"Smoothed L0 (SL0): A fast and accurate algorithm for finding the sparse solution of an underdetermined system of linear equations"

Massoud BABAIE-ZADEH

October 2010

Outline

- Part I: Problem statement
- Part II: Applications
  - Signal Decomposition with overcomplete dictionaries
  - Blind Source Separation (BSS) and Sparse Component Analysis (SCA)
  - Compressed Sensing (CS)
  - Real-field coding
- Part III: Some ideas to find the sparsest solution
  - Direct method (minimum L0 norm method) $\rightarrow$ Computationally intractable
  - Matching Pursuit idea
  - Minimum L1 norm idea
- Part IV: Smoothed L0 (SL0) idea

Part I

Problem Statement
and
Uniqueness

Problem statement

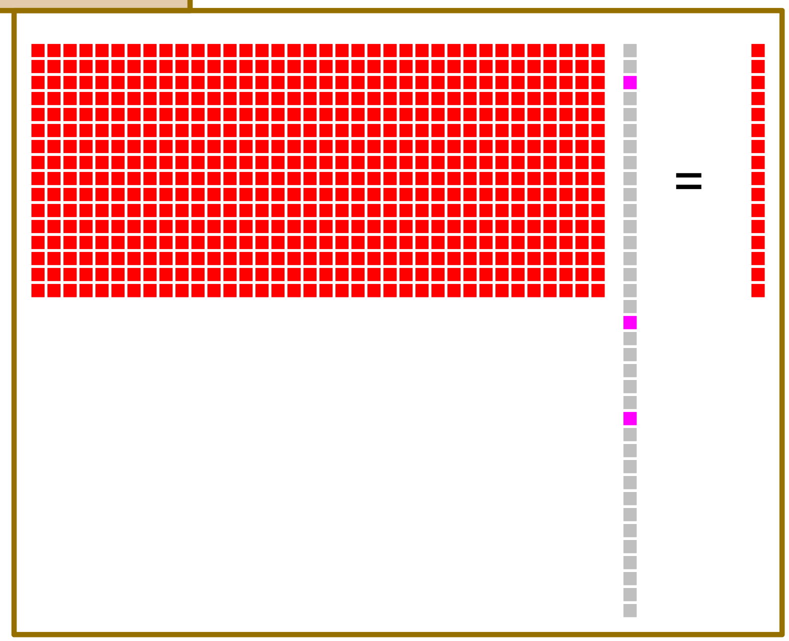
Underdetermined System of Linear equations (USLE):

$$\underbrace{\mathbf{A}}_{n \times m}\ \underbrace{\overset{?}{\mathbf{s}}}_{m \times 1} = \underbrace{\mathbf{x}}_{n \times 1}$$

n equations *m* unknowns *m*>*n*

$\Rightarrow$ Infinitely many solutions!

What is the sparsest solution?



Example (2 equations, 4 unknowns)

$$\begin{bmatrix} 1 & 2 & 1 & 1 \\ 1 & -1 & 2 & -2 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix} = \begin{bmatrix} 4 \\ -2 \end{bmatrix}$$

- Some of solutions:

$$\begin{bmatrix} 0 \\ 0 \\ 1.5 \\ 2.5 \end{bmatrix}, \begin{bmatrix} 5 \\ 1 \\ -3 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \\ -0.75 \\ 0.75 \end{bmatrix}, \underbrace{\begin{bmatrix} 0 \\ 2 \\ 0 \\ 0 \end{bmatrix}}_{\text{Sparsest}}, \begin{bmatrix} 2 \\ 0 \\ 0 \\ -2 \end{bmatrix}, \begin{bmatrix} 6 \\ 0 \\ -3 \\ 1 \end{bmatrix}$$

Issues

- Applications?
- Uniqueness? $\rightarrow$ Yes! $\Rightarrow$Useful
- How to find the sparsest solution?
- Stability (sensitivity to noise)

Uniqueness of the sparse solution

- **x**=**As**, n equations, m unknowns, m>n

- Theorem (Gorodnitsky & Rao 1997, Donoho 2004, Gribonval&Nielson2003, Donoho&Elad2003): if there is a solution **s** with less than or equal n/2 non-zero components, then it is unique under some mild conditions.

