10-806 Foundations of Machine Learning and Data Science

Homework # 4 **Due: November 30, 2015**

Groundrules:

- Your work will be graded on correctness, clarity, and conciseness. You should only submit work that you believe to be correct; if you cannot solve a problem completely, you will get significantly more partial credit if you clearly identify the gap(s) in your solution. It is good practice to start any long solution with an informal (but accurate) proof summary that describes the main idea.
- 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. [33 pts] **Boosting and Game Theory.** In class, we discussed how the minimax theorem can be used to argue why boosting should be possible in principle. Specifically, given a data sample S and a finite class of hypotheses H, we created a matrix game with one row for each $h_i \in H$ and one column for each $x_j \in S$. Entry $M_{ij} = 1$ if $h_i(x_j)$ is correct and $M_{ij} = -1$ if $h_i(x_j)$ is incorrect (these are payoffs to the row player). We then used the minimax theorem to argue that if for any distribution D over columns there exists a row i with expected payoff $\mathbf{E}_{j\sim D}[M_{ij}] \geq \gamma$, then this means there must exist a distribution P over rows such that for any column j the expected payoff $\mathbf{E}_{i\sim P}[M_{ij}] \geq \gamma$.

   (a) Even though the minimax theorem only applies to finite games, argue that this applies to infinite classes H as well. Specifically, suppose H is an infinite class, and that for any distribution D over S there exists $h \in H$ such that $\Pr_{j\sim D}[h(x_j) \neq f(x_j)] \leq 1/2 - \gamma/2$. We want to argue that this implies there must exist a distribution P over H such that for any $x_j \in S$, $Pr_{h\sim P}[h(x_j) \neq f(x_j)] \leq 1/2 - \gamma/2$. Show how we can argue this by reducing (or converting) it to the finite case.

   (b) Use Hoeffding bounds to argue that not only is there *some* weighted vote over rows that is correct on every $x_j \in S$, but in fact there is a *sparse* combination, in which only a multiset of $O(\frac{1}{\gamma^2}\log|S|)$ hypotheses are combined via majority vote.

2. [33 pts] **"Pruning" a Decision Tree Online via Sleeping Experts.** Consider a decision tree T with L leaves. A *pruning* of T is a new tree h in which one or more of the internal nodes of T have been turned into leaves, labeled as "+" or "−", and with all of their original descendants removed (since they are now leaves). For instance, the very simple tree h in which the root is a leaf labeled "+" would be a pruning of any tree T with $L \geq 2$ leaves, as would be the very simple tree h in which the root is a leaf labeled "−".

   Suppose we are handed a decision tree T and we believe that either it, or some pruning of it, will be a good predictor on future data arriving online. One interesting way we can make predictions is as follows. For each internal node v of T create two sleeping experts: one that predicts positive on any example that reaches v and one that predicts negative on any

example that reaches v. Also, for each leaf v of T, create a sleeping expert that predicts the same as v on any example that reaches v. (In all cases, the expert does not raise its hand if the example does not reach v). So, the total number of sleeping experts is $O(L)$.

(a) Say why any pruning h of T, and any assignment of $\{+,-\}$ labels to the leaves of h, corresponds to a subset of sleeping experts with the property that exactly one sleeping expert in the subset makes a prediction on any given example.

(b) Prove that for any sequence S of examples, if we run the sleeping-experts algorithm using $\epsilon = \sqrt{\frac{\ell \log L}{|S|}}$, then the expected error rate of the algorithm on S (the total number of mistakes of the algorithm divided by $|S|$) will be at most $err_S(h_\ell) + O(\sqrt{\frac{\ell \log L}{|S|}})$, where h_ℓ is the pruning of T with ℓ leaves having the lowest error on S.

(c) (10 pts Extra Credit.) The above algorithm requires knowing a good value of ℓ in advance (to use in setting ϵ). Show how one can achieve a bound of the form:

$$\min_\ell \left[err_S(h_\ell) + O(\sqrt{\frac{\ell \log L}{|S|}}) \right],$$

perhaps by applying another round of combining expert advice.

3. [34 pts] **Estimating frequency counts.** Recall that in class we used the Count-Min sketch to produce estimates of frequency counts such that for each element a_i, our estimate $\hat{F}_i$ satisfies $\hat{F}_i \geq F_i$, and with probability $\geq 1 - \delta$, also satisfies $\hat{F}_i \leq F_i + \epsilon m$, where F_i is the true number of occurences of a_i in the stream.

Here is a different approach that uses a bit more space. What we will do is choose $k = O(\frac{1}{\epsilon} \log \frac{1}{\epsilon})$ random locations in the stream, i.e., k independent uniform random numbers $t_1, \ldots, t_k \in \{1, \ldots, m\}$. For each one, we record what we see there in the stream, and then count all subsequent copies of that element. E.g., if $t_1 = 37$ and we see a 3 there, we will count all 3's that we see from location 37 onward. This will be our estimate of the number of 3's in the stream. If multiple locations t_j have 3's in them, then our estimate $\hat{F}_3$ will just be the largest of the counts, i.e., the count from the smallest such t_j. If some element a_i is not seen in any of the locations t_j then our estimate $\hat{F}_i$ is zero.

(a) It is clear that our estimates $\hat{F}_i$ satisfy $\hat{F}_i \leq F_i$. Argue that with probability $\geq 1 - \delta$, for all i we will have $\hat{F}_s \geq F_s - \epsilon m$, so long as $k \geq \frac{1}{\epsilon} \log \frac{1}{\epsilon\delta}$.

(b) The above algorithm requires the ability to pick a random location in the stream. If we know m ahead of time, this is easy. What if we don't know m ahead of time? How can we implement the algorithm in that case? (This part of the question should be fairly easy given what was discussed in class).

For this problem you can think of $k = 1$ (since you will just be replicating this for each counter).