CS260: Machine Learning Theory
Problem Set 3
Due Monday, November 7, 2011

Ground Rules:

- This problem set is due at the beginning of class on November 7. Slightly late assignments should be submitted directly to the grader, and will be penalized 25% (i.e., 25 points). No assignments will be accepted more than 24 hours late.
- You are strongly encouraged to discuss the problem set with other students in the class, as long as you follow the rules outlined in the course academic honesty policy.
- All solutions must be typed; LaTeX is strongly recommended. Hand-written solutions will be penalized 25%, and unreadable answers will not be graded.
- You will be graded on both correctness and clarity. Be concise and clear, especially when writing proofs. **Please make sure you define any notation that you use or variables that you introduce!!** You will be graded on what you wrote, not what you intended to write.

Problems:

1. **The Normalized Perceptron** *(25 points total)*

   In the basic version of the Perceptron algorithm that we covered in class, every time a mistake is made on round t, weights are updated according to the formula

   $$\vec{w}_{t+1} \leftarrow \vec{w}_t + y_t \vec{x}_t \ .$$

   Consider a modified version of the Perceptron algorithm in which, every time a mistake is made on round t, weights are instead updated using the rule

   $$\vec{w}_{t+1} \leftarrow \vec{w}_t + y_t \frac{\vec{x}_t}{||\vec{x}_t||} \ .$$

   Call this the *Normalized Perceptron*. Note that weights are still updated only on rounds in which a mistake is made.

   (a) State and prove a mistake bound for the Normalized Perceptron algorithm that holds under the assumption that there exist a vector $\vec{u}$ and value $\lambda > 0$ such that $||\vec{u}|| = 1$ and for all t,

   $$y_t \left(\frac{\vec{x}_t}{||\vec{x}_t||} \cdot \vec{u} \right) \geq \lambda \ .$$

   *Hint: You **do not** need to modify the Perceptron mistake bound proof that we discussed in class to solve this problem. (15 points)*

(b) Compare the mistake bounds of the Perceptron and the Normalized Perceptron under the typical Perceptron assumptions (i.e., the assumption there exist a vector $\vec{u}$ and values $\gamma > 0$ and $D > 0$ such that $||\vec{u}|| = 1$ and for all t, $y_t\left(\vec{x}_t \cdot \vec{u}\right) \geq \gamma$ and $||\vec{x}_t|| \leq D$). Does one algorithm yield a tighter bound? *(10 points)*

2. **Perceptron with No Perfect Target** *(30 points)*

For any fixed value $\gamma > 0$, we can 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

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

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 alternative 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

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

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 $H = 0$. Describe how to modify proofs that we discussed in class to derive this bound. You **do not** need to provide a full proof, only a description of the modifications.

3. **Variations on Winnow** *(45 points total)*

In class, we showed that when $\beta = 1$, Winnow achieves a mistake bound of $2 + 3k(1 + \log n)$ on the class of monotone disjunctions, where n is the total number of variables (i.e., features) and k 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. 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. *(25 points)*

(You might ask why we would not always use this version of Winnow if the mistake bound is better. Well, would you expect this variation to perform better or worse than the original Winnow if the assumption of a perfect target function were violated?)

(b) 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. *(20 points)*