- Sparsity Revolution!

Part II

Examples

of applications of

Sparse solutions of USLE’s

Application 1:

Signal decomposition using overcomplete dictionaries

Signal Decomposition

- Decomposition of a signal x(t) as a linear combination of a set of known signals:

$$x(t) = \alpha_1 \varphi_1(t) + \cdots + \alpha_m \varphi_m(t)$$

- Examples:
  - Fourier Transform ($\varphi_i \rightarrow$ complex sinusoids)
  - Wavelet Transform
  - DCT
  - …

Signal Decomposition

- Decomposition of a signal x(t) as a linear combination of a set of known signals:

$$x(t) = \alpha_1 \varphi_1(t) + \cdots + \alpha_M \varphi_M(t)$$

- Terminology:
  - Atomic Decomposition (=Signal Decomposition)
  - Atoms $\rightarrow \varphi_i$
  - Dictionary $\rightarrow$ Set of all atoms: $\{\varphi_1, \varphi_2, \ldots\}$

Discrete Case

$$x(t) = \alpha_1 \varphi_1(t) + \cdots + \alpha_M \varphi_M(t), \qquad t = 1, \ldots, N$$

$$\begin{array}{c} \text{Time} \\ \\ \downarrow \end{array} \begin{bmatrix} x(1) \\ x(2) \\ x(3) \\ \vdots \\ x(N) \end{bmatrix} = \alpha_1 \begin{bmatrix} \varphi_1(1) \\ \varphi_1(2) \\ \varphi_1(3) \\ \vdots \\ \varphi_1(N) \end{bmatrix} + \cdots + \alpha_M \begin{bmatrix} \varphi_M(1) \\ \varphi_M(2) \\ \varphi_M(3) \\ \vdots \\ \varphi_M(N) \end{bmatrix}$$

$$\mathbf{x} = \alpha_1 \underline{\varphi}_1 + \cdots + \alpha_M \underline{\varphi}_M$$

Matrix form

$$
\begin{array}{c}
\text{Time} \\
\downarrow
\end{array}
\begin{bmatrix} x(1) \\ x(2) \\ x(3) \\ \vdots \\ x(N) \end{bmatrix}
= \alpha_1 \begin{bmatrix} \varphi_1(1) \\ \varphi_1(2) \\ \varphi_1(3) \\ \vdots \\ \varphi_1(N) \end{bmatrix}
+ \cdots + \alpha_M \begin{bmatrix} \varphi_M(1) \\ \varphi_M(2) \\ \varphi_M(3) \\ \vdots \\ \varphi_M(N) \end{bmatrix}
$$

$$
\mathbf{x} = \alpha_1 \underline{\varphi}_1 + \cdots + \alpha_M \underline{\varphi}_M
$$

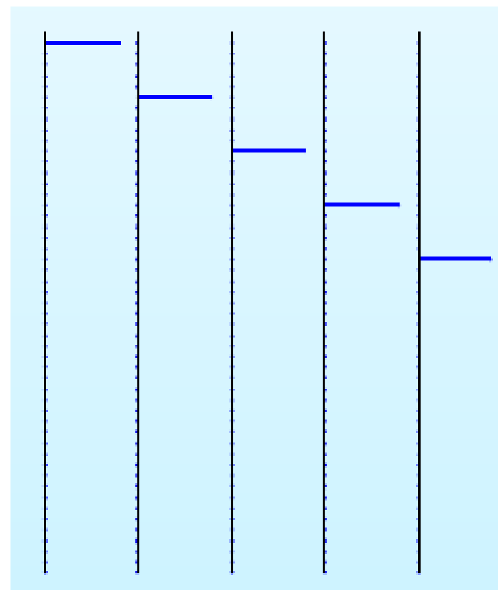
$$
\mathbf{x} = \begin{bmatrix} \varphi_1 & \cdots & \varphi_M \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_M \end{bmatrix} \rightarrow \boxed{\underbrace{\boldsymbol{\Phi}}_{N\times M}\, \underbrace{\boldsymbol{\alpha}}_{M\times 1} = \underbrace{\mathbf{x}}_{N\times 1}}
$$

