Compressive Sensing:
Theory, Applications, Challenges and Future

Dr. Mohammad Eslami and Seyed Hamid Safavi

Shahid Beheshti University
Digital Signal Processing Laboratory (DiSPLaY)
http://display.sbu.ac.ir
m_eslami@sbu.ac.ir, h_safavi@sbu.ac.ir

21 May 2017
Outline

1. Introduction
2. Compressive Sensing
   - Recovery Constraints
     - Spark
     - NSP
     - RIP
     - Mutual Coherence
   - Recovery Algorithms
     - MP
     - OMP
     - BP
     - $\ell_0$-regularized algorithms
3. Challenges
4. Applications
5. Solvers
   - SparseLab
   - CVX
   - $\ell_1$-Magic
   - SPGL1
   - SL0
   - ISTA, FISTA
   - ADMM
6. Summary and Future Directions
Introduction
\[ x \in \mathbb{R}^n \]
\[ x = D\alpha \quad D \in \mathbb{R}^{n \times m} \quad \alpha \in \mathbb{R}^m \]

Atom Composition

\[ \|\alpha\|_0 = k_0 \quad k_0 \ll m \quad k_0 - \text{sparse} \]
\[ x = D\alpha + e \quad \|e\|_2 \leq \varepsilon \quad \mathcal{M}(D, k_0, \varepsilon) \]
Atomic Decomposition

\[ \text{P}_0(D, x, \delta) : \min_{\alpha} \|\alpha\|_0 \quad \text{subject to} \quad \|x - D\alpha\|_2 < \delta \]

\[ \text{P}_1(D, x, \delta) : \min_{\alpha} \|\alpha\|_1 \quad \text{subject to} \quad \|x - D\alpha\|_2 < \delta \]
Inverse Problem

\[ y = Hx + \nu \quad \|\nu\|_2 < \delta \]

\[ P_0(\mathbf{H}, y, \delta + \varepsilon) \quad \hat{x} = D\alpha_0^{\delta+\varepsilon} \]
In this equation:

- $\alpha_0^{\delta+\varepsilon}$ is the minimization of $\|\alpha\|_0$ subject to $\|y - HD\alpha\|_2 < \delta + \varepsilon$

- $\hat{x} = D\alpha_0^{\delta+\varepsilon}$

- Similar equation for analysis-based formulation:

$\hat{x} = \min_x \|D^*\alpha\|_0$ subject to $\|y - Hx\|_2 < \delta + \varepsilon$
Compressive Sensing

\[
x, \quad N = 512, \quad 20 - \text{sparse}
\]

\[
y = \Phi x, \quad L = 50
\]
\[ \tilde{x} = \Phi^{-1} y \]
Compressive Sensing

\[ \hat{x} \]
Problem Definition

Linear Inverse Problems

System of linear equations:

\[
\begin{align*}
  y_1 &= a_{11} x_1 + a_{12} x_2 + \ldots + a_{1N} x_N \\
  y_2 &= a_{21} x_1 + a_{22} x_2 + \ldots + a_{2N} x_N \\
  &\vdots \\
  y_M &= a_{M1} x_1 + a_{M2} x_2 + \ldots + a_{MN} x_N
\end{align*}
\]

Or in a matrix form:

\[ y = Ax \]

**Applications of Linear inverse problems**: optics, radar, acoustics, communication theory, signal processing, medical imaging, computer vision, geophysics, oceanography, astronomy, remote sensing, natural language processing, machine learning, nondestructive testing, and many other fields.
Problem Definition

Linear Inverse Problems

\( y \): Measurements,
\( A \in \mathbb{R}^{M \times N} \): Linear Measurement Matrix

- \( M = N \), In this case the \( A \) is square matrix and the system has a unique solution if \( \det(A) \neq 0 \).
- \( M < N \), the problem is underdetermined (fewer equations than unknowns). An underdetermined linear system has either no solution or infinitely many solutions.
- \( M > N \), the problem is overdetermined (more equations than unknowns). An overdetermined system is almost always inconsistent (it has no solution) when constructed with random coefficients. Homogeneous case has a trivial all zero solution.
Problem Definition

ill-posedness

\[ f(\theta) = 1 - \theta^2 \]

Non-existence: No \( \theta_3 \) such that \( f(\theta_3) = y_3 \)

Non-uniqueness: \( y_j = f(\theta_j) = f(\tilde{\theta}_j) \quad j = 1, 2 \)

Lack of continuity of inverse map:

\[ |y_1 - y_2| \text{ small } \not\Rightarrow |f^{-1}(y_1) - f^{-1}(y_2)| = |\theta_1 - \tilde{\theta}_2| \text{ small} \]
**Problem Definition**

<table>
<thead>
<tr>
<th><strong>ill-posedness</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Q.</strong> what is ill-posedness?</td>
</tr>
<tr>
<td><strong>A.</strong> unstability with respect to measurement errors.</td>
</tr>
<tr>
<td><strong>Q.</strong> How to deal with ill-posed inverse problems?</td>
</tr>
<tr>
<td><strong>A.</strong> It depends! There is no universal method for solving ill-posed problems. In every specific case, the main “trouble” — instability — has to be tackled in its own way.</td>
</tr>
<tr>
<td>“All happy families resemble one another, each unhappy family is unhappy in its own way”. Leo Tolstoy</td>
</tr>
</tbody>
</table>

**Solution:** Adding regularization such as Tikhonov regularization, $\ell_1$-regularization, etc.
Finding fake coin between coins?

