6.893: Algorithms and Signal Processing

Lecture 14:
Sparse Recovery via Sparse Matrices
Compressed Sensing: Recap

• Want to acquire a signal \( x = [x_1 \ldots x_n] \)
• Acquisition proceeds by computing \( Ax \) (+noise) of dimension \( m \ll n \)
• From \( Ax \) we want to recover an approximation \( x^* \) of \( x \)
• Method: solve the following program (Lecture 12)

\[
\begin{align*}
\text{minimize} & \quad ||x^*||_1 \\
\text{subject to} & \quad Ax^* = Ax
\end{align*}
\]

or use a greedy algorithm (Lecture 13)

• Guarantee: for some \( C > 1 \)

\[
||x-x^*||_1 \leq C \min_{k\text{-sparse } x''} ||x-x''||_1
\]

as long as \( A \) satisfies \((ck, \delta)\)-RIP: for all \( ck\)-sparse \( x \)

\[
(1-\delta) ||x||_2 \leq ||Ax||_2 \leq (1+\delta) ||x||_2
\]
Constructing matrix $A$

- “Most” matrices $A$ work
  - Dense matrices (Gaussian, Partial Fourier)
  - Sparse matrices:
    - Data stream algorithms
    - Coding theory (LDPCs)
Application II: Monitoring
Network Traffic Data Streams

- Router routes packets
  - Where do they come from?
  - Where do they go to?
- Ideally, would like to maintain a traffic matrix $x_{[.,.]}$
  - Easy to update: given a $(src,dst)$ packet, increment $x_{src,dst}$
  - Requires way too much space! ($2^{32} \times 2^{32}$ entries)
  - Need to compress $x$, increment easily
- Using linear compression we can:
  - Maintain sketch $Ax$ under increments to $x$, since $A(x+\Delta) = Ax + A\Delta$
  - Recover $x^*$ from $Ax$
Application III: Pooling Experiments

- Pooling Experiments
  [Kainkaryam, Woolf’08], [Hassibi et al’07], [Dai-Sheikh, Milenkovic, Baraniuk], [Shental-Amir-Zuk’09],[Erlich-Shental-Amir-Zuk’09]
## Results

<table>
<thead>
<tr>
<th>Paper</th>
<th>R/D</th>
<th>Sketch length</th>
<th>Encode time</th>
<th>Column sparsity</th>
<th>Recovery time</th>
<th>Approx</th>
</tr>
</thead>
<tbody>
<tr>
<td>[CCF'02], [CM'06]</td>
<td>R</td>
<td>k log n</td>
<td>n log n</td>
<td>log n</td>
<td>n log n</td>
<td>l2 / l2</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>k log^c n</td>
<td>n log^c n</td>
<td>log^c n</td>
<td>k log^c n</td>
<td>l2 / l2</td>
</tr>
<tr>
<td>[CM'04]</td>
<td>R</td>
<td>k log n</td>
<td>n log n</td>
<td>log n</td>
<td>n log n</td>
<td>l1 / l1</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>k log^c n</td>
<td>n log^c n</td>
<td>log^c n</td>
<td>k log^c n</td>
<td>l1 / l1</td>
</tr>
<tr>
<td>[CRT'04] [RV'05]</td>
<td>D</td>
<td>k log(n/k)</td>
<td>nk log(n/k)</td>
<td>k log(n/k)</td>
<td>n^c</td>
<td>l2 / l2</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>k log^c n</td>
<td>n log n</td>
<td>k log^c n</td>
<td>n^c</td>
<td>l2 / l2</td>
</tr>
<tr>
<td>[GSTV'06] [GSTV'07]</td>
<td>D</td>
<td>k log^c n</td>
<td>n log^c n</td>
<td>log^c n</td>
<td>k log^c n</td>
<td>l1 / l1</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>k log^c n</td>
<td>n log^c n</td>
<td>k log^c n</td>
<td>k^2 log^c n</td>
<td>l2 / l2</td>
</tr>
<tr>
<td>[BGIKS'08]</td>
<td>D</td>
<td>k log(n/k)</td>
<td>n log(n/k)</td>
<td>log(n/k)</td>
<td>n^c</td>
<td>l1 / l1</td>
</tr>
<tr>
<td>[GLR'08]</td>
<td>D</td>
<td>k logn^{logloglogn}</td>
<td>kn^{1-a}</td>
<td>n^{1-a}</td>
<td>n^c</td>
<td>l2 / l2</td>
</tr>
<tr>
<td>[NV'07], [DM'08], [NT'08], [BD'08], [GK'09], ...</td>
<td>D</td>
<td>k log(n/k)</td>
<td>nk log(n/k)</td>
<td>k log(n/k)</td>
<td>nk log(n/k) * log</td>
<td>l2 / l1</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>k log^c n</td>
<td>n log n</td>
<td>k log^c n</td>
<td>n log n * log</td>
<td>l2 / l1</td>
</tr>
<tr>
<td>[IR'08], [BIR'08],[BI'09]</td>
<td>D</td>
<td>k log(n/k)</td>
<td>n log(n/k)</td>
<td>log(n/k)</td>
<td>n log(n/k)* log</td>
<td>l1 / l1</td>
</tr>
<tr>
<td>[GLSP'09]</td>
<td>R</td>
<td>k log(n/k)</td>
<td>n log^c n</td>
<td>log^c n</td>
<td>k log^c n</td>
<td>l2 / l2</td>
</tr>
</tbody>
</table>