Complete decomposition: M=N

$$\begin{array}{c} \\ \text{Time} \\ \\ \downarrow \\ \\ \end{array} \begin{bmatrix} x(1) \\ x(2) \\ x(3) \\ \vdots \\ x(N) \end{bmatrix} = \alpha_1 \begin{bmatrix} \varphi_1(1) \\ \varphi_1(2) \\ \varphi_1(3) \\ \vdots \\ \varphi_1(N) \end{bmatrix} + \cdots + \alpha_M \begin{bmatrix} \varphi_M(1) \\ \varphi_M(2) \\ \varphi_M(3) \\ \vdots \\ \varphi_M(N) \end{bmatrix}$$

$$\mathbf{x} = \alpha_1 \; \underline{\varphi}_1 + \cdots + \alpha_M \; \underline{\varphi}_M$$

- **M=N** $\rightarrow$ Complete dictionary $\rightarrow$ Unique set of coefficients
- Examples: Dirac dictionary, Fourier Dictionary



Dirac Dictionary:

$$\underline{\varphi}_k(n) = \begin{cases} 1 & n = k \\ 0 & n \neq k \end{cases}$$

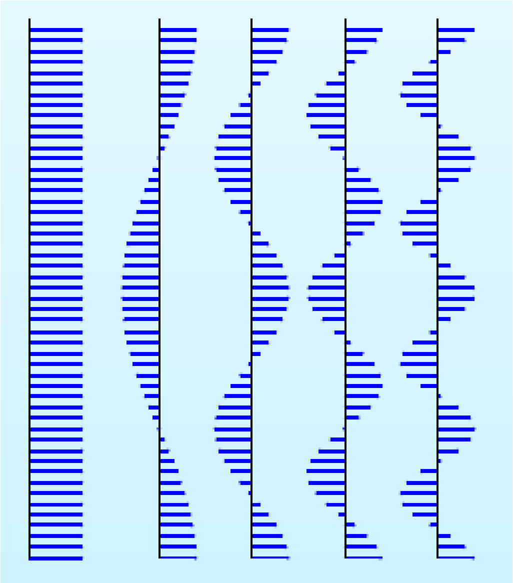
$$\Rightarrow \alpha_k = x(k)$$

Complete decomposition: M=N

$$\text{Time}\ \downarrow \begin{bmatrix} x(1) \\ x(2) \\ x(3) \\ \vdots \\ x(N) \end{bmatrix} = \alpha_1 \begin{bmatrix} \varphi_1(1) \\ \varphi_1(2) \\ \varphi_1(3) \\ \vdots \\ \varphi_1(N) \end{bmatrix} + \cdots + \alpha_M \begin{bmatrix} \varphi_M(1) \\ \varphi_M(2) \\ \varphi_M(3) \\ \vdots \\ \varphi_M(N) \end{bmatrix}$$

$$\mathbf{x} = \alpha_1 \underline{\varphi}_1 + \cdots + \alpha_M \underline{\varphi}_M$$

- **M=N** $\rightarrow$ Complete dictionary $\rightarrow$ Unique set of coefficients
- Examples: Dirac dictionary, Fourier Dictionary



Fourier Dictionary:

$$\underline{\varphi}_k = \left(1,\ e^{\frac{2k\pi}{N}},\ e^{\frac{2k\pi}{N}2}, \ldots,\ e^{\frac{2k\pi}{N}(N-1)}\right)^T$$

Over-complete decomposition: M>N

$$\begin{array}{c} \text{Time} \\ \downarrow \end{array} \begin{bmatrix} x(1) \\ x(2) \\ x(3) \\ \vdots \\ x(N) \end{bmatrix} = \alpha_1 \begin{bmatrix} \varphi_1(1) \\ \varphi_1(2) \\ \varphi_1(3) \\ \vdots \\ \varphi_1(N) \end{bmatrix} + \cdots + \alpha_M \begin{bmatrix} \varphi_M(1) \\ \varphi_M(2) \\ \varphi_M(3) \\ \vdots \\ \varphi_M(N) \end{bmatrix}$$

$$\mathbf{x} = \alpha_1 \underline{\varphi}_1 + \cdots + \alpha_M \underline{\varphi}_M$$

- M > N
- Over-complete dictionary
- Under-determined linear system: $\Phi\alpha=\mathbf{x}$
- Non-unique α