سکه تقلبی
عدد سکه داریم. می‌دانیم که یکی از این سکه‌ها تقلبی است. این سکه تقلبی دارای وزن کمتر (بیشتر) نسبت به سایر سکه‌ها است. در ضمن وزن سکه سالم را می‌دانیم. حال سوال این است که آیا می‌توان با 3 بار وزن کردن با ترازوی یک کفهای دقیق دیجیتال و به صورت غیرفوقی (non-Adaptive) سکه ناسالم را تشخیص داد؟
با اندکی تأمل به راحتی می‌توان متوجه شد که روش زیر می‌تواند ما را به نتیجه برساند. ابتدا به هر سکه برچسبی بین ۱ تا ۷ می‌زنیم. سپس در اندازه‌گیری اول سکه‌های ۵، ۱، ۳، ۰ و ۷ را با هم اندازه‌گیری کنیم. در اندازه‌گیری دوم سکه‌های ۴، ۲، ۰ و ۷ را با هم اندازه‌گیری کنیم و در نهایت در اندازه‌گیری سوم سکه‌های ۴، ۲، ۰ و ۷ را با هم اندازه‌گیری کنیم. می‌توانیم روش بالا را به صورت ماتریسی نمایش دهیم که امین سطر نشان می‌دهد کدام یک از سکه‌ها در این اندازه‌گیری سه‌هم بوده‌اند.

\[
\Phi = \begin{bmatrix}
1 & 0 & 1 & 1 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

توجه کنید که امین سطر معادل باینری عدد \(k\) است. برای متوجه شدن عملکرد این روش فرض کنید آزمایش اول و دوم با نتیجه‌ای مورد انتظار (۴ برابر وزن سکه سالم) مطابقت نداشته باشد، در این صورت چون فقط سکه ۳ در این دو آزمایش است و در آزمایش سوم وجود ندارد، پس متوجه می‌شویم سکه ۳ همان سکه تقلبی است. اگر وزن سکه‌ها را به صورت \(c = [c_1, c_2, \ldots, c_7]^T\) و نتایج آزمایش را به صورت \(\Phi c = m\) دهیم. می‌توانیم وزن کردن‌های بالا را به صورت \(m = [m_1, m_2, m_3]^T\) در نظر بگیریم، نکته وزن سکه اصلی (\(a\)) را می‌دانیم می‌توانیم بردار \(x\) به صورت \(x = c - [a, \ldots, a]^T\) حائز اهمیت در \(x\) این است که با توجه به فرض سوال ۶ تا از درایه‌های این بردار ۰ است و فقط در ایزدی ناصفر دارد.
Finding fake coin between coins?

حالت ديگر این سوال این است که فرض کنیم به جای ترازوی یک کفه‌ای دیجیتال یک ترازوی دوکفه‌ای داریم. یعنی با توجه به هر گونه توزیع سکه که در دو طرف کفه‌ها است ما فقط می‌توانیم متوجه شویم که کفه‌ی راست وزن بیشتری دارد یا کفه‌ی چپ یعنی خود را به علامت محدود کرده.
در اینجا هم می‌توان مانند قسمت قبل در نظر گرفت فقط چون در هر آزمایش ۴ سکه شرکت می‌کنند می‌توان ۲ تا از آنها را در یک طرف و ۲ تای دیگر را در طرف دیگر قرار می‌دهیم. یعنی این بار معادلات به صورت زیر است:

\[
\begin{bmatrix}
1 & 0 & 0 & -1 \\
0 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 \\
\end{bmatrix}
\begin{bmatrix}
c_1 \\
c_2 \\
c_3 \\
\end{bmatrix}
= \begin{bmatrix}
m_1 \\
m_2 \\
m_3 \\
\end{bmatrix}
\]

البته با توجه به روش وزن کردن این بار فقط ما علامت بردار \( m \) را داریم. واضح است که به جای بردار \( c \) استفاده کنیم (چرا؟). حال به عنوان مثال اگر در هر ۳ آزمایش به نتیجه غیر صفر علامت دار) بررسی می‌شود که سکه \( v \) همان سکهی تقلبی است.
Imaging at wavelengths where silicon is blind is considerably more complicated, bulky, and expensive. Thus, for comparable resolution, a US$500 digital camera for the **visible** becomes a US$50,000 camera for the **infrared**. 
Data with **high-frequency** content is often not intrinsically **high-dimensional**.

Signals often obey low-dimensional models such as:
- Sparsity
- Low-rank matrices
- Manifolds

The intrinsic dimension can be **much less** than the ambient dimension.

Courtesy of Dr. Mark Davenport
Success has many fathers ...

**Sampling Theorem**: sampling at twice the highest frequency.

**Compressive Sensing**: sampling at sub-Nyquist rate!

Whittaker, Nyquist, Kotelnikov, Shannon

- “Can we not just directly measure the part that will not end up being thrown away?” [Donoho, 2006]
- “why spend so much effort acquiring all the data when we know that most of it will be discarded?” [Candes, 2006]

Donoho, Candes, Romberg, Tao
“Measure what **should** be measured ...”

**Compressive Sensing**

1. **Transform Sparsity**
   - Enable solving underdetermined linear inverse problems

2. **Non-adaptive Linear Sampling**
   - Spark
   - Null Space Property
   - RIP
   - Mutual Coherence
   - Greedy Algorithms (MP, OMP, StOMP, …)
   - Basis Pursuit (BP), Basis Pursuit Denoising (BPDN)
   - LASSO
   - Smoothed L0 (SL0)
   - …

3. **Reconstruction**
Single Pixel Camera

Courtesy of Dr. Marco F. Duarte and Dr. Richard G. Baraniuk
Problem Statement

Definition:
We say that a signal $x$ is $K$-sparse when it has at most $K$ nonzeros, i.e., $\|x\|_0 \leq k$. Also, we let $\Sigma_k = \{x : \|x\|_0 \leq k\}$ denote the set of all $K$-sparse signals.

