

COS 511: Theoretical Machine Learning

Homework #3
Learning models & Chernoff

Due:
March 11, 2019

Handwritten homeworks: Although it is certainly okay to handwrite your homeworks (even if submitting electronically), we want to emphasize again that they must be neat and easily readable by the graders. Going forward, we are asking that handwritten homeworks *not* be written in cursive.

Problem 1

In the *batch version* of the PAC model, which is the one that we have been studying in class, the learner must specify how many examples it requires *before* seeing any data. Thus, before learning begins, the learning algorithm specifies the number of examples needed, and cannot later ask for more examples.

In contrast, in the *oracle version* of the PAC model, the learner is not provided with a fixed batch of examples, but is instead provided with an example “oracle” EX. The learner requests one example at a time from EX. Each call provides the learner with a single example. As usual, each example x is selected at random according to the distribution D , and both x and its label $c(x)$ are provided to the learner. Thus, in this model, the learner is provided with ϵ , δ and the oracle EX, and can request as many examples as it wishes from EX. As usual, the hypothesis that the learner outputs must have error at most ϵ with probability at least $1 - \delta$. Moreover, the total number of examples requested must always be bounded by a polynomial.

So, the difference between these two models is that in the batch version, the learner must decide ahead of time how many examples it needs (with knowledge of ϵ and δ), while in the oracle version, it can dynamically decide how many examples it needs based on the data received so far. This problem explores this difference for a simple example studied on previous problem sets.

As on homework #1 (problem 1), let the domain be $X = \mathbb{R}$, and let \mathcal{C}_s be the class of concepts defined by unions of s intervals. That is, each concept c is defined by real numbers $a_1, b_1, \dots, a_s, b_s$ where $c(x) = 1$ if and only if $x \in [a_1, b_1] \cup \dots \cup [a_s, b_s]$. For the purposes of this problem, “efficient” means that the time and sample requirements are polynomial in $1/\epsilon$, $1/\delta$ and s .

On that previous problem, you were asked to show that, in the batch version, there exists an efficient algorithm that learns the class \mathcal{C}_s for every s when s is *known* ahead of time to the learner.

- a. [10] Still in the batch version, assume now that the learner does *not* know s ahead of time, so that the number of examples needed is only a function of ϵ and δ . In this case, prove that there is no algorithm (whether efficient or not, and regardless of the hypothesis space used) that can PAC-learn the class \mathcal{C}_s for more than finitely many values of s . In other words, for any algorithm A (taking as input $m(\epsilon, \delta)$ random examples and outputting a hypothesis from any space), prove that there exists some number s_0 such that for all $s \geq s_0$, A does *not* PAC-learn \mathcal{C}_s .

b. [15] Turning now to the oracle version, let us continue to assume that the learner does not know s ahead of time. Describe a single, efficient algorithm that learns the class \mathcal{C}_s for every s even though s is not known by the learner. Then show that your algorithm has all of the following, desired properties. (It is okay on your solution to present these in a different order, if you find that easier.)

- (i) Although s is not known at the beginning of the learning process, argue that your algorithm *always* halts, and that when it does halt, the total number of examples requested has not exceeded a polynomial in $1/\epsilon$, $1/\delta$ and s . Give a “big-Oh” expression for the number of examples needed.
- (ii) Prove that your algorithm is PAC, that is, that it finds a hypothesis with error at most ϵ with probability at least $1 - \delta$. Be especially careful that your argument proves that the *total* probability of failing to find an ϵ -good hypothesis is at most δ .
- (iii) Argue that the total running time of your algorithm is bounded by a polynomial in $1/\epsilon$, $1/\delta$ and s .

(For this problem, do not make any extraneous assumptions about the distribution D , for instance, of the kind that had earlier been allowed on problem 1 of homework #1.)

Problem 2

[15] Let X_1, \dots, X_m be m random variables that are independent, and which each take values in $[0, 1]$, but which are *not* necessarily identically distributed. Let $p_i = \mathbf{E}[X_i]$, and let us also define

$$\hat{p} = \frac{1}{m} \sum_{i=1}^m X_i$$
$$p = \frac{1}{m} \sum_{i=1}^m p_i.$$

For any $q > p$, prove that

$$\Pr[\hat{p} \geq q] \leq \exp(-\text{RE}(q \parallel p)m).$$

Problem 3

Let D be a distribution over $X \times \{0, 1\}$, and let $S = \langle (x_1, y_1), \dots, (x_m, y_m) \rangle$ be a random sample from D . Let

$$\begin{aligned}\text{err}(h) &= \Pr_{(x,y) \sim D} [h(x) \neq y] \\ \widehat{\text{err}}(h) &= \frac{1}{m} \sum_{i=1}^m \mathbf{1}\{h(x_i) \neq y_i\}.\end{aligned}$$

For simplicity, we will assume that \mathcal{H} is finite, although the results of this problem can be carried over to the infinite case. Note that none of the results depend on $|\mathcal{H}|$.

Let \hat{h} and h^* be the hypotheses in \mathcal{H} with minimum training error and generalization error, respectively:

$$\begin{aligned}\hat{h} &= \arg \min_{h \in \mathcal{H}} \widehat{\text{err}}(h) \\ h^* &= \arg \min_{h \in \mathcal{H}} \text{err}(h).\end{aligned}$$

Be sure to keep in mind that, unlike h^* , \hat{h} is a *random variable* that depends on the random sample S .

- a. [10] Prove that

$$\mathbb{E} [\widehat{\text{err}}(\hat{h})] \leq \text{err}(h^*) \leq \mathbb{E} [\text{err}(\hat{h})].$$

- b. [10] Prove that, with probability at least $1 - \delta$,

$$|\widehat{\text{err}}(\hat{h}) - \mathbb{E} [\widehat{\text{err}}(\hat{h})]| \leq O \left(\sqrt{\frac{\ln(1/\delta)}{m}} \right).$$

Give explicit constants, and be sure to end up with a result that does not depend on $|\mathcal{H}|$.

- c. [5] Explain in words the meaning of what you proved in both of the preceding parts, and how we would expect training error to compare to test error when using a machine learning algorithm on actual data.