a formula; it is equal to 1 if and only if x_2, x_7 and x_{11} are all equal to 1.

a. Let $\mathcal{C}$ be the concept class consisting of all monotone monomials (on n variables). Describe an efficient PAC-learning algorithm for $\mathcal{C}$. Specifically, be sure to show that your algorithm, with probability at least $1 - \delta$, will output an ϵ-good hypothesis (that is, with generalization error at most ϵ). Derive an exact bound on the number of examples m required to achieve this, and show that m is polynomial in n, $1/\epsilon$ and $1/\delta$. Finally, argue that your algorithm runs in time polynomial in n and m.

b. Use part (a) to derive a PAC learning algorithm for the class of all k-CNF (conjunctive normal form) formulas, that is, formulas which are the conjunction of any number of clauses, where each clause is the disjunction (logical OR) of at most k literals (variables that are either negated or unnegated). Here, k is a small constant, like 2 or 3. For instance,

$$(x_5 \vee \overline{x}_6) \wedge (x_{17}) \wedge (x_2 \vee x_6) \wedge (\overline{x}_9 \vee \overline{x}_1)$$

is an example of a 2-CNF formula, where $\overline{x}_j$ denotes the logical negation of x_j.

Problem 1-2

In class we proved a general result for analyzing learning algorithms that use a finite hypothesis space $\mathcal{H}$ by showing that, with high probability, for *every* hypothesis $h \in \mathcal{H}$, if h is consistent with the training set, then h is ϵ-good. However, when working with a particular learning algorithm A, we really only care about the *one* hypothesis returned by the algorithm, which we here denote h_A. Since we only care about that one hypothesis, and none of the others, it seems that we should be able to get a better bound on the number of examples needed for learning by only focusing on h_A.

Indeed the following argument shows that this is possible. In particular, the argument shows that, with probability at least $1 - \delta$, if h_A is consistent, then h_A is ϵ-good; that is,

$$\Pr\left[(h_A \text{ is consistent}) \Rightarrow (h_A\ \epsilon\text{-good})\right] \geq 1 - \delta,$$

or equivalently,

$$\Pr\left[(h_A \text{ is consistent}) \wedge (h_A\ \epsilon\text{-bad})\right] \leq \delta.$$

Thus, an algorithm that always finds a consistent hypothesis h_A is also guaranteed to be PAC. Note, however, that compared to what was given in lecture, the argument below yields a far better bound on the number of examples sufficient for learning, namely, $\ln(1/\delta)/\epsilon$, a bound that is entirely independent of $\mathcal{H}$.

As usual, the sample consists of random training examples $x_1, \ldots, x_m$ labeled according to the target concept c. The algorithm A takes these examples as input and outputs the hypothesis h_A (for simplicity, we assume A is deterministic). All of the probabilities below are with respect to the random choice of training examples. Here is the argument:

$$\begin{array}{lll}
\multicolumn{3}{l}{\Pr\left[(h_A \text{ is consistent}) \wedge (h_A\ \epsilon\text{-bad})\right]} \\
= & \Pr\left[h_A \text{ is consistent} \mid h_A\ \epsilon\text{-bad}\right] \cdot \Pr\left[h_A\ \epsilon\text{-bad}\right] & \text{definition of conditional probability} \\
\leq & \Pr\left[h_A \text{ is consistent} \mid h_A\ \epsilon\text{-bad}\right] & \text{since } \Pr\left[h_A\ \epsilon\text{-bad}\right] \leq 1 \\
= & \Pr\left[h_A(x_1) = c(x_1) \wedge \cdots \wedge h_A(x_m) = c(x_m) \mid h_A\ \epsilon\text{-bad}\right] & \\
 & & \text{meaning of consistency} \\
= & \prod_{i=1}^{m} \Pr\left[h_A(x_i) = c(x_i) \mid h_A\ \epsilon\text{-bad}\right] & \text{by independence of the } x_i\text{'s} \\
\leq & \prod_{i=1}^{m} (1 - \epsilon) & \text{an } \epsilon\text{-bad hypothesis is correct} \\
 & & \quad \text{with probability} \leq 1 - \epsilon \\
= & (1 - \epsilon)^m & \\
\leq & e^{-\epsilon m} & \text{since } 1 + x \leq e^x \text{ for all } x \\
\leq & \delta &
\end{array}$$

where the last inequality holds if

$$m \geq \frac{\ln(1/\delta)}{\epsilon}.$$

This result, which shows that the number of examples need not depend on the complexity or size of the hypothesis space, should seem too good to be true. *And indeed it is!* The claimed result is actually *false*, which means that the argument above must be wrong.

a. What are the mistakes or false steps in the argument above? (There are at least two.) To answer this, be sure you understand which are the random variables, and what they each depend on.

b. In general, to avoid this kind of fallacious argument, it is important to specify the hypotheses we are working with *before* any data has been randomly generated. How does the false argument above fall into that trap? And how did the proof given in class avoid that?

Problem 1-3

This problem explores a general method for bounding the error when the hypothesis space is infinite, a different approach from the ones being given in lecture.

Some algorithms output hypotheses that can be represented by a small number of examples from the training set. For instance, suppose the domain is $\mathbb{R}$ and we are learning a (positive) half-line of the form $x \geq a$ where a is a threshold defining the half-line. A simple algorithm chooses the leftmost positive training example and outputs the half-line defined by using this point as a threshold, which is clearly consistent with the training data. Thus, in this case, the hypothesis can be represented by just one of the training examples.

More formally, let F be a function mapping labeled examples to concepts, and assume that algorithm A, when given training examples $(x_1, c(x_1)), \ldots, (x_m, c(x_m))$ labeled by some unknown $c \in \mathcal{C}$, chooses k indices $i_1, \ldots, i_k \in \{1, \ldots, m\}$ and outputs a hypothesis $h = F((x_{i_1}, c(x_{i_1})), \ldots, (x_{i_k}, c(x_{i_k})))$ which is consistent with all m training examples. In a sense, the algorithm has "compressed" the sample down to a sequence of just k of the m training examples. (We assume throughout that $m > k$.) For instance, in the example of half-lines: $k = 1$; i_1 is the index of the leftmost positive training example; and the function F returns a half-line hypothesis with threshold x_{i_1}, that is, the hypothesis $h = F((x_{i_1}, c(x_{i_1})))$ which classifies x positive if and only if $x \geq x_{i_1}$.

a. Give such an algorithm for axis-aligned hyper-rectangles in $\mathbb{R}^n$ with $k = O(n)$. (An axis-aligned hyper-rectangle is a set of the form $[a_1, b_1] \times \cdots \times [a_n, b_n]$, and the corresponding concept is the binary function that is 1 for points inside the rectangle and 0 otherwise.) Your algorithm should run in time polynomial in m and n.

b. Returning to the general case, assume as usual that the examples are chosen at random from some distribution D. Also assume that the size k is fixed. Argue carefully that, with probability at least $1 - \delta$, if the output hypothesis h is consistent with all m random training examples, then its error will be

$$\text{err}_D(h) \leq O\left(\frac{\ln(1/\delta) + k \ln m}{m - k}\right).$$

Side note: A difficult, long-standing open problem asks if it is always possible to find such a "compression scheme" whose size k is equal to (or proportional to) the VC-dimension d of the target class $\mathcal{C}$.

Hint for part (b): Fix $i_1, \ldots, i_k$ as in the statement of the problem, and bound the probability of the hypothesis defined by exactly these k examples being consistent (with all m training examples) but having generalization error greater than ϵ. Then apply the union bound.