Figure: (a) Measurement process with a random Gaussian matrix $\Phi$ and discrete cosine transform (DCT) matrix $\Psi$. (b) The measurement vector $y$ is a linear combination of 4 columns in dictionary [Baraniuk2007]
Recovery Constraints
Recovery Constraints

\( P_0 : \min_{s} \|s\|_0 \quad \text{subject to } y = \Phi \Psi s \)

\( P_1 : \min_{s} \|s\|_1 \quad \text{subject to } y = \Phi \Psi s \)

Figure: (a) The set of all k-sparse (k=2) vectors in \( R^3 \). (b) Visualization of \( \ell_2 \) minimization. (c) Visualization of \( \ell_1 \) minimization. [Baraniuk, 2007]
Recovery Constraints [Davenport, 2011]

- **Spark** (for sparse signals) \( \sigma(\Phi) > 2k \).
- **Null Space Property** (for approximately sparse signals)
  \[ \|h^{\Lambda}\|_2 < \frac{C_{NSP}}{\sqrt{k}} \|h^{\Lambda^c}\|_1. \]
- **Restricted Isometry Property** (for approximate sparse signals with noise)
  \( (1 - \delta_k)\|x\|_2^2 \leq \|\Phi x\|_2^2 \leq (1 + \delta_k)\|x\|_2^2. \)
- **Mutual Coherence** \( \mu(\Phi) = \max_{i \neq j} \frac{|\phi_i^T \phi_j|}{\|\phi_i\|_2 \|\phi_j\|_2} \)

Spark(\( \Phi \)) \( \geq 1 + \frac{1}{\mu(\Phi)} \)
Definition:
The spark of a given matrix $\Phi$ is the smallest number of columns of $\Phi$ that are linearly dependent.

$$\text{Spark}(\Phi_{M \times N}) \triangleq \min \{ k; \exists 1 \leq i_1 \leq \ldots \leq i_k \leq N, \alpha_1, \ldots, \alpha_k \in \mathbb{R},$$

$$\|\alpha\|_2 \neq 0, \sum_{j=1}^{k} \alpha_j \Phi_{i_j} = 0 \}$$

$$\Phi = \begin{bmatrix}
1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

$$\begin{cases}
\text{Rank}(\Phi) = 4 \\
\text{Spark}(\Phi) = 3 \\
\Phi_1 + \Phi_2 - \Phi_5 = 0
\end{cases}$$
Theorem:
For any vector \( y \in \mathbb{R}^M \), there exists at most one signal \( x \in \Sigma_k \) such that \( y = \Phi x \) if and only if \( \sigma(\Phi) > 2k \).

Proof:
If we wish to be able to recover all sparse signals \( x \) from the measurements \( \Phi x \), then it is immediately clear that for any pair of distinct vectors \( x_1, x_2 \in \Sigma_k \), we must have \( \Phi x_1 \neq \Phi x_2 \), since otherwise it would be impossible to distinguish \( x_1 \) from \( x_2 \) based solely on the measurements. Mathematically

\[
\text{if } x_1 \neq x_2, \text{ and } x_1, x_2 \text{ k-sparse } \Rightarrow \Phi x_1 \neq \Phi x_2 \Rightarrow \Phi (x_1 - x_2) \neq 0
\]

Note:
\[
\text{Spark}(\Phi_{M \times N}) \in [2, M + 1]
\]
Null Space Property [Davenport, 2011]

**Definition:**

A matrix $\Phi$ satisfies the null space property (NSP) of order $K$ if there exists a constant $C_{NSP} > 0$ such that,

\[ \forall h \in \mathcal{N}_\Phi, \forall \Lambda \subseteq \{1, ..., N\}, |\Lambda| = k, \Lambda^c = \{1, ..., N\} \setminus \Lambda \]

\[ \|h_\Lambda\|_2 < \frac{C_{NSP}}{\sqrt{k}} \|h_{\Lambda^c}\|_1 \]

**Note:**

we must also ensure that $\mathcal{N}_\Phi$ does not contain any vectors that are too compressible in addition to vectors that are sparse.

**Theorem:**

Any $K$-sparse vector $x$ is a unique solution to $P_1$ if and only if matrix $\Phi$ satisfies NSP of order $K$ with $C_{NSP} = 1$. 
**Definition:**

A matrix $\Phi$ satisfies the restricted isometry property (RIP) of order $K$ if there exists a $\delta_k \in (0, 1)$ such that

$$(1 - \delta_k) \|x\|_2^2 \leq \|\Phi x\|_2^2 \leq (1 + \delta_k) \|x\|_2^2$$

holds for all $x \in \Sigma_K$.

**Theorem: Noiseless recovery**

Assume that $\delta_{2k} < \sqrt{2} - 1$. Then the solution $x^*_N \times 1$ to $\ell_1$ minimization obeys

$$\|x^* - x\|_2 \leq \frac{C_0}{\sqrt{k}} \|x - x^{(k)}\|_1$$

where $x^{(k)}_N \times 1$ is the best $K$-sparse approximate of $x$ and $C_0$ is a constant.
Theorem: Recovery in the presence of bounded noise

Assume that $\delta_{2k} < \sqrt{2} - 1$ and let $y = \Phi x + e$, where $\|e\|_2 \leq \epsilon$. Then the solution $x^*_{N\times 1}$ to $\ell_1$ minimization obeys

$$\|x^* - x\|_2 \leq \frac{C_0}{\sqrt{k}} \left\| x - x^{(k)} \right\|_1 + C_1 \epsilon$$

where $x^{(k)}_{N\times 1}$ is the best $K$-sparse approximate of $x$ and $C_0$ and $C_1$ is a constant as follows:

$$C_0 = 2 \frac{1 - (1 - \sqrt{2}) \delta_{2k}}{1 - (1 + \sqrt{2}) \delta_{2k}}, \quad C_1 = 4 \frac{\sqrt{1 + \delta_{2k}}}{1 - (1 + \sqrt{2}) \delta_{2k}}$$
Theorem: The relationship between the RIP and the NSP