Overcomplete Sparse Decomposition: Motivation

$$\mathbf{x} = \alpha_1 \underline{\varphi}_1 + \cdots + \alpha_m \underline{\varphi}_m = \left[\underline{\varphi}_1, \ldots, \underline{\varphi}_m \right] \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_m \end{bmatrix} = \mathbf{\Phi}\,\boldsymbol{\alpha}$$

<u>Example:</u>

- A sinusoidal signal, $\sin(\omega_0 t)$, $\rightarrow$ Fourier Dictionary
- A signal with just one non-zero value, $\delta(t-t_0)$, $\rightarrow$ Dirac Dictionary
- How about the signal: $\sin(\omega_0 t)+\delta(t-t_0)$?

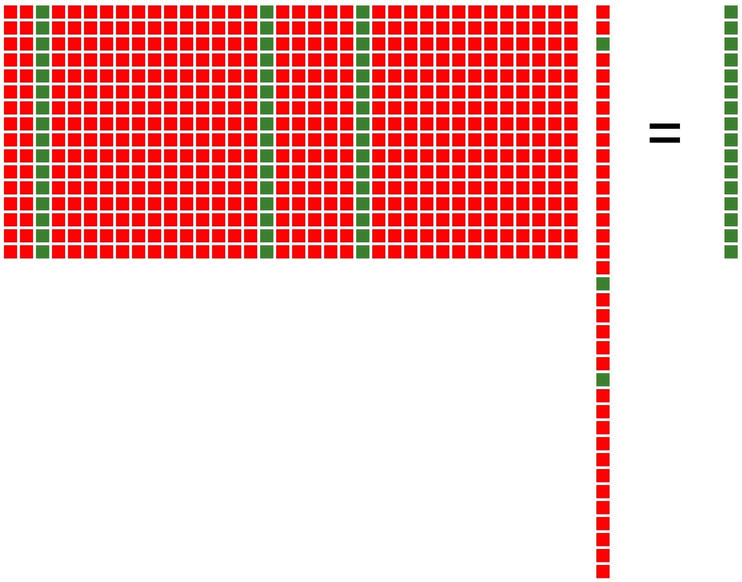
- A larger dictionary, containing both Dirac and Fourier atoms?
  $\rightarrow$ Non-unique α ☹

- Sparse solution of $\Phi\alpha=\mathbf{x}$

Overcomplete Sparse Decomposition

$$\mathbf{\Phi}\,\boldsymbol{\alpha} = \mathbf{x}$$

$$\alpha_1\varphi_1 + \cdots + \alpha_M\,\varphi_M = \mathbf{x}$$

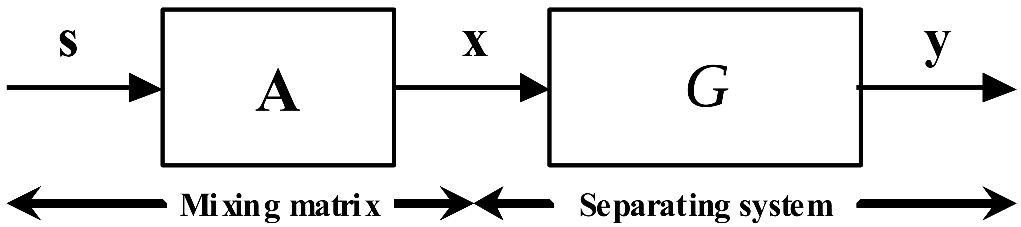


Application 2:

Blind Source Separation (BSS) and Sparse Component Analysis (SCA)

