

# 15-859(B) Machine Learning Theory

Homework # 2

Solutions

---

## Exercises:

1. **A bad modification to Winnow.** Suppose that we modify Winnow so that it doubles its weights on positive examples even when it did *not* make a mistake. Show how this can cause the algorithm to make an unbounded number of mistakes, even if all examples *are* consistent with some disjunction.

Solution: Consider 2-bit examples and the target concept  $x_1$ . If we begin with the hypothesis  $x_1 + x_2 \geq 2$ , then alternating between the positive example 11 and the negative example 01 will cause us to make an unbounded number of mistakes.

2. **Balanced Winnow.** Here is a variation on the Winnow algorithm, called *Balanced Winnow*. First of all, we introduce a fake variable  $x_0$  which is set to 1 in every example. For each variable  $x_i$  ( $0 \leq i \leq n$ ), and each output value  $y$  (as usual,  $y \in \{-, +\}$ , but you can also use this algorithm for multi-valued outputs) we have a weight  $w_{iy}$ . All weights are initialized to 1. In addition, we are given parameters  $\alpha > 1$  and  $\beta < 1$ . The algorithm proceeds as follows:

- (a) Given example  $x$ , predict the label  $y$  such that  $\sum_i x_i w_{iy}$  is largest.
- (b) If the algorithm makes a mistake, predicting  $y'$  when then correct answer is  $y$ , then for each  $x_i = 1$ , multiply the weight  $w_{iy}$  by  $\alpha$ , and multiply  $w_{iy'}$  by  $\beta$ .

Using  $\alpha = 3/2$  and  $\beta = 1/2$ , prove that as with the standard Winnow algorithm, this algorithm makes at most  $O(r \log n)$  mistakes on any disjunction (OR-function) of  $r$  variables.

Solution: First of all, notice that the total sum of all weights begins at  $2(n+1)$ , and by using  $\alpha = 3/2$  and  $\beta = 1/2$ , this sum *never* increases. That is because every time we make a mistake, the amount we subtract away from the total weight (on the side we predicted) is always at least as much as the amount we add to the total weight (on the side of the correct answer). Each mistake made on a positive example rewards some relevant weight  $w_{i+}$  (multiplies it by  $\alpha$ ), and the relevant weights  $w_{i+}$  are never penalized. Therefore, the number of mistakes made on positive examples is at most  $r(1 + \log_\alpha(2n+2)) = O(r \log n)$ .

We now argue that the number of mistakes made on negative examples cannot be much more than the number of mistakes made on positive examples. To do this, we consider  $x_0$ . Let  $M_n$  be the number of mistakes on negatives, and  $M_p$  be the number of mistakes on positives. Notice that  $w_{0-} = \alpha^{M_n} \beta^{M_p}$ . Also,  $\alpha^2 \beta > 1$ . So,  $w_{0-} > \alpha^{M_n - 2M_p}$ , which cannot be more than  $2(n+1)$ . Therefore,  $M_n < 2M_p + O(\log n)$ .

3. **About  $\delta$ .** In the first lecture, we argued that if we had an algorithm  $\mathcal{A}$  with at least a  $\frac{1}{2}$  chance of producing a hypothesis of error at most  $\epsilon/2$ , we could convert it into an algorithm  $\mathcal{B}$  that has a  $1 - \delta$  probability of producing a hypothesis of error at most  $\epsilon$ . The reduction is that we first run  $\mathcal{A}$  for  $N = \lg \frac{2}{\delta}$  times (so with probability at least  $1 - \delta/2$ , at least one of the  $N$  hypotheses produced has error at most  $\epsilon/2$ ), and we then test the  $N$  hypotheses produced on a new test set, choosing the one that performs best. Use Chernoff bounds to analyze this second step and finish the argument. That is, assuming that at least one of  $N$  hypotheses has error at most  $\epsilon/2$ , give an explicit bound (without  $O$  notation) on a size for the test set that is sufficient so that with probability at least  $1 - \delta/2$ , the hypothesis that performs best on the test set has error at most  $\epsilon$ .