Suppose that $\Phi$ satisfies the RIP of order $2k$ with $\delta_{2k} < \sqrt{2} - 1$. Then $\Phi$ satisfies the NSP of order $2k$ with constant

$$C_0 = \frac{\sqrt{2} \delta_{2k}}{1 - (1 + \sqrt{2}) \delta_{2k}}$$
**Mutual Coherence**

**Definition:** The coherence of a matrix $\Phi$, $\mu(\Phi)$, is the largest absolute inner product between any two columns $\phi_i, \phi_j$ of $\Phi$:

$$\mu(\Phi) = \max_{i \neq j} \frac{|\phi_i^T \phi_j|}{\|\phi_i\|_2 \|\phi_j\|_2}$$

**Lemma:** For any matrix $\Phi$

$$\text{Spark}(\Phi) \geq 1 + \frac{1}{\mu(\Phi)}$$

**Theorem:** If $k < \frac{1}{2} \left(1 + \frac{1}{\mu(\Phi)}\right)$ then for each measurement vector $y \in \mathbb{R}^M$ there exists at most one signal $x \in \Sigma_k$ such that $y = \Phi x$.

**Welch Bound:**

It is possible to show that the lower bound for coherence of a matrix is:

$$\mu \geq \sqrt{\frac{N-M}{M(N-1)}}$$

Recovery Constraints [Davenport, 2011]
Recovery Constraints

Courtesy of the concept by Dr. Arash Amini
Measurement Matrices

How should we design the measurement matrix so that the required number of measurements becomes as small as possible?

- Gaussian random matrices
- Bernulli random matrices
- Hadamard matrices
- Fourier matrices
- Deterministic matrices

Measurements bounds

Let $\Phi$ be an $M \times N$ matrix that satisfies the RIP of order $2k$ with constant $\delta \in (0, \frac{1}{2}]$. Then

$$M \geq C \left( K \log \left( \frac{N}{K} \right) \right)$$

where $C = \frac{1}{2} \log \left( \sqrt{24} + 1 \right) \approx 0.28$.

How can we recover the desired signal from the measurements?
Recovery Algorithms
Recovery Algorithms

Designing Criteria:

- Minimal number of measurements
- Robustness to measurement noise and model mismatch
- Speed
- Performance guarantees

Greedy Algorithms

<table>
<thead>
<tr>
<th>Matching Pursuit (MP)</th>
<th>Orthogonal MP (OMP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stage-wise OMP</td>
<td>Subspace Pursuit (SP)</td>
</tr>
<tr>
<td>CoSaMP</td>
<td>Iterative Hard Thresholding (IHT)</td>
</tr>
</tbody>
</table>

Convex Optimization based Methods

<table>
<thead>
<tr>
<th>Basis Pursuit (BP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basis Pursuit Denoising (BPDN)</td>
</tr>
<tr>
<td>LASSO</td>
</tr>
</tbody>
</table>

Smoothed $\ell_0$
Matching Pursuit (MP)

\[ y = Dx \Rightarrow y = \sum_{i=1}^{k} x_{a(i)}^{(i)} d_{a(i)} + r_k = y^{(k)} + r^{(k)} \]

\[ D = [d_1, \ldots, d_N] \]

که در آن \( y^{(k)} \) تقریب در مرحله \( k \) باقیمانده خطا در مرحله \( k \) است و \( r^{(k)} \) مجموعه‌ای از بردارها است که بهترین تطبیق داده بر روی این بردارها مانند \( m \) باشد و \( a(i) \) اندیس ستون انتخاب شده در مرحله \( i \) ام می‌باشد. \( y^{(i)} \) نیز ضریب ستون انتخاب شده در مرحله \( i \) ام می‌باشد.

الگوریتم جستجوی تطبیق (MP):

\[ r^{(0)} = y \]

1. مقدار دهی اولیه: \( f \) باقیمانده در شروع الگوریتم

\[ x^{(0)} = [x_1^{(0)}, \ldots, x_N^{(0)}] = 0 \]

ب) بردار ضرایب دیکشنری

ج) شماره مرحله \( k = 1 \)

\[ c^{(i)} = [c_1^{(i)}, \ldots, c_N^{(i)}] = D^T r^{(i-1)} \]

\[ d^{(i)} = \arg \max_j |c_j^{(i)}| \]

\[ a^{(i)} = \arg \max_j |c_j^{(i)}| \]

الف) به روز کردن نمایش تتنک:

\[ x_{a^{(i)}}^{(i)} = x_{a^{(i)}}^{(i-1)} + c_{a^{(i)}}^{(i)} \]

ب) به روز کردن باقیمانده:

\[ r^{(i)} = r^{(i-1)} - d_{a^{(i)}} c_{a^{(i)}}^{(i)} \]

\[ k \leftarrow k + 1 \]

و مراحل 1 تا 3 را با بزرگی شرط توقف ادامه بده.
الگوریتم جستجوی تطبیق متعامد (OMP):

\[ \mathbf{x}^{(0)} = [x_1^{(0)}, \ldots, x_N^{(0)}] = \mathbf{0} \]

ب) بردار ضرایب دیکشنری 

\[ k = 1 \]

\[ \text{شماره مرحله} \]

\[ c^{(i)} = [c_1^{(i)}, \ldots, c_N^{(i)}] = \mathbf{D}^T \mathbf{r}^{(i-1)} \]

الف) محاسبه بردار همبستگی ستون‌های دیکشنری با باقیمانده: 

\[ a^{(i)} = \underset{j}{\text{argmax}} |c_j^{(i)}| \]

الف) انتخاب بهترین ستون از دیکشنری: 

\[ \Gamma^{(i)} = \Gamma^{(i-1)} \cap \{a^{(i)}\} \]

الف) اضافه کردن ستون انتخاب شده به مجموعه ستون‌های فعال: 

\[ \mathbf{x}^{(i)} = \mathbf{D}_{\Gamma^{(i)}}^H \mathbf{y} \]

