Tutorial: Sparse Recovery Using Sparse Matrices

Piotr Indyk
MIT
Problem Formulation

(approximation theory, learning Fourier coeffs, linear sketching, finite rate of innovation, **compressed sensing**...)

- **Setup:**
  - Data/signal in \( n \)-dimensional space: \( x \)
    - E.g., \( x \) is an 256x256 image \( \Rightarrow n=65536 \)
  - Goal: compress \( x \) into a “sketch” \( Ax \),
    where \( A \) is a \( m \times n \) “sketch matrix”, \( m \ll n \)

- **Requirements:**
  - Plan A: want to recover \( x \) from \( Ax \)
    - Impossible: underdetermined system of equations
  - Plan B: want to recover an “approximation” \( x^* \) of \( x \)
    - Sparsity parameter \( k \)
    - Informally: want to recover largest \( k \) coordinates of \( x \)
    - Formally: want \( x^* \) such that
      \[
      ||x^*-x||_p \leq C(k) \min_{x'} ||x'-x||_q
      \]
      over all \( x' \) that are \( k \)-sparse (at most \( k \) non-zero entries)

- **Want:**
  - Good compression (small \( m=m(k,n) \))
  - Efficient algorithms for encoding and recovery

- **Why linear compression?**
  - Broader functionality!
  - Useful for compressed signal acquisition, streaming algorithms, etc
    (see Appendix for more info)
Constructing matrix $A$

- “Most” matrices $A$ work
  - Sparse matrices:
    - Data stream algorithms
    - Coding theory (LDPCs)
  - Dense matrices:
    - Compressed sensing
    - Complexity/learning theory (Fourier matrices)

- “Traditional” tradeoffs:
  - Sparse: computationally more efficient, explicit
  - Dense: shorter sketches

- Recent results: the “best of both worlds”
## Prior and New Results

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

Scale: Excellent Very Good Good Fair
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: $||x-x^*||_\infty \leq C \frac{\text{Err}_1}{k}$

This is actually stronger than the $l_1/l_1$ guarantee (cf. [CM’06], see also the Appendix)

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 $\text{Err}_2$
First attempt

- Matrix view:
  - A 0-1 $wxn$ 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:
  - $Z=Ax$
  - $h$ hashes coordinates into “buckets” $Z_1…Z_w$

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

Closely related: [Estan-Varghese’03], “counting” Bloom filters
Analysis

- We show
  \[ x_i^* \leq x_i \pm \alpha \frac{\text{Err}}{k} \]
  with probability \( >\frac{1}{2} \)
- Assume
  \[ |x_{i_1}| \geq \ldots \geq |x_{i_m}| \]
  and let \( S=\{i_1\ldots i_k\} \) ("elephants")
- When is \( x_i^* > x_i \pm \alpha \frac{\text{Err}}{k} \) ?
  - **Event 1**: \( S \) and \( i \) collide, i.e., \( h(i) \) in \( h(S-\{i\}) \)
    Probability: at most \( \frac{k}{(4/\alpha)k} = \frac{\alpha}{4} < \frac{1}{4} \) (if \( \alpha < 1 \))
  - **Event 2**: many “mice” collide with \( i \), i.e.,
    \[ \sum_{t \text{ not in } S \cup \{i\}, h(t)=h(i)} x_t > \alpha \frac{\text{Err}}{k} \]
    Probability: at most \( \frac{1}{4} \) (Markov inequality)
- Total probability of “bad” events <\( \frac{1}{2} \)
Second try

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

• Analysis:
  – $\Pr[|x_i^*-x_i| \geq \alpha \text{ Err}/k] \leq 1/2^d$
  – Setting $d=O(\log n)$ (and thus $m=O(k \log n)$ )
    ensures that w.h.p
    \[
    |x_i^*-x_i| < \alpha \text{ Err}/k
    \]
## Part II

<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>l1 / 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>
</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
Expander

