15-859(A) Machine Learning Theory

Homework # 6 **Due: April 27, 2004**

Fun with Winnow. You may want to review the lecture notes for 1/22.

Note: This is the last homework assignment. Projects are due on the last day of class. More info on the take-home final later.

Problems:

1. 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 ($y \in \{-, +\}$, but you can also use this 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.

2. Show for $\alpha = 1 + \epsilon$ and $\beta = 1 - \epsilon$, that Balanced-Winnow approximates the constraints in the maxent algorithm. You may assume for simplicity that there are only two labels, positive and negative, and $\epsilon \leq 1/4$. Specifically,

   (a) If M_p is the number of mistakes made on positive examples, and M_n is the number of mistakes on negative examples, then

   $$M_p \leq M_n(1 + O(\epsilon)) + O(\frac{1}{\epsilon} \log n)$$

   and vice-versa. Hint: think about the fake variable x_0.

   (b) Show this implies that if N_p is the true number of positive examples seen, and $\hat{N}_p$ is the number of times the algorithm has predicted positive, then

   $$\hat{N}_p \leq N_p(1 + O(\epsilon)) + O(\frac{1}{\epsilon} \log n),$$

   and similarly for negative examples.

   (c) Show that the same statements hold if we only consider the subset of examples having $x_i = 1$. That is, the number of mistakes on positives having $x_i = 1$ is approximately the number of mistakes on negatives having $x_i = 1$, implying the same statement about the number of times the algorithm *predicts* positive given that $x_i = 1$ versus the number of true positives with $x_i = 1$.