Caveats: (1) most “dominated” results not shown (2) only results for general vectors x are displayed (3) sometimes the matrix type matters (Fourier, etc)
Algorithms for Sparse Matrices

- **Sketching/streaming**: [Charikar-Chen-Colton’02, Estan-Varghese’03, Cormode-Muthukrishnan’04]
- **LDPC-like**: [Xu-Hassibi’07, Indyk’08, Jafarpour-Xu-Hassibi-Calderbank’08]
- **L1 minimization**: [Berinde-Gilbert-Indyk-Karloff-Strauss’08, Wang-Wainwright-Ramchandran’08]
- **Message passing**: [Sarvotham-Baron-Baraniuk’06,’08, Lu-Montanari-Prabhakar’08, Akcakaya-Tarokh’11]
- **Matching pursuit**: [Indyk-Ruzic’08, Berinde-Indyk-Ruzic’08, Berinde-Indyk’09, Gilbert-Li-Porat-Strauss’10, Cevher’10]
- **Group testing**: …[Gupta-Indyk-Price-Rachlin’11] (for “folding” matrices)
- **Surveys**:
  - A. Gilbert, P. Indyk, Proceedings of IEEE, June 2010
### Part I

<table>
<thead>
<tr>
<th>Paper</th>
<th>R/ D</th>
<th>Sketch length</th>
<th>Encode time</th>
<th>Column sparsity</th>
<th>Recovery time</th>
<th>Approx</th>
</tr>
</thead>
<tbody>
<tr>
<td>[CM’04]</td>
<td>R</td>
<td>k log n</td>
<td>n log n</td>
<td>log n</td>
<td>n log n</td>
<td>l₁ / l₁</td>
</tr>
</tbody>
</table>

Theorem: There is a distribution over mxn matrices $A$, $m=O(k \log n)$, such that for any $x$, given $Ax$, we can recover $x^*$ such that

$$||x-x^*||_1 \leq C \text{Err}_1,$$

where $\text{Err}_1 = \min_{k\text{-sparse } x'} ||x-x'||_1$

with probability $1-1/n$.

The recovery algorithm runs in $O(n \log n)$ time.

This talk:

- Assume $x \geq 0$ – this simplifies the algorithm and analysis; see the original paper for the general case
- Prove the following $l_\infty/l_1$ guarantee: $||x-x^*||_\infty \leq C \text{Err}_1 /k$

This is actually stronger than the $l_1/l_1$ guarantee

Note: [CM’04] originally proved a weaker statement where $||x-x^*||_\infty \leq C||x||_1 /k$. The stronger guarantee follows from the analysis of [CCF’02] (cf. [GGIKMS’02]) who applied it to $\text{Err}_2$.
Basic block

- **Matrix view:**
  - A 0-1 $w \times n$ matrix $A$, with one 1 per column
  - The $i$-th column has 1 at position $h(i)$, where $h(i)$ be chosen uniformly at random from $\{1…w\}$

- **Hashing view:**
  - $c = Ax$
  - $h$ hashes coordinates into “buckets” $c_1…c_w$

- **Estimator:** $x_i^* = c_{h(i)}$

Closely related: [Estan-Varghese’02], “counting”/”spectral” Bloom filters
What are the guarantees?

