MTH 995-003: Intro to CS and Big Data	Spring 2014
Lecture 28 — April 10, 2014	
Inst. Mark Iwen	*Scribe: Chinh Dang*

1 Review

From lectures 26 and 27 we know that $\exists A \in \{0,1\}^{O\left(k^2 \log_k^2 N/\varepsilon^2\right)\times N}$ that is $\left(K := \frac{4k\lceil \log_k N\rceil}{\varepsilon}, \lfloor \log_k N \rfloor\right)$-coherent, and has $\left(A\left(K,n\right)\vec{x}\right)_j \in B\left(x_n, \frac{\varepsilon\left\|\vec{x}-\vec{x}_{S_0(k/\epsilon)}\right\|_1}{k}\right)$ for more than half of $j \in [K]$ $\forall n \in \mathbb{N}$, and $\vec{x} \in \mathbb{C}^N$.

For the remainder of this lecture,

- A is the matrix above. It is entirely deterministic, and easy to store (or encode in hardware).
- $A\left(K,n\right)$ is a submatrix of A for all n. So, if we know $A\vec{x}$, we also know $\left(A\left(K,n\right)\overrightarrow{x}\right)$ $\forall n \in \mathbb{N}$
- The rows in $A\left(K,n\right)$ can be found quickly "on the fly" in $O(K \cdot \log_k N)$-time

2 Main Lecture

We will consider the following reconstruction algorithm for approximating $\vec{x}$ given $A\vec{x}$, and a subset $S \subseteq [N]$.

Algorithm 1.

Input: $A\vec{x}$, and $S \subseteq [N]$

1. For each $n \in S$
2. Let $\mathbf{Re}\left[z_n\right]$ be the median of the entries $\mathbf{Re}\left\{A\left(K,n\right)\vec{x}\right\}$
   Let $\mathbf{Im}\left[z_n\right]$ be the median of the entries $\mathbf{Im}\left\{A\left(K,n\right)\vec{x}\right\}$
3. End for
4. Sort $\vec{z}_S$ by the magnitude of its entries $|z_{n_1}| \geq |z_{n_2}| \geq \dots$
5. **Output** $\vec{z}_{\widetilde{S}}$ for $\widetilde{S} = \{n_1, \dots, n_{2k}\}$

Running time of Algorithm 1:

Lines 1- 3: $O\left(|S|\, K \log N\right)$

Line 4: $O\left(|S| \log |S|\right)$

Hence, the total runtime becomes $O\left(|S|\, \frac{K\log^2 N}{\varepsilon^2}\right)$. In case $S = [N]$, it will run in $O\left(N\frac{K\log^2 N}{\varepsilon^2}\right)$-time.

Theorem 1. *Let $\vec{x} \in \mathbb{C}^N$ and $A \in \{0,1\}^{m\times N}$ be as above. Let $S_0(k) \subset [N]$ be the $\lfloor k \rfloor$-largest magnitude entries of $\vec{x} \in \mathbb{C}^N$ for any $k \in (1, N)$. Suppose that the S passed into Algorithm 1 has $S_0(k) \subset S,\ |S| \geq 2k$, then the output of Algorithm 1, $z_{\widetilde{S}} \in \mathbb{C}^N$ satisfies:*

$$\left\|\vec{x} - \vec{z}_{\widetilde{S}}\right\|_2 \leq \left\|\vec{x} - \vec{x}_{S_0(k)}\right\|_2 + \frac{22\varepsilon \left\|\vec{x} - \vec{x}_{S_0(k/\varepsilon)}\right\|_1}{\sqrt{k}} \qquad (*)$$

Proof: Let $\delta = \frac{\varepsilon \left\|x - x_{S_0(k/\varepsilon)}\right\|_1}{k}$. Theorem 1 from lecture 27 implies $|\vec{z}_n - \vec{x}_n| \leq \sqrt{2}\delta\ \forall n \in S$
Thus:

$$\left\|\vec{x} - \vec{z}_{\widetilde{S}}\right\|_2 \leq \left\|\vec{x} - \vec{x}_{\widetilde{S}}\right\|_2 + \left\|\vec{x}_{\widetilde{S}} - \vec{z}_{\widetilde{S}}\right\|_2 \leq \left\|\vec{x} - \vec{x}_{\widetilde{S}}\right\|_2 + 2\sqrt{k}\delta$$

Since

$$\left\|\vec{x} - \vec{x}_{\widetilde{S}}\right\|_2 = \sqrt{\left\|\vec{x} - \vec{x}_{S_0(k)}\right\|_2^2 + \sum_{n\in S_0(k)\setminus\widetilde{S}} |\vec{x}_n|^2 - \sum_{n\in\widetilde{S}\setminus S_0(k)} |\vec{x}_n|^2}$$

we need $\sum_{n\in S_0(k)\setminus\widetilde{S}} |x_n|^2 - \sum_{n\in\widetilde{S}\setminus S_0(k)} |x_n|^2 \leq \left(20\sqrt{k}\delta\right)^2$ in order to get $(*)$. Let

$$\nu := \sum_{n\in S_0(k)\setminus\widetilde{S}} |x_n|^2 - \sum_{n\in\widetilde{S}\setminus S_0(k)} |x_n|^2.$$

