Problem Set 4

Sublinear Algorithms

Due Tuesday, November 11

1. Show that any algorithm that computes an $\ell_2/\ell_2$ approximate sparse Fourier transform must look at $\Omega(k \log(n/k) / \log \log n)$ positions of the input, even if the algorithm uses adaptivity.

2. In class we showed how to do $O(1)$-approximate 1-sparse recovery with $O(\log \log n)$ adaptive linear measurements. Show how to do $1 + \epsilon$-approximate 1-sparse recovery with $O(\frac{1}{\epsilon^2} + \log \log n)$ adaptive linear measurements.

3. In class we described an algorithm for computing semi-equispaced Fourier transforms. In particular, we described how if $x$ is $k$-sparse with support $\{1, 2, \ldots, k\}$ then you can compute $\hat{x}_\Omega$ for any set $\Omega$ of size $k$ in $O(k \log^c n)$ time.

For this problem, show how to solve the reverse problem: suppose that you are given $\hat{x}_\Omega$ for an arbitrary set $\Omega$ of size $k$, and know that $x$ is $k$-sparse with support $\{1, 2, \ldots, k\}$. Show how to reconstruct $x$.

4. In this problem we will consider the sparse Hadamard transform. The Hadamard transform on $N = 2^n$ is given by $\hat{x} = Hx$ for $H_{i,j} = (-1)^{(i,j)}$

where $i, j \in \{0, 1\}^n$ are identified with $[N]$. The fast Hadamard transform gives an $O(N \log N)$ time algorithm for converting $x$ to $\hat{x}$. We will show how to recover a $K$-sparse $\hat{x}$ from query access to $x$ in $O(K \log^c N)$ time.

(a) Suppose that $\hat{x}$ is approximately 1-sparse, i.e. there exists an $i$ such that $|\hat{x}_i| > 0.99 \|\hat{x}\|_2$. Use a linear code to find $\hat{x}$ with $O(n)$ samples from $x$ and $O(n^c)$ time.
(b) Now let’s look at extending this to $K$-sparse recovery. Suppose $K = 2^k$, and consider the $K$-dimensional hadamard transform of the vector $y \in \mathbb{R}^K$ given that contains $x_i$ for all $i$ with the last $n - k$ bits equaling some fixed value $r$:

$$y_i = x_i || r$$ for $r \in \{0, 1\}^{n-k}$

Express $\hat{y}_i$ in terms of $\hat{x}$ and $r$.

(c) Now consider any $A \in \{0, 1\}^{n \times k}$ and $r \in \{0, 1\}^n$ in the orthogonal subspace to $A$ (i.e., $A^T r = 0 \mod 2$), and

$$y_i = x_{Ai+r}$$

Express $\hat{y}_i$ in terms of $\hat{x}$, $A$ and $r$.

(d) Show how to use this to “hash” the elements of $\hat{x}$ into $K$ buckets and perform sparse recovery in each bucket. Give an algorithm that, for any $\hat{x} \in \mathbb{R}^N$, recovers most of the coordinates $i$ where $\hat{x}_i^2 > \|\hat{x} - \hat{x}_K\|^2_2/K$, with large constant probability, in $O(K \log^2 N)$ time.

(e) Conclude with an algorithm to perform $\ell_2/\ell_2$ recovery in $O(K \log^2 N)$ time.

5. This problem looks at the 1-sparse Fourier transform. Consider a vector $x \in \mathbb{R}^n$ such that there exists an $i^*$ with

$$|x_{i^*}| > (1 - \epsilon)\|x\|_2.$$ for a sufficiently small constant $\epsilon$. Our goal is to find $i^*$ from samples of the Fourier transform

$$\hat{x}_j = \sum_{i=0}^{n-1} x_i \omega^{ij}$$

for $\omega$ being a primitive $n$th root of unity.

(a) Consider observations of the form

$$f_r(a) = \hat{x}_{r+a}/\hat{x}_r.$$ Show that $f_r(a) \approx \omega^{ai^*}$, in the sense that

$$E_{r \in [n]} |f_r(a) - \omega^{ai^*}|^2 \leq 1/100.$$
(b) Show how, using $O(\log n)$ samples of $f_r(a)$ for random $r, a \in [n]$, you can find $i^*$ in $O(n \log n)$ time with $1/n^c$ failure probability. This would be sample-efficient but not time efficient.

(c) Now suppose you had a sampling method $g(a)$ such that

$$|g(a) - \omega^{a_{i^*}}|^2 \leq 1/100.$$

always. Show how to use $O(\log n)$ samples of $g$ to identify $i^*$ in $O(\log n)$ time.

(d) Based on the previous part, give a method that uses $O(\log n \log \log n)$ time and samples of $f_r(a)$ to recover $i^*$ with $1 - 1/\log^c n$ probability. This is time efficient but not sample efficient.

(e) Combine the above methods – one slow but with exponential failure probability, and the other fast but needing low failure probability in each step – to use $O(\log n)$ samples of $f_r(a)$ and $O(\log^2 n)$ time to recover $i^*$ with constant probability.

Ideally the algorithm should be nonadaptive, but you may use adaptivity if you wish.

Hint: recover $i^*$ $O(\log \log n)$ bits at a time.