الف) به روش کردن بردار نمايش تنگک: 

\[ \mathbf{r}^{(i)} = \mathbf{y} - \mathbf{Dx}^{(i)} \]

الف) به روش کردن باقیمانده: 

و مراحل ۱ تا ۳ را تا برقراری شرط توقف ادامه بده.

\[ k \leftarrow k + 1 \]
$l_1$-minimization:

- **Basis Pursuit (BP)**

\[
P_1 : \min_{s \in \mathbb{R}^N} \|s\|_1 \\
\text{s.t. } y = \Phi \Psi s
\]

- **Basis Pursuit Denoising (BPDN)**

\[
\begin{align*}
&\min_{s \in \mathbb{R}^N} \|s\|_1 \\
&\text{s.t. } \|y - \Phi \Psi s\|_2^2 \leq \delta
\end{align*}
\]

- **Least Absolute Shrinkage and Selection Operator (LASSO)**

\[
\begin{align*}
&\min_{s \in \mathbb{R}^N} \frac{1}{2} \|y - \Phi \Psi s\|_2^2 + \lambda \|s\|_1 \\
&\text{s.t. } \|s\|_1 \leq \tau
\end{align*}
\]
\[ \lim_{\sigma \to 0} f_\sigma (x_i) = \lim_{\sigma \to 0} \exp \left( - \frac{x_i^2}{2\sigma^2} \right) = \begin{cases} 1 & \text{if } x_i = 0 \\ 0 & \text{if } x_i \neq 0 \end{cases} \]

\[ F_\sigma (\mathbf{x}) = \sum_{i=1}^{N} f_\sigma (x_i) \rightarrow \|\mathbf{x}\|_0 \approx N - F_\sigma (\mathbf{x}) \]

- SL0 is an algorithm for finding the sparsest solutions of an underdetermined system of linear equations.
- SL0 is a very fast algorithm. For example, it is about 2 to 3 orders of magnitude faster than \( \ell_1 \)-magic.
- SL0 tries to directly minimize the L0 norm. This is contrary to most of other existing algorithms (e.g. Basis Pursuit), which replace L0 norm by other cost functions (like L1). Note also that the equivalence of minimizing L1 and L0 is only asymptotic, and does not always hold.

**Software Link**: http://ee.sharif.edu/~SLzero
Challenges
Galileo Galilei

“Measure what can be measured and make measurable what cannot be measured” This quote attributed to Galileo Galilei.

Thomas Strohmer

“Measure what should be measured”

Challenges:

- Structured Sensing Matrices
- Structured Sparsity and Prior Information
- Non-Linear Compressive Sensing
- Better Bounds for Compressive Sensing
- Hardware Implementation
Structured Sensing Matrices

Question:
Can we use a novel structure for sensing matrices that results in a good performance?

- We usually do not have luxury to choose measurement matrix as we please.
- Measurement matrix is often dictated by the physical properties of the measurement process. e.g. the laws of wave propagation.
- Specific structure in measurement matrix can give rise to fast algorithms which will significantly speed up recovery algorithms.

Structure: Block based compressive sensing, Separable matrices like Kronecker product matrices

Remark: The typical measurement matrix is not Gaussian or Bernoulli, but one with a very specific structure.
Structured Sensing Matrices

**Question:**
Can we use a novel structure for sensing matrices that results in a good performance?

- We usually do not have luxury to choose measurement matrix as we please.
- Measurement matrix is often dictated by the physical properties of the measurement process. e.g. the laws of wave propagation.
- Specific structure in measurement matrix can give rise to fast algorithms which will significantly speed up recovery algorithms.

**Structure:** Block based compressive sensing, Separable matrices like Kronecker product matrices

**Remark:** The typical measurement matrix is not Gaussian or Bernoulli, but one with a very specific structure.
Structured Sparsity and Prior Information

**Question:**
How can we best take advantage of the knowledge that all sparsity patterns may not be equally likely in a signal?

**Structured Sparsity**

**Block Sparsity**

\[
y = \begin{bmatrix} A_1 & \cdots & A_i & \cdots & A_n \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_i \\ \vdots \\ x_n \end{bmatrix}
\]

- \( A_1 \) - columns 1, 2, \ldots, \( d \)
- \( A_i \) - columns \( id - d + 1, id - d + 2, \ldots, id \)
- \( A_n \) - columns \( nd - d + 1, nd - d + 2, \ldots, nd \)

\[
y = Ax = \sum_{i=1}^{n} A_i x_i
\]

**Tree Structure of Wavelet Coefficients**
Non-Linear Compressive Sensing

Question:

For which types of nonlinear measurements can we build an interesting and relevant compressive sensing theory?

- In many applications we can only take **nonlinear measurements**.
- An important example is the case where we observe **signal intensities**. So, the phase information is missing.
- **Phase retrieval**: X-ray crystallography, diffraction imaging, astronomy, and quantum tomography.
- **Quantized samples**, and in the extreme case, 1-bit measurements.
Question:

For which types of nonlinear measurements can we build an interesting and relevant compressive sensing theory?

- In many applications we can only take **nonlinear measurements**.
- An important example is the case where we observe **signal intensities**. So, the phase information is missing.
- **Phase retrieval**: X-ray crystallography, diffraction imaging, astronomy, and quantum tomography.
- **Quantized samples**, and in the extreme case, 1-bit measurements.
Weighted CS

Figure: **Enhancing Sparsity**: At the origin, the canonical \( \ell_0 \) sparsity count \( f_0(t) \) is better approximated by the log-sum penalty function \( f_{\log,\epsilon}(t) \) than by the traditional convex \( \ell_1 \) relaxation \( f_1(t) \) [Candes2008].

\[
\begin{align*}
P_{\ell_1} & : \min_{s \in \mathbb{R}^N} \|s\|_1 \quad \text{s.t.} \quad y = \theta s \\
P_{RW\ell_1} & : \min_{s \in \mathbb{R}^N} \sum_{i=1}^{N} \log (|s_i| + \epsilon) \quad \text{s.t.} \quad y = \theta s
\end{align*}
\]

Solve using Majorization Minimization:

\[
\begin{align*}
P_{RW\ell_1} & : \min_{s \in \mathbb{R}^N} \|Ws\|_1 \quad \text{s.t.} \quad y = \theta s, \quad P_{RW\ell_1} & : \min_{z \in \mathbb{R}^N} \|z\|_1 \quad \text{s.t.} \quad y = \theta W^{-1}z
\end{align*}
\]

\[
w_i^{(l+1)} = \frac{1}{|s_i^{(l)}| + \epsilon}
\]
How we can avoid acquisition of perceptual redundancies in compressive sensing?