• Definitions:
  – Let \( k = \frac{w}{C} \)
  – Let \( S \) be the set of \( k \) heaviest coefficients of \( x \), i.e., the “head”
  – Let \( \text{Err}_{1}^{k} = \|x_{-S}\|_{1} \)
    (i.e., the sum of coeffs not in the head)

• Will show that, with constant probability
  \[ x_{a} \leq x^{*}_{a} \leq x_{a} + \text{Err}_{1}^{k}/k \]

\[ c_{j} = \sum_{a: h(a) = j} x_{a} \]

\[ x^{*}_{a} = c_{h(a)} \]
Analysis

- We show how to get an estimate

\[ x^*_a \leq x_a + \frac{\text{Err}}{k} \]

- \( \Pr[ |x^*_a - x_a| > \frac{\text{Err}}{k}] \leq P_1 + P_2 \), where

\[ P_1 = \Pr[ \text{a collides with (another) head element} ] \]
\[ P_2 = \Pr[ \text{sum of tail elems colliding with a is } > \frac{\text{Err}}{k} ] \]

- We have

\[ P_1 \leq \frac{k}{w} = \frac{1}{C} \]
\[ P_2 \leq \frac{\text{Err}/w}{\text{Err}/k} = \frac{k}{w} = \frac{1}{C} \]

- Total probability of failure \( \leq \frac{2}{C} \)
Amplification

• Algorithm:
  – Maintain \( d \) functions \( h_1 \ldots h_d \) and vectors \( c^1 \ldots c^d \)
  – Estimator:
    \[
    x_i^* = \min_t c^t_{h_t(i)}
    \]

• Analysis:
  – \( \Pr[|x_i^*-x_i| \geq \frac{\text{Err}}{k}] \leq \left(\frac{2}{C}\right)^d \)
  – Setting \( d=O(\log n) \) (and thus \( m=O(k \log n) \)) ensures that w.h.p
    \[
    |x_i^*-x_i| < \frac{\text{Err}}{k}
    \]
<table>
<thead>
<tr>
<th>Paper</th>
<th>R/D</th>
<th>Sketch length</th>
<th>Encode time</th>
<th>Column sparsity</th>
<th>Recovery time</th>
<th>Approx</th>
</tr>
</thead>
<tbody>
<tr>
<td>[BGIKS’08]</td>
<td>D</td>
<td>$k \log(n/k)$</td>
<td>$n \log(n/k)$</td>
<td>$\log(n/k)$</td>
<td>$n^c$</td>
<td>$l_1 / l_1$</td>
</tr>
<tr>
<td>[IR’08], [BIR’08],[BI’09]</td>
<td>D</td>
<td>$k \log(n/k)$</td>
<td>$n \log(n/k)$</td>
<td>$\log(n/k)$</td>
<td>$n \log(n/k) \ast \log$</td>
<td>$l_1 / l_1$</td>
</tr>
</tbody>
</table>
- **Restricted Isometry Property (RIP)** [Candes-Tao’04]
  \[ \Delta \text{ is } k\text{-sparse} \Rightarrow ||\Delta||_2 \leq ||A\Delta||_2 \leq C ||\Delta||_2 \]

- Holds w.h.p. for:
  - Random Gaussian/Bernoulli: \( m = O(k \log (n/k)) \)
  - Random Fourier: \( m = O(k \log^O(1) n) \)

- Consider \( m \times n \) 0-1 matrices with \( d \) ones per column

- Do they satisfy RIP?
  - No, unless \( m = \Omega(k^2) \) [Chandar’07]

- However, they can satisfy the following **RIP-1** property [Berinde-Gilbert-Indyk-Karloff-Strauss’08]:
  \[ \Delta \text{ is } k\text{-sparse} \Rightarrow d (1-\varepsilon) ||\Delta||_1 \leq ||A\Delta||_1 \leq d||\Delta||_1 \]

- Sufficient (and necessary) condition: the underlying graph is a \( (k, d(1-\varepsilon/2)) \)-expander
Expanders

