Sublinear time algorithms	December 28, 2009

Homework 3

Lecturer: Ronitt Rubinfeld — *Due Date: January 11, 2010*

Solve but don't turn in:

1. Influence of monotone functions:

   Definitions for this problem only: For $x = (x_1, \ldots, x_n) \in \{+1, -1\}^n$, $x^{\oplus i}$ is x with the i-th bit flipped, that is,
   $$x^{\oplus i} = (x_1, \ldots, x_{i-1}, -x_i, x_{i+1}, \ldots, x_n).$$
   The *influence of the i-th variable on* $f : \{+1, -1\}^n \to \{+1, -1\}$ is
   $$\mathrm{Inf}_i(f) = \Pr_x \left[f(x) \neq f\left(x^{\oplus i}\right) \right].$$
   The *total influence* of f is
   $$\mathrm{Inf}(f) = \sum_{i=1}^{n} \mathrm{Inf}_i(f).$$
   A function $f : \{+1, -1\}^n \to \{+1, -1\}$ is *monotone* if for all $x, y \in \{+1, -1\}^n$ such that $x_i \leq y_i$ for each i, $f(x) \leq f(y)$. Assume that $-1 < +1$.

   - Using the definition of influence given above, show that for any monotone function $f : \{+1, -1\}^n \to \{+1, -1\}$, the influence of the i^{th} variable is equal to the value of the Fourier coefficient of $\{i\}$, that is $\mathrm{inf}_i(f) = \hat{f}(\{i\})$.
   - Show that the majority function $f(x) = \mathrm{sign}(\sum_i x_i)$ maximizes the total influence among n-variable monotone functions mapping $\{+1, -1\}^n$ to $\{+1, -1\}$, for n odd.

Turn in your solution to *each* of the following problems on a separate sheet of paper, with your name on each one.

1. For the definition of influence given in class, show that
   $$Inf_f(S) = \sum_{T : S \bigcap T \neq \emptyset} \hat{f}(T)^2$$

2. (Cancelled, don't turn in) Let $\mathcal{C}$ be an arbitrary subset of k-junta functions which is symmetric – that is, for any permutation π on $[n]$, if $f \in \mathcal{C}$ then $f(\pi(x)) \in \mathcal{C}$. For example, $\mathcal{C}$ could be the set of dictator functions. Building on the junta test shown in class, show that we can construct a set of m labeled examples $\{(x^1, y^1), \ldots, (x^m, y^m)\}$ where each x^i is independent and uniformly distributed over $\{+1, -1\}^k$ (corresponding to the settings of the k relevant variables) and if $f \in \mathcal{C}$ is a k-junta corresponding to the function f' on the k relevant variables, then y^i corresponds to the value of $f'(x^i)$. The algorithm for constructing these examples should make at most poly(k) queries to f. Though dependent on each other, the queries to f should be uniformly distributed (it will be more difficult, and maybe not even possible, to prove the correctness of your algorithm if the queries are not from the uniform distribution).