Answer: Perceptual Weighted Compressive Sensing

Question
How we can avoid acquisition of perceptual redundancies in compressive sensing?

Answer: Perceptual Weighted Compressive Sensing

Figure: Contrast Sensitivity Function (CSF)

\[ H(f_{i,j}) = 2.6 \left( 0.0192 + 0.114f_{i,j} \right) e^{-\left(0.114f_{i,j}\right)^{1.1}} \]

\[ f_{i,j} = \frac{1}{2N} \sqrt{\left( \frac{i}{\theta_x} \right)^2 + \left( \frac{j}{\theta_y} \right)^2} \ \text{(cpd)} \]

Diffusive Compressive Sensing

Applications:

- Thermal monitoring of CPU hot spots to avoid leakage and reduce the CPU time failure
- Climate change study
- Environmental Monitoring
- ...

Figure: Thermal Field example: Localized Source

Thermal Field Reconstruction

Side Information: 2D Partial Differential Equation (PDE)

\[
\frac{\partial f(x, y, t)}{\partial t} = \gamma \left( \frac{\partial^2 f(x, y, t)}{\partial x^2} + \frac{\partial^2 f(x, y, t)}{\partial y^2} \right), \quad t, \gamma > 0.
\]

Frame Processing:

\[
f_t^{t_0}(x, y) = \gamma \left( \frac{\partial^2 f_t^{t_0}(x, y)}{\partial x^2} + \frac{\partial^2 f_t^{t_0}(x, y)}{\partial y^2} \right)
\]

where, \( f_t^{t_0}(x, y) = \frac{\partial f(x, y, t)}{\partial t} \bigg|_{t=t_0} \). The above equation can be discretized as follows:

\[
b = \gamma (D_x D_x + D_y D_y) f.
\]

Figure: Geometrical depiction of the proposed method. This figure intuitively explains why adding the additional constraint can help to decrease overcompleteness of the CS problem.

Visualization of 10'th frame of reconstructed fields when $x_d = y_d = 4$ for setting A (first row) and setting B (second row): (a,e) Original field, reconstructed fields for (b,f) CS, (c,g) DCS-I, (d,h) DCS-II.
Applications
Solvers
SparseLab

Functions

- SolveBP($\mathbf{A}$, $\mathbf{y}$, $N$, maxIters, lambda, OptTol)
- SolveLasso($\mathbf{A}$, $\mathbf{y}$, $N$, algType, maxIters, lambdaStop, resStop, solFreq, verbose, OptTol)
- SolveMP($\mathbf{A}$, $\mathbf{y}$, $N$, maxIters, lambdaStop, solFreq, verbose, OptTol)
- SolveOMP($\mathbf{A}$, $\mathbf{y}$, $N$, maxIters, lambdaStop, solFreq, verbose, OptTol)
- SolveStOMP($\mathbf{A}$, $\mathbf{y}$, $N$, thresh, param, maxIters, verbose, OptTol)

Software Link: https://sparselab.stanford.edu
SparseLab

**SolveBP**

- **A**: Either an explicit $M \times N$ matrix, with $\text{rank}(A) = \min(M, N)$ by assumption, or a string containing the name of a function implementing an implicit matrix.
- **y**: Measurement vector of length $M$.
- **N**: length of the solution vector $s$
- **maxIters**: maximum number of PDCO iterations to perform, default 20.
- **lambda**: If 0 or omitted, Basis Pursuit is applied to the data, otherwise, Basis Pursuit Denoising is applied with parameter lambda (default 0).
- **OptTol**: Error tolerance, default $1e^{-3}$. 

Compressive Sensing
**SparseLab**

**SolveLasso**

- **A**: Either an explicit $M \times N$ matrix, with $\text{rank}(A) = \min(M, N)$ by assumption, or a string containing the name of a function implementing an implicit matrix.
- **y**: Measurement vector of length $M$.
- **N**: length of the solution vector $s$
- **algType**: 'lars' for the Lars algorithm, 'lasso' for lars with the lasso modification (default). Add prefix 'nn' (i.e. 'nnlars' or 'nnlasso') to add a non-negativity constraint (omitted by default)
- **maxIters**: maximum number of Lars iterations to perform. If not specified, runs to stopping condition (default)
- **lambdaStop**: If specified (and $> 0$), the algorithm terminates when the Lagrange multiplier $\leq$ lambdaStop.
resStop: If specified (and $> 0$), the algorithm terminates when the L2 norm of the residual $\leq$ resStop.

solFreq: if $= 0$ returns only the final solution, if $> 0$, returns an array of solutions, one every solFreq iterations (default 0).

verbose: 1 to print out detailed progress at each iteration, 0 for no output (default)

OptTol: Error tolerance, default $1e^{-5}$.
SolveMP and SolveOMP

- **A**: Either an explicit $M \times N$ matrix, with $\text{rank}(A) = \text{min}(M, N)$ by assumption, or a string containing the name of a function implementing an implicit matrix.

- **y**: Measurement vector of length $M$.

- **N**: Length of the solution vector $s$.