- A bipartite graph is a \((k,d(1-\varepsilon))\)-expander if for any left set \(S\), \(|S| \leq k\), we have \(|N(S)| \geq (1-\varepsilon)d |S|\)
- Objects well-studied in theoretical computer science and coding theory
- Constructions:
  - Probabilistic: \(m = O(k \log (n/k))\)
  - Explicit: \(m = k \text{ quasipolylog } n\)
- High expansion implies RIP-1:
  \[ \Delta \text{ is } k\text{-sparse } \Rightarrow d \ (1-\varepsilon) \ ||\Delta||_1 \leq ||A\Delta||_1 \leq d||\Delta||_1 \]
  [Berinde-Gilbert-Indyk-Karloff-Strauss’08]
Proof: $d(1-\varepsilon/2)$-expansion $\Rightarrow$ RIP-1

- Want to show that for any k-sparse $\Delta$ we have
  \[ d(1-\varepsilon) \|\Delta\|_1 \leq \|A\Delta\|_1 \leq d\|\Delta\|_1 \]
- RHS inequality holds for any $\Delta$
- LHS inequality:
  - W.l.o.g. assume $|\Delta_1| \geq \cdots \geq |\Delta_k| \geq |\Delta_{k+1}| = \cdots = |\Delta_n| = 0$
  - Consider the edges $e=(i,j)$ in a lexicographic order
  - For each edge $e=(i,j)$ define $r(e)$ s.t.
    - $r(e)=-1$ if there exists an edge $(i',j)<(i,j)$
    - $r(e)=1$ if there is no such edge
- Claim 1: $\|A\Delta\|_1 \geq \sum_{e=(i,j)} |\Delta_i|r_e$
- Claim 2: $\sum_{e=(i,j)} |\Delta_i|r_e \geq (1-\varepsilon) d\|\Delta\|_1$
Algorithms
L1 minimization

- **Algorithm:**
  
  minimize $\|x^*\|_1$
  subject to $Ax = Ax^*$

- **Theoretical performance:**
  - RIP1 replaces RIP2 *
  - L1/L1 guarantee

- **Experimental performance**
  - Thick line: Gaussian matrices
  - Thin lines: prob. levels for sparse matrices ($d=10$)
  - Same performance!
Sequential Sparse Matching Pursuit [Berinde-Indyk’09]

- **Algorithm:**
  - \( x^* = 0 \)
  - Repeat \( T \) times
    - Repeat \( S = O(k) \) times
      - Find \( i \) and \( z \) that minimize \( \| A(x^* + z e_i) - A x \|_1 \)
      - \( x^* = x^* + z e_i \)
    - Sparsify \( x^* \)
      - (set all but \( k \) largest entries of \( x^* \) to 0)
  - Similar to SMP, but updates done sequentially

* Set \( z = \text{median}[ (A x^* - A x)_{N(i)} ] \). Instead, one could first optimize (gradient) \( i \) and then \( z \) [ Fuchs’09]
\( l_\infty / l_1 \) implies \( l_1 / l_1 \)

**Algorithm:**
- Assume we have \( x^* \) s.t. \( \| x - x^* \|_\infty \leq C \text{Err}_1 / k \).
- Let vector \( x' \) consist of \( k \) largest (in magnitude) elements of \( x^* \).

**Analysis**
- Let \( S \) (or \( S^* \)) be the set of \( k \) largest in magnitude coordinates of \( x \) (or \( x^* \)).
- Note that \( \| x^*_{S} \| \leq \| x^*_{S^*} \|_1 \)
- We have
  \[
  \| x - x' \|_1 \leq \| x \|_1 - \| x_{S^*} \|_1 + \| x_{S^*} - x^*_{S^*} \|_1 \\
  \leq \| x \|_1 - \| x^*_S \|_1 + 2\| x_{S^*} - x^*_S \|_1 \\
  \leq \| x \|_1 - \| x_S \|_1 + 2\| x_{S^*} - x^*_S \|_1 \\
  \leq \| x \|_1 - \| x_S \|_1 + \| x^*_S - x_S \|_1 + 2\| x_{S^*} - x^*_S \|_1 \\
  \leq \text{Err} + 3\alpha / k * k \\
  \leq (1 + 3\alpha) \text{Err}
  \]