15-859(B) Machine Learning Theory

Homework # 2 **Due: February 11, 2008**

Groundrules: Same as before. You should work on the exercises by yourself but may work with a partner on the problems (just write down who you worked with). Also if you use material from outside sources, say where you got it.

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.

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 α, and multiply $w_{iy'}$ by β.

   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.

3. **About δ.** 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 ϵ. 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 ϵ.

Problems:

4. **Tracking a moving target.** Here is a variation on the deterministic Weighted-Majority algorithm, designed to make it more adaptive.

   (a) Each expert begins with weight 1 (as before).

   (b) We predict the result of a weighted-majority vote of the experts (as before).

   (c) 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.

5. [**More on margins**] Algorithms such as Perceptron and SVMs do well when data is linearly separable by a large margin γ.[1] For example, the Perceptron algorithm makes at most $O(1/\gamma^2)$ mistakes; so, if the margin γ 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.

[1] 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.