- **maxIters**: Maximum number of iterations to perform. If not specified, runs to stopping condition (default).

- **lambdaStop**: If specified, the algorithm stops when the last coefficient entered has residual correlation $\leq \text{lambdaStop}$.

- **solFreq**: If $= 0$ returns only the final solution, if $> 0$, returns an array of solutions, one every solFreq iterations (default 0).

- **verbose**: 1 to print out detailed progress at each iteration, 0 for no output (default).

- **OptTol**: Error tolerance, default $1e^{-5}$. 
CVX is a Matlab-based modeling system for convex optimization. CVX turns Matlab into a modeling language, allowing constraints and objectives to be specified using standard Matlab expression syntax. Website link: http://cvxr.com/cvx

**Notation**

**BP:**

\[ \text{cvx}_\text{begin} \]

\[ \text{variables } s(N) \]

\[ \text{minimize } (\text{norm}(s,1)) \]

\[ \text{subject to } \]

\[ y == \Phi \Psi s \]

\[ \text{cvx}_\text{end} \]

**BPDN:**

\[ \text{cvx}_\text{begin} \]

\[ \text{variables } s(N) \]

\[ \text{minimize } (0.5 * \text{norm}(y - \Phi \Psi s,2) + \lambda \text{norm}(s,1)) \]

\[ \text{cvx}_\text{end} \]

**Software Link:** http://cvxr.com/cvx
L1-MAGIC is a collection of MATLAB routines for solving the convex optimization programs central to compressive sampling. The algorithms are based on standard interior-point methods, and are suitable for large-scale problems.

- **l1decode_pd**: Decoding via linear programming. Solve $\min_x \|b - Ax\|_1$. Recast as the linear program and solve using primal-dual interior point method.
- **l1eq_pd**: Solve $\min_x \|x\|_1 \text{ s.t. } Ax = b$. Recast as linear program and use primal-dual interior point method.
- **l1qc_logbarrier**: Solve quadratically constrained l1 minimization: $\min \|x\|_1 \text{ s.t. } \|Ax - b\|_2 \leq \epsilon$. Reformulate as the Second-Order Cone Program (SOCP) and use a log barrier algorithm.

**Software Link**: [https://statweb.stanford.edu/~candes/l1magic](https://statweb.stanford.edu/~candes/l1magic)
SPGL1: A solver for large-scale sparse reconstruction

SPGL1 is a Matlab solver for large-scale one-norm regularized least squares. It is designed to solve any of the following three problems:

1. **BP**: minimize $\|s\|_1$ subject to $As = b$,
2. **BPDN**: minimize $\|s\|_1$ subject to $\|As - b\| \leq \sigma$,
3. **LASSO**: minimize $\|As - b\|_2$ subject to $\|s\|_1 \leq \tau$.

SPGL1 relies only on matrix-vector operations $As$ and $A^T y$ and accepts both explicit matrices and functions that evaluate these products. SPGL1 is suitable for problems that are in the complex domain.

**Software Link:**
http://www.cs.ubc.ca/~mpf/spgl1/downloads/spgl1-1.9.zip
SPGL1: A solver for large-scale sparse reconstruction

- `spg_bp`: Solve the basis pursuit (BP) problem.
  \[ [s, r, g, info] = spg_bp(A, y, options) \]

- `spg_bpdn`: Solve the basis pursuit Denoising (BPDN) problem.
  \[ [s, r, g, info] = spg_bpdn(A, y, sigma, options) \]

- `spg_lasso`: Solve the LASSO problem.
  \[ [s, r, g, info] = spg_lasso(A, y, tau, options) \]

- `spg_group`: Solve jointly-sparse BPDN.
  \[ [s, r, g, info] = spg_group(A, y, groups, sigma, options) \]

- `spg_mmv`: Solve multi-measurement BPDN.
  \[ [s, r, g, info] = spg_mmv(A, y, sigma, options) \]

- `spgl1`: Solve BP, BPDN, and LASSO.
  \[ [s, r, g, info] = spgl1(A, y, tau, sigma, s, options) \]
SL0: Smoothed L0 (SL0) Algorithm

Function

\[
s = \text{SL0}(A, y, \text{sigma\_min}, \text{sigma\_decrease\_factor}, \mu_0, L, A\_pinv, \text{true\_s})
\]

- **A**: The Dictionary.
- **y**: Measurement vector of length \( M \).
- **sigma\_min**: The first element of the sequence of sigma is calculated automatically. The last element is given by 'sigma\_min', and the change factor for decreasing sigma is given by 'sigma\_decrease\_factor'.
- **sigma\_decrease\_factor**: The default value of 'sigma\_decrease\_factor' is 0.5. Larger value gives better results for less sparse sources, but it uses more steps on sigma to reach sigma\_min, and hence it requires higher computational cost.
- **\(\mu_0\)**: The value of \(\mu_0\) scales the sequence of mu. For each value of sigma, the value of mu is chosen via \(\mu = \mu_0 \times \text{sigma}^2\). Note that this value affects Convergence. The default value is \(\mu_0 = 2\) (see the paper).
**Function**

- **L**: number of iterations of the internal (steepest ascent) loop. The default value is $L=3$.
- **A\_pinv**: is the pseudo-inverse of matrix $A$ defined by $A\_pinv = A' \ast inv(A \ast A')$. If it is not provided, it will be calculated within the function.
- **true\_s**: is the true value of the sparse solution. This argument is for simulation purposes. If it is provided by the user, then the function will calculate the SNR of the estimation for each value of sigma and it provides a progress report.

