Sparse recovery via greedy algorithms

Piotr Indyk
MIT
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 $x$
  - Note: $x^*$ does not have to be k-sparse in general
- Method: solve the following program:
  
  \[
  \begin{align*}
  \text{minimize} & \quad ||x^*||_1 \\
  \text{subject to} & \quad Ax^* = Ax
  \end{align*}
  \]

- Guarantee: for some $C>1$
  
  \[\|x-x^*\|_1 \leq C \min_{\text{k-sparse } x''} \|x-x''\|_1\]

  as long as $A$ satisfies $(ck, \delta)$-RIP-$p$, for $p=1$ or $p=2$

  \[\left(1-\delta\right) \|x\|_p \leq \|Ax\|_p \leq \left(1+\delta\right) \|x\|_p\]

- Main drawback: running time
  - Somewhat alleviated by using sparse matrices
    (or Fourier matrices, see next slide)
<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>
</table>

Results
## 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>I2 / I2</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>I2 / I2</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>I1 / I1</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>I1 / I1</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>I2 / I2</td>
</tr>
<tr>
<td>[GSTV’06], [GSTV’07]</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>I2 / I2</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>I1 / I1</td>
</tr>
<tr>
<td>[GLR’08]</td>
<td>D</td>
<td>k log n log^c</td>
<td>kn^1-a</td>
<td>n^1-a</td>
<td>n^c</td>
<td>I2 / I2</td>
</tr>
<tr>
<td>[NV’07], [DM’08], [NT’08], [BD’08], [HK’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>I2 / I1</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>I2 / I2</td>
</tr>
<tr>
<td>[IR’08], [BIR’08], [BI’09], [P’10]</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>I1 / I1</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>I2 / I2</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)
Matching Pursuit(s)

- Iterative algorithm: given current approximation $x^*$, try to find an update

\[
\begin{align*}
A & \quad x^*-x \\
\text{i} & \quad \text{i} \quad \text{Ax-Ax*} \\
\end{align*}
\]

- 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 (based on RIP)
  - $p=1$: SMP, SSMP (based on RIP-1)
  - $p=0$: LDPC bit flipping (based on expander matrices)
Iterated Hard Thresholding

- **Setup:**
  - $x$: $k$-sparse
  - $A$: satisfies RIP of order $3k$ with constant $\delta$
  - $y = Ax + e$

- **Algorithm:**
  - $x^0 = 0$
  - Repeat $t = \Theta(\log \frac{\|x\|_2}{\|e\|_2})$
  - $x^{i+1} = H_k[x^i + A^T(y - Ax^i)]$

where $H_k[x]$ returns the $k$ largest (in magnitude) coefficients of $x$
\[ x^{i+1} = H_k[x^i + A^T(y - Ax^i)]: \text{intuition} \]

- Suppose that all columns of \( A \) were orthogonal, i.e., \( A^T A = I \)
- Then \( x^i + A^T(y - Ax^i) \)
  \[ = x^i + A^T A x + A^t e - A^T A x^i \]
  \[ = x^i + x + A^t e - x^i \]
  \[ = x + A^t e \]
- Thus, modulo \( e \), \( H_k[x + A^t e] \) would return a correct answer
- We will show that RIP implies that \( A^T A \) is “close to” \( I \), at least when applied to “sparse” vectors
Running time

• Compute

\[ x^{i+1} = H_k [x^i + A^T(y - Ax^i)] \]

\[ t = O(\log \|x\|_2 / \|e\|_2) \] times

• Time dominated by computing \( Ax, A^T y \)

• Options:
  – \( A \) is a **Gaussian** matrix: time \( O(nmt) \)
  – \( A \) consists of random rows of Fourier matrix:
    time \( O(n \log n \cdot t) \)
    • But \( m = O(k \log^c n) \) to get RIP [Candes-Tao, Rudelson-Vershynin]
SSMP
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)

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

- 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, i.e., can find $i, z$ s.t.
  $$||A(x^*+ze_i)-Ax||_1 < (1-1/(ck))||Ax^*-Ax||_1$$
- This is a somewhat subtle issue. E.g., consider $A=\begin{bmatrix} a_1 & a_2 \end{bmatrix}$
- Need to show we are always in case (2), not (1)
- Approaches:
  - Analyze the overlaps between the columns (expansion) [Berinde-Indyk'09]
  - Use the fact that
    $$||Ax^*||_1 > (1-\delta)\Sigma_i x_i ||a_i||_1$$
    which is a restatement of RIP-1 [Price’10]
## 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>I² / I²</td>
</tr>
<tr>
<td>[CM’04]</td>
<td>R</td>
<td>k log⁶ n</td>
<td>n log⁶ n</td>
<td>log⁶ n</td>
<td>k log⁶ n</td>
<td>I² / I²</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>n⁶</td>
<td>I² / I²</td>
</tr>
<tr>
<td>[GSTV’06] [GSTV’07]</td>
<td>D</td>
<td>k log⁶ n</td>
<td>n log⁶ n</td>
<td>log⁶ n</td>
<td>k log⁶ n</td>
<td>I² / I²</td>
</tr>
<tr>
<td>[BGIKS’08] [GLR’08]</td>
<td>D</td>
<td>k log(n/k)</td>
<td>n log(n/k)</td>
<td>log(n/k)</td>
<td>n⁶</td>
<td>I² / I²</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>I² / I²</td>
</tr>
<tr>
<td>[IR’08], [BIR’08], [BI’09], [P’10]</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>I² / I²</td>
</tr>
<tr>
<td>[GLSP’09]</td>
<td>R</td>
<td>k log(n/k)</td>
<td>n log⁶ n</td>
<td>log⁶ n</td>
<td>k log⁶ n</td>
<td>I² / I²</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)