CS269: Machine Learning Theory
Problem Set 2
Due November 29, 2010

Ground Rules:

- This problem set is due at the beginning of class on November 29. Please bring a hard copy of your solutions with you to class. Late assignments may be submitted to Edna Todd in 4532N Boelter Hall, and will be penalized 25%. No assignments will be accepted more than 24 hours late.
- All solutions must be typed and printed out. Solutions that are not typed will be penalized 20%, and unreadable answers will not be graded.
- You will be graded on both correctness and clarity. Be concise and clear, especially when writing proofs! If you cannot solve a problem completely, you will get more partial credit if you identify the gaps in your argument rather than trying to cover them up.
- You are strongly encouraged to discuss the problem set with other students in the class. You may also consult other references, such as textbooks or online notes. However, you must write out the solutions to the problems on your own. Your answers should be in your own words, and you should understand the answers you are submitting. Additionally, you must list all of your collaborators on your assignment and properly credit any sources that you consult.

Problems:

1. **Variations on Winnow** *(30 points total)*

   In class, we showed that when $\epsilon = 1$, Winnow achieves a mistake bound of $2 + 3r(1 + \log n)$ on the class of monotone disjunctions, where n is the total number of variables (i.e., features) and r is the number of "relevant" variables (i.e., variables that appear in the target disjunction).

   (a) Assuming a perfect target disjunction exists, it is possible to achieve a better regret bound by updating the weights of irrelevant variables more aggressively. In particular, suppose we modify Winnow so that if a mistake is made on an example that should have been labeled negative (i.e., when $y_t = 0$), we set the weights of all positive variables to zero. Give a mistake bound for this algorithm. Briefly describe how to modify proofs we discussed in class to derive this bound. You **do not** need to provide a full proof, only a description of the modifications. *(10 points)*

   (b) Would you expect this variation to perform better or worse than the original Winnow if the assumption of a perfect target function were violated? Justify your answer in **one or two sentences**. *(5 points)*

   (c) Suppose we modified the version of Winnow that we analyzed in class so that it doubled the weights on positive variables whenever $y_t = 1$, even on rounds in which no mistake was made. Show how an adversary could force this algorithm to make an unbounded number of mistakes, even if there is a monotone disjunction that perfectly labels the data. *(15 points)*

2. **No Perfect Targets** *(20 points total)*

Define the hinge loss of a linear threshold function represented by its normal vector $\vec{w}$ on a sequence of points $(\vec{x}_1, y_1), \cdots, (\vec{x}_T, y_T)$ as

$$\mathfrak{L}_\gamma(\vec{w}) = \sum_{t=1}^{T} \max(0, \gamma - (\vec{w} \cdot \vec{x}_t) y_t).$$

In class, we derived a mistake bound for the Perceptron algorithm under the assumption that a perfect target function with a margin of γ exists. In this problem, we will derive an alternate guarantee for the Perceptron algorithm that holds even when there is no perfect target. Suppose that we run the basic Perceptron algorithm on a sequence of T points $(\vec{x}_1, y_1), \cdots, (\vec{x}_T, y_T)$, such that $||\vec{x}_t|| = 1$ and $y_t \in \{-1, 1\}$ for all t. Define

$$H = \min_{\vec{u}:||\vec{u}||=1} \mathfrak{L}_\gamma(\vec{u}) \ .$$

State an upper bound for the **number of mistakes** made by the Perceptron algorithm on this sequence in terms of γ and H. Your bound should reduce to $1/\gamma^2$ when $H = 0$. Briefly describe how to modify proofs that we discussed in class to derive this bound. As before, you **do not** need to provide a full proof, only a description of the modifications.

3. **The Doubling Trick** *(20 points)*

The original regret bound that we derived for Randomized Weighted Majority relied on the assumption that the algorithm has access to an upper bound L^* on the loss of the best expert in advance, and can use this upper bound to choose a value of the learning rate η. If the number of rounds T is known in advance, we can simply set $L^* = T$. However, it is not always realistic to assume T is known. One common technique used to get rid of these assumptions is the so-called "Doubling Trick". We start by making a guess about the value of L^*, say $L^* = g$, and run RWM with η tuned with this value. If we ever get to a point where the loss of every expert is at least g, we reset our weights and start running RWM from scratch using a new guess of $2g$ for L^*. If the subsequent loss of every expert gets this high, we reset our weights again and start running RWM from scratch with $L^* = 4g$, and so on. Show that the regret of this algorithm is almost as good (in terms of dependence on T) as the regret of RWM when a bound L^* is known in advance.

4. **Variations on Adaboost** *(30 points total)*

   (a) Suppose that we run Adaboost for T rounds and find that the weak learning assumption holds at each round. That is, for every $t \in \{1, \cdots, T\}$, $\epsilon_t \leq 1/2 - \gamma$. Show that it is never the case that $h_t = h_{t+1}$. *(10 points)*

   (b) Suppose that we decided to simplify Adaboost by setting $\alpha_t = \alpha$ for a fixed value α on each round. Assume that the weak learning assumption will always hold, so for every $t \in \{1, \cdots, T\}$, $\epsilon_t \leq 1/2 - \gamma$. What is the best value of α to use? Briefly defend this choice of α. *(10 points)*

   (c) Give a bound on the empirical error of the function output by this modified version of Adaboost if the weak learning assumption holds. Your bound should be in terms of γ and T only (and not, for example, the ϵ_t values). You need not provide a full proof of this bound, but briefly discuss how the proof differs from the derivation of the bound on empirical error for standard Adaboost. *(10 points)*