**Software Link**: [http://ee.sharif.edu/~SLzero](http://ee.sharif.edu/~SLzero)
Which type of optimization problem is of interest to solve?

$$\min \left\{ F(x) \equiv f(x) + g(x) : x \in \mathbb{R}^N \right\}$$

- $g : \mathbb{R}^N \rightarrow \mathbb{R}$ is a continuous convex function which is possibly non-smooth.
- $f : \mathbb{R}^N \rightarrow \mathbb{R}$ is a smooth convex function, i.e., continuously differentiable with Lipschitz continuous gradient $L(f)$:

$$\|\nabla f(x) - \nabla f(y)\| \leq L(f) \|x - y\| \text{ for every } x, y \in \mathbb{R}^N$$

where $L(f)$ is the Lipschitz constant of $\nabla f$

The $\ell_1$ regularization problem is obviously a special instance of the above problem by substituting $f(x) = \|Ax - b\|^2$, $g(x) = \|x\|_1$. The (smallest) Lipschitz constant of the gradient $\nabla f$ is $L(f) = 2\lambda_{max}(A^T A)$. 

Compressive Sensing

DiSPLaY Group, Shahid Beheshti University, Faculty of Electrical Engineering
Quadratic approximation of $F(x) := f(x) + g(x)$ at a given point $y$:

$$Q_L(x, y) := f(y) + \langle x - y, \nabla f(y) \rangle + \frac{L}{2} \| x - y \|^2 + g(x)$$

It admits a unique minimizer:

$$p_L(y) := \arg \min \left\{ Q_L(x, y) : x \in \mathbb{R}^N \right\}$$

By using simple linear algebra and ignoring constant terms in $y$:

$$p_L(y) = \arg \min_x \left\{ g(x) + \frac{L}{2} \left\| x - \left( y - \frac{1}{L} \nabla f(y) \right) \right\|^2 \right\}$$

Hence, the basic step of ISTA becomes:

$$x_k = p_L(x_{k-1})$$
ISTA: Iterative Shrinkage-Thresholding Algorithm

Solving \( \ell_1 \)-regularization:

\[
x_{k+1} = T_{\lambda t} \left( G \left( x_k \right) \right)
\]

where \( G \left( . \right) \) stands for a gradient step of the fit-to-data LS term and ISTA is an extension of the classical gradient method.

\[
x_{k+1} = T_{\lambda t} \left( x_k - 2tA^T (Ax_k - b) \right)
\]

where \( t = \frac{1}{L(f)} \) is an appropriate stepsize and \( T_\alpha : \mathbb{R}^N \rightarrow \mathbb{R}^N \) is the shrinkage operator defined by

\[
T_\alpha(x)_i = (|x_i| - \alpha)_+ \text{sgn}(x_i)
\]
ISTA: Iterative Shrinkage-Thresholding Algorithm

A possible drawback of this basic scheme is that the Lipschitz constant $L(f)$ is not always known or computable. For instance, the Lipschitz constant in the $\ell_1$-regularization problem depends on the maximum eigenvalue of $A^T A$. For large-scale problems, this quantity is not always easily computable. We therefore also analyze ISTA with a backtracking stepsize rule.

**ISTA with backtracking**

Find the smallest nonnegative integers $i_k$ such that with $\bar{L} = \eta^{i_k} L_{k-1}$,

$$F(p_{\bar{L}}(x_{k-1})) \leq Q_{\bar{L}}(p_{\bar{L}}(x_{k-1}), x_{k-1})$$

Set $L_k = \eta^{i_k} L_{k-1}$ and compute

$$x_k = p_{L_k}(x_{k-1})$$
The ISTA approach has a convergence rate of $O \left( \frac{1}{k} \right)$. i.e it has a sublinear global rate of convergence. Practically, this rate could be somehow slow. Therefore, the question is how we can improve this convergence rate?

**FISTA**

- $L = L(f)$ - A Lipschitz constant of $\nabla f$
- $y_1 = x_0 \in \mathbb{R}^N, t_1 = 1.$
- $x_k = p_L(y_k)$
- $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$
- $y_{k+1} = x_k + \left( \frac{t_k-1}{t_{k+1}} \right) (x_k - x_{k-1})$

The FISTA approach has a convergence rate of $O \left( \frac{1}{k^2} \right)$. We can develop FISTA with backtracing similar to the ISTA approach.
The alternating direction method of multipliers (ADMM) is an algorithm that solves convex optimization problems by breaking them into smaller pieces, each of which are then easier to handle.

Main References

- “Proximal algorithms”, N. Parikh and S. Boyd, 2014

Optimization Problem:

\[
\begin{align*}
\text{min} & \quad f(x) + g(z) \\
\text{s.t.} & \quad Ax + Bz = c
\end{align*}
\]

Software Link: http://stanford.edu/~boyd/admm.html
Alternating Direction Method of Multipliers

Augmented Lagrangian:

\[ L_\rho (x, y, z) = f (x) + g (z) + y^T (Ax + Bz - c) + \frac{\rho}{2} \|Ax + Bz - c\|^2 \]

Solution:

\[ x^{k+1} = \arg \min_x L_\rho \left( x, z^k, y^k \right) \quad \text{x – minimization step} \]
\[ z^{k+1} = \arg \min_z L_\rho \left( x^{k+1}, z, y^k \right) \quad \text{z – minimization step} \]
\[ y^{k+1} = y^k + \rho \left( Ax^{k+1} + Bz^{k+1} - c \right) \quad \text{dual variable update} \]

BP:

\[ \min_x f (x) + \|z\|_1 \]
\[ s.t. \quad x - z = 0 \]
Summary and Future Directions
Summary

1. Linear Inverse Problem
2. Compressive Sensing
3. Recovery Constraints such as: Spark, NSP, RIP, and Mutual Coherence.
4. Recovery Algorithms such as: MP, OMP, BP, BPDN, and SL0.
5. CS Challenges
6. CS Applications
7. CS Solvers


Thanks for attention.

http://display.sbu.ac.ir
http://students.sbu.ac.ir/h_safavi

Questions and Comments?

h_safavi@sbu.ac.ir, m_eslami@sbu.ac.ir

Sabalan Mountain, Ardabil, Iran