Solutions: It is enough to have a test set such that with probability  $1 - \delta/2$ , any hypotheses of true error at most  $\epsilon/2$  has empirical error at most  $3\epsilon/4$ , and any of true error greater than  $\epsilon$  has empirical error greater than  $3\epsilon/4$ . By Chernoff bounds, we just need a number of examples  $m$  such that  $N \cdot \max\{e^{-m(\epsilon/2)(1/2)^2/3}, e^{-m\epsilon(1/4)^2/2}\} \leq \delta/2$ . So,  $m = \frac{32}{\epsilon} \ln(2N/\delta)$  is sufficient.

#### Problems:

4. **Tracking a moving target.** Here is a variation on the deterministic Weighted-Majority algorithm, designed to make it more adaptive.
- Each expert begins with weight 1 (as before).
  - We predict the result of a weighted-majority vote of the experts (as before).
  - If an expert makes a mistake, we penalize it by dividing its weight by 2, but *only* if its weight was at least  $1/4$  of the average weight of experts.

Prove that in any contiguous block of trials (e.g., the 51st example through the 77th example), the number of mistakes made by the algorithm is at most  $O(m + \log n)$ , where  $m$  is the number of mistakes made by the best expert *in that block*, and  $n$  is the total number of experts.

Solution: Let  $W_{init}$  be the total weight at the beginning of the interval and  $W_{final}$  be the total weight at the end of the interval.

First, notice that all weights are at least  $1/8$  of the average. We can see this by induction: the average never increases, so the statement holds for weights that were not lowered in the last round. Also, if a weight was lowered, then it must have been at least  $1/4$  of the old average, so it is now at least  $1/8$  of the old average which is at least  $1/8$  of the new average.

This means that the weight of the best expert at beginning of the interval is at least  $W_{init}/(8n)$ , and therefore by end of the interval it is at least  $(1/2)^m W_{init}/(8n)$ .

Also, on each mistake, at most  $W/4$  of the total weight is fixed. So at least  $(W/2 - W/4) = W/4$  gets cut in half. In other words,  $W/8$  is removed from the total weight. This means  $W_{final} < W_{init}(7/8)^M$ .

The bound results from solving  $(1/2)^m W_{init}/(8n) \leq W_{init}(7/8)^M$ .

5. **[More on margins]** Algorithms such as Perceptron and SVMs do well when data is linearly separable by a large margin  $\gamma$ .<sup>1</sup> For example, the Perceptron algorithm makes at most  $O(1/\gamma^2)$  mistakes; so, if the margin  $\gamma$  is large compared to  $1/\sqrt{n}$ , then the number of mistakes is small compared to the VC-dimension bound. On the other hand, it is also possible for the margin bound to be much worse than the VC-dimension bound. Give an example of  $O(n)$  points in  $\{0, 1\}^n$  that *are* linearly separable but where the Perceptron algorithm would make an exponential number of mistakes. For concreteness, let us consider a version of the Perceptron algorithm that does not normalize the examples to all have Euclidean length 1: it just adds or subtracts the given positive/negative example from the weight vector on a mistake (this will make things conceptually easier). In particular, with this version the weights are always integral. So, it is sufficient to come up with a set of  $O(n)$  linearly-separable examples in  $\{0, 1\}^n$  such that the only integral-weight linear separator has exponential-sized weights.

Hint: your example will also prove that the Perceptron algorithm is not a legal solution to problem 4 on hwk 1.

Solution: Define the linear separator to be  $w_1x_1 + w_2x_2 + \dots + w_nx_n > 0$ . Now consider the following examples:

example	label	note
1000000000	+	$w_1 \geq 1$
0100000000	+	$w_2 \geq 1$
1110000000	-	$w_3 \leq -2$
1101000000	-	$w_4 \leq -2$
0011100000	+	$w_5 \geq 4$
0011010000	+	$w_6 \geq 4$
0000111000	+	$w_7 \leq -8$
0000110100	+	$w_8 \leq -8$
...	...	...

At the end, the largest weight has magnitude at least  $2^{n/2}$ .

If we want to talk directly about margins, notice that if we tweak  $w$  by setting  $w_1 = -1$ , then it no longer is consistent, and yet we have only made an exponentially small change in the angle of  $w$ . Or, to put it another way, if  $x$  is the first example, then  $w \cdot x/|w|$  is exponentially small, because  $|w|$  is exponentially large.

---

<sup>1</sup>As in Lecture 4, defining margin as the minimum distance of any example to the separator when examples have been normalized to unit Euclidean length.