# 8803 Machine Learning Theory

**Homework # 2** **Due: October 4th 2011**

---

This homework is due by the start of class on October 4th. You can either submit the homework via the course page on T-Square or hand it in at the beginning of the class on October 4th.

**Groundrules:**

- Your work will be graded on correctness, clarity, and conciseness.
- You may collaborate with others on this problem set and consult external sources. However, you must *write your own solutions* and *list your collaborators/sources* for each problem.

**Problems:**

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 $\alpha$, 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.

3. **VCdimension of axix-parallel rectangles.** What is the VC-dimension $d$ of axis-parallel rectangles in $R^3$? Specifically, a legal target function is specified by three intervals $[x_{min}, x_{max}]$, $[y_{min}, y_{max}]$, and $[z_{min}, z_{max}]$, and classifies an example $(x, y, z)$ as positive iff $x \in [x_{min}, x_{max}]$, $y \in [y_{min}, y_{max}]$, and $z \in [z_{min}, z_{max}]$.

**Extra Credit:**

4. **PNF.** The class of $k$-term PNF is just like $k$-term DNF except that we use a *parity function* in place of the *OR* function. For instance, the following is a 2-term PNF:

   $$x_1 x_3 \bar{x}_5 \oplus x_2 x_4.$$

   This function is positive on examples 11100 and 11011, and is negative on examples 00000 and 11110.

Give an algorithm that learns the class of 2-term PNF over $\{0,1\}^n$ in the mistake-bound model. Your algorithm should have a mistake bound polynomial in $n$ and should be efficient (running in polynomial time per example) too. Your algorithm need *not* use 2-term PNF as its hypothesis representation.