There are 3 cases to consider:

- Case 1. $S_0(k)\setminus\widetilde{S} = \emptyset$. It implies that $\nu \leq 0 < \left(20\sqrt{k}\delta\right)^2$
- Cases 2 and 3. Suppose $j_l \in S_0(k)\setminus\widetilde{S}$. Since $S_0(k) \subset S$, this only happens if line 4 of Algorithm 1 found $|z_n| \geq |z_{j_l}|$ for all $n \in \widetilde{S}\setminus S_0(k)$.
  However, $|z_n| \geq |z_{jl}|$ implies that

  $$|x_{j_k}| + \sqrt{2}\delta \geq |x_n| + \sqrt{2}\delta \geq |z_n| \geq |z_{j_l}| \geq |x_{j_l}| - \sqrt{2}\delta \geq |x_{j_k}| - \sqrt{2}\delta$$

  Thus,

  $$\sum_{n\in\widetilde{S}\setminus S_0(k)} |x_n|^2 \geq \left|\widetilde{S}\setminus S_0(k)\right| \left(|x_{j_k}| - 2\sqrt{2}\delta\right)^2$$

  $$\geq A := 2\left|S_0(k)\setminus\widetilde{S}\right| \left(|x_{j_k}| - 2\sqrt{2}\delta\right)^2.$$

  On the other hand, $B := \left|S_0(k)\setminus\widetilde{S}\right| \left(|x_{j_k}| + 2\sqrt{2\delta}\right)^2 \geq \sum_{n\in S_0(k)\setminus\widetilde{S}} |x_n|^2$.
  Case 2: if $A \geq B$, we got the result.
  Case 3: if $B > A$, we have:

  $$|x_{jk}|^2 - 12\sqrt{2}\delta\,|x_{jk}| + 8\delta^2 < 0$$
  $$\Rightarrow |x_{jk}| \leq \left(8 + 6\sqrt{2}\right)\delta \leq (20\delta)^2.$$

  Thus, we have our result. $\square$

Note: If we can quickly find an S with $S_0(k) \subset S$, then theorem 1 implies that Algorithm 1 will quickly produce a good approximation to $\vec{x}$. In the next lecture, we will focus on how to find such an $S \subset [N]$ quickly. This will be done with "bit testing matrices".

Definition 1. *The N^{th} bit testing matrix is $B_N \in \{0,1\}^{(1+\lceil \log_2 N \rceil)\times N}$ where:*

$$(B_N)_{ij} = \begin{cases} 1 & \text{if } i = 0 \\ (i-1)^{\text{th}} \text{ bit in the binary expansion of } j & \text{if } i > 0 \end{cases}.$$

Example 1. *$B_4 \in \{0,1\}^{3\times 4}$, we have*

$$B_4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

We will also need "row tensor products" of matrices.

Definition 2. *Let $A \in \mathbb{R}^{m\times N}, C \in \mathbb{R}^{\widetilde{m}\times N}$. Their row tensor product is $A \otimes C \in \mathbb{R}^{(m\cdot\widetilde{m})\times N}$ with*

$$(A \otimes C)_{ij} = A_{i \bmod m, j}\; C_{\frac{i-(i \bmod m)}{m}, j}.$$

Example 2. *Let $A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ -1 & 1 & 1 & 1 \end{pmatrix}$, and let the matrix B_4 be as above. Then, their row tensor product is*

$$A \otimes B_4 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ -1 & 1 & 1 & 1 \\ 0 & 2 & 0 & 4 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 3 & 4 \\ 0 & 0 & 1 & 1 \end{pmatrix}.$$

References

[1] Iwen, M. A. Compressed sensing with sparse binary matrices: Instance optimal error guarantees in near-optimal time. *Journal of Complexity* (2013).

[2] Cormode, Graham, and S. Muthukrishnan. Combinatorial algorithms for compressed sensing. In *Structural Information and Communication Complexity,* pp. 280-294. Springer Berlin Heidelberg, 2006.

[3] Gilbert, Anna C., Martin J. Strauss, Joel A. Tropp, and Roman Vershynin. One sketch for all: fast algorithms for compressed sensing. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing,* pp. 237-246. ACM, 2007.

[4] Gilbert, Anna C., Yi Li, Ely Porat, and Martin J. Strauss. Approximate sparse recovery: optimizing time and measurements. *SIAM Journal on Computing 41,* no. 2 (2012): 436-453.

[5] Bailey, J., Mark A. Iwen, and Craig V. Spencer. On the design of deterministic matrices for fast recovery of fourier compressible functions. *SIAM Journal on Matrix Analysis and Applications* 33, no. 1 (2012): 263-289.