Machine Learning 4771: Homework 2

(15% of the grade)

Due on February 26, 2:40pm

Note: You are not allowed to submit an answer you cannot explain in person. Please take this seriously: You can be quizzed on your homework.

Late assignments will not be accepted.

Problem 1

(Regression) Given a set of training examples $(x_1, y_1), \ldots, (x_N, y_N) \in \mathbf{R}^n \times \mathbf{R}$, let $\mathbf{X}$ be a $N \times n$ matrix with row i corresponding to example x_i, and let $\mathbf{y} = [y_1, \ldots, y_N]^T$ (column vector containing the N training labels). Consider the problem of finding $\mathbf{w} \in \mathbf{R}^n$ minimizing

$$\|\mathbf{X} \cdot \mathbf{w} - \mathbf{y}\|^2 + \|\mathbf{w}\|^2,$$

where $\|\cdot\|$ is the Euclidean norm.

Does the regularization term $\|\mathbf{w}\|^2$ force the solution vector $\mathbf{w}$ to have a small number of non-zero entries? Explain why or why not.

Solution: The answer is no, due to the nature of quadratic loss. When there are several correlated features with a significant effect on y, ridge regression tends to "share" the coefficient value among them (which results in a smaller L_2 penalty than putting a large value on a small subset of them). If we use the L_1 penalty $\|\mathbf{w}\|_1 = \sum_{i=1}^{n} |w_i|$ instead, there will be a tendency to zero out most (if not all but one) correlated features, resulting in a sparse coefficient vector $\mathbf{w}$.

Problem 2

Describe a concept class C for which the halving algorithm is not optimal, i.e., you would get a better worst-case mistake bound by *not* going with the majority vote among the concepts in C consistent with examples observed so far. Explain your answer.

Solution: Example 2 in the paper below.

Problem 3

Describe a concept class C where the worst-case mistake bound of the halving algorithm ($\log|C|$) is not tight. Explain your answer.

Solution: Example 1 in the paper below.

Bonus: You will receive bonus points for examples significantly different from the ones appearing in the following paper:

> Nick Littlestone, Learning Quickly When Irrelevant Attributes Abound: A New Linear-Threshold Algorithm, *Machine Learning*, 2(4): 285–318, 1998.

Copying any text from the paper will deterministically lead to a quiz. You can use examples from the paper, but you have to explain them yourself.

Problem 4

Describe a concept class C where *no* randomized algorithm can do better than $(\log|C|)/2$ mistakes in expectation (over the randomness in the algorithm). Explain your answer.

Solution: Let C be the class of monotone disjunctions on n boolean variables and consider the following sequence of n examples: examle x_t has t-th bit set to 1 and all other bits set to 0, for t from 1 to n. Consider any randomized algorithm making binary predictions on this sequence. Let p_t be the probability that the algorithm outputs 1 in trial t. Imagine that the true labeling of these examples given by $\mathbf{1}(p_t \leq 1/2)$, so the label of x_t is 1 if $p_t \leq 1/2$ and 0 otherwise; this labeling is certainly consistent with a disjunction. The expected number of mistakes that the algorithm makes on the sequence is $\sum_{t=1}^{n} \max\{p_t, 1-p_t\} \geq n/2$.

This lower bound can be matched by an algorithm that outputs according to a random OR function, which includes each variable with probability 1/2. Thus each $p_t = 1/2$, and the expected number of mistakes is $n/2$. ∎

Problem 5

An online learning algorithm is *lazy* if it changes its state only when it makes a mistake. Show that for any deterministic online learning algorithm A achieving a mistake bound M with respect to a concept class C, there exists a lazy algorithm A' that also achieves M with respect to C.

Recall the definition: Algorithm A has a mistake bound M with respect to a learning class C if A makes at most M mistakes on *any* sequence that is consistent with a function in C.

Solution: Let A' be the lazy version of A (has the same update rule as A on mistakes and doesn't change its state on correctly labeled examples). We will show that A' has a mistake bound M with respect to C. Assume that there exists a sequence of examples on which A' makes $M' > M$ mistakes. Cross out all examples on which A' doesn't make a mistake, and let s denote the resulting sequence. Both A and A' behave identically on s, and A makes at most M mistakes on any sequence of examples, including s. This leads to the desired contradiction. (Obviously, if the original sequence is consistent with a concept in C, so is s.) ∎

Problem 6

Prove mistake bounds for the following two modifications to WINNOW, for the class of monotone disjunctions over n boolean variables. (Such a disjunction is a boolean function of the form $\bigvee_{i \in S} x_i$ for some subset $S \subseteq \{1, \ldots, n\}$. Here $\bigvee$ denotes the boolean OR function.) Assume that all labels are consistent with a monotone disjunction.

Given an example $x = (x_1, \ldots, x_n)$,

- If the algorithm makes a mistake on a negative (predicts 1 when the correct label of x is 0), then for each x_i equal to 1 set $w_i = 0$.
- Whenever the correct label is 0 (regardless of whether the algorithm made a mistake or not), set $w_i = 0$ for each x_i equal to 1.

A formal proof is expected.

Solution:

- On each mistake on a positive, the weight of at least one of r relevant variables in the target disjunction must be doubled (otherwise the example would not be positive). Thus each relevant variable can be doubled at most $\lceil \log n \rceil$ times, and the total number of mistakes due to mistakes on positives is at most $r\lceil \log n \rceil$. (This part of the analysis didn't change.)

  On each mistake on a positive, the total weight W increases by at most n (since we predicted 0). On each mistake on a negative, W decreases by at least n. Since the total weight began at n is always positive, the number of mistakes on negatives is never more than the number of mistakes on positives plus 1. Thus the mistake bound is $2r\lceil \log n \rceil + 1$.

- The mistake bound is the same (see Problem 5).