- 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
  \[ d \left(1-\varepsilon\right) \|\Delta\|_1 \leq \|A\Delta\|_1 \leq d\|\Delta\|_1 \]
- RHS inequality holds for any $\Delta$
- LHS inequality:
  - W.l.o.g. assume
    \[ |\Delta_1| \geq \ldots \geq |\Delta_k| \geq |\Delta_{k+1}| = \ldots = |\Delta_n| = 0 \]
  - 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: $\sum_{e=(i,j)} |\Delta_i| r_e \geq (1-\varepsilon) d\|\Delta\|_1$
Recovery: algorithms
Matching Pursuit(s)

- Iterative algorithm: given current approximation $x^*$:
  - Find (possibly several) $i$ s. t. $A_i$ “correlates” with $Ax-Ax^*$. This yields $i$ and $z$ s. t.
    $||x^*+ze_i-x||_p << ||x^*-x||_p$
  - Update $x^*$
  - Sparsify $x^*$ (keep only $k$ largest entries)
  - Repeat

- Norms:
  - $p=2$ : CoSaMP, SP, IHT etc (RIP)
  - $p=1$ : SMP, SSMP (RIP-1)
  - $p=0$ : LDPC bit flipping (sparse matrices)
Sequential Sparse Matching Pursuit

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

* Set \( z=\text{median}[ (Ax^*-Ax)_{N(i)} ] \). Instead, one could first optimize (gradient) \( i \) and then \( z \) [Fuchs’09]
SSMP: Approximation guarantee

- Want to find k-sparse $x^*$ that minimizes $||x-x^*||_1$
- By RIP1, this is approximately the same as minimizing $||Ax-Ax^*||_1$
- Need to show we can do it *greedily*

Supports of $a_1$ and $a_2$ have small overlap (typically)
Conclusions

• Sparse approximation using sparse matrices
• State of the art: deterministically can do 2 out of 3:
  – Near-linear encoding/decoding
  – $O(k \log (n/k))$ measurements
  – Approximation guarantee with respect to L2/L1 norm

• Open problems:
  – 3 out of 3 ?
  – Explicit constructions ?

• For more, see
Appendix
$l_\infty/l_1$ 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
    
    $$
    ||x-x'||_1 \leq ||x||_1 - ||x^*_S||_1 + ||x^*_S-x^*_S'||_1 \\
    \leq ||x||_1 - ||x^*_S||_1 + 2||x^*_S-x^*_S'||_1 \\
    \leq ||x||_1 - ||x^*_S||_1 + 2||x^*_S-x^*_S'||_1 \\
    \leq ||x||_1 - ||x_S^*||_1 + ||x^*_S-x_S||_1 + 2||x^*_S-x^*_S'||_1 \\
    \leq \text{Err} + 3\alpha/k \times k \\
    \leq (1+3\alpha)\text{Err}
    $$
Application I: 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 $x_{[.,\cdot]}$
  - Easy to update: given a $(\text{src},\text{dst})$ packet, increment $x_{\text{src},\text{dst}}$
  - 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 $Ax$
Applications, ctd.

• Single pixel camera
  [Wakin, Laska, Duarte, Baron, Sarvotham, Takhar, Kelly, Baraniuk’06]

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

SSMP is ran with $S=10000, T=20$. SMP is ran for 100 iterations. Matrix sparsity is $d=8$. 
SSMP: Running time

- **Algorithm:**
  - \( x^* = 0 \)
  - Repeat \( T \) times
    - For each \( i = 1 \ldots n \) compute \( z_i \) that achieves
      \[
      D_i = \min_z \| A(x^* + z e_i) - b \|_1
      \]
      and store \( D_i \) in a heap
    - Repeat \( S = O(k) \) times
      - Pick \( i, z \) that yield the best gain
      - Update \( x^* = x^* + z e_i \)
      - Recompute and store \( D_i \) for all \( i' \) such that \( N(i) \) and \( N(i') \) intersect
    - Sparsify \( x^* \)
      (set all but \( k \) largest entries of \( x^* \) to 0)

- **Running time:**
  \[
  T \left[ n(d + \log n) + k \frac{nd}{m^*d} (d + \log n) \right] = T \left[ n(d + \log n) + nd (d + \log n) \right] = T \left[ nd (d + \log n) \right]
  \]