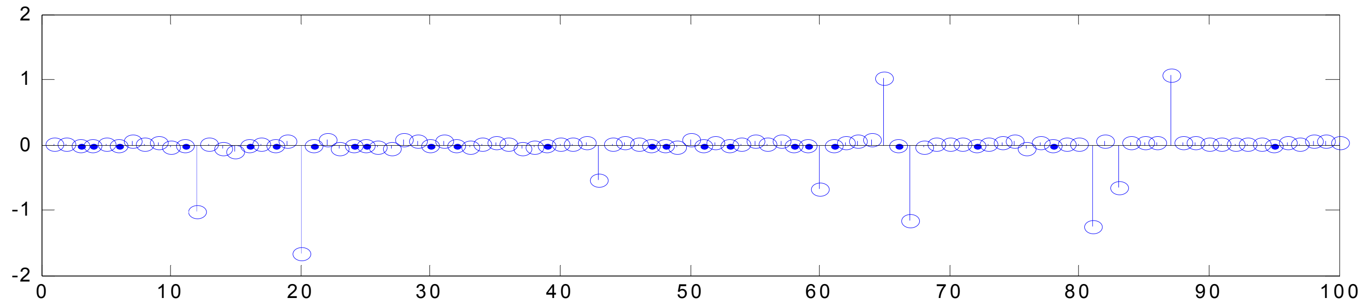
Blind Source Separation (BSS)

- Source signals $s_1, s_2, \ldots, s_M$
- Source vector: $\mathbf{s}=(s_1, s_2, \ldots, s_M)^T$
- Observation vector: $\mathbf{x}=(x_1, x_2, \ldots, x_N)^T$
- Mixing system $\rightarrow$ $\mathbf{x} = \mathbf{As}$



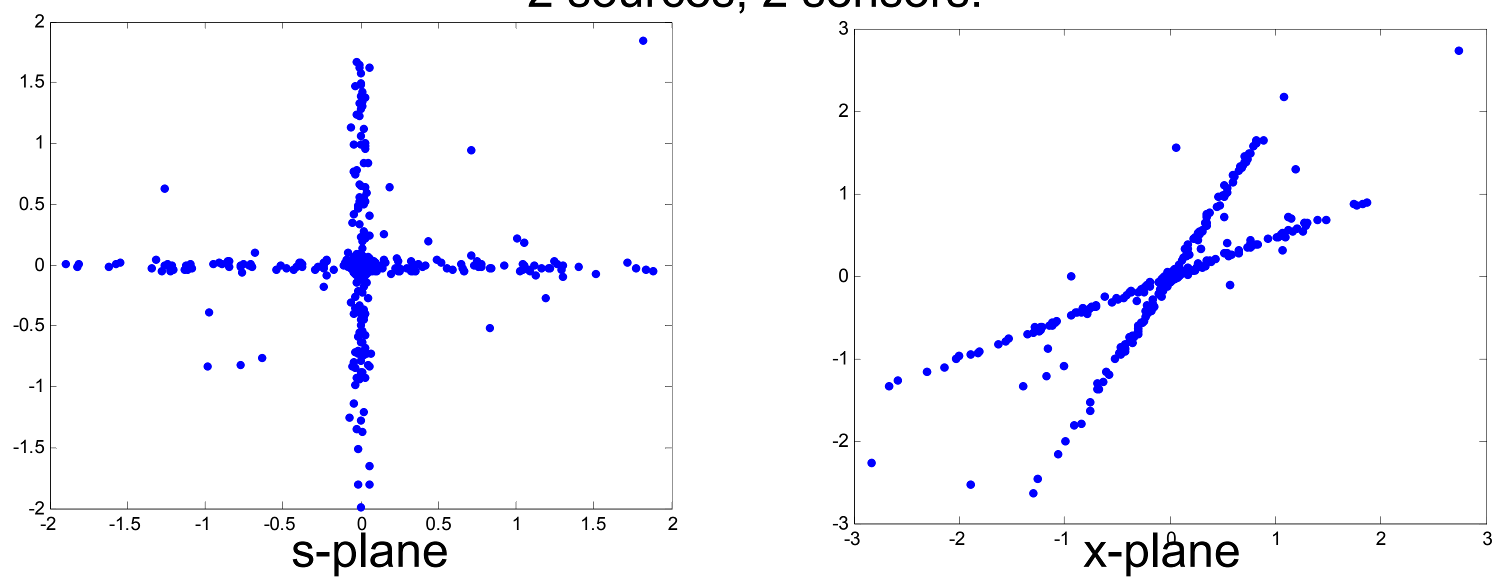
- Goal $\rightarrow$ Finding a separating system $\mathbf{y} = G(\mathbf{x})$

Sparse Sources



Note: The sources may be not sparse in time, but sparse in another domain (frequency, time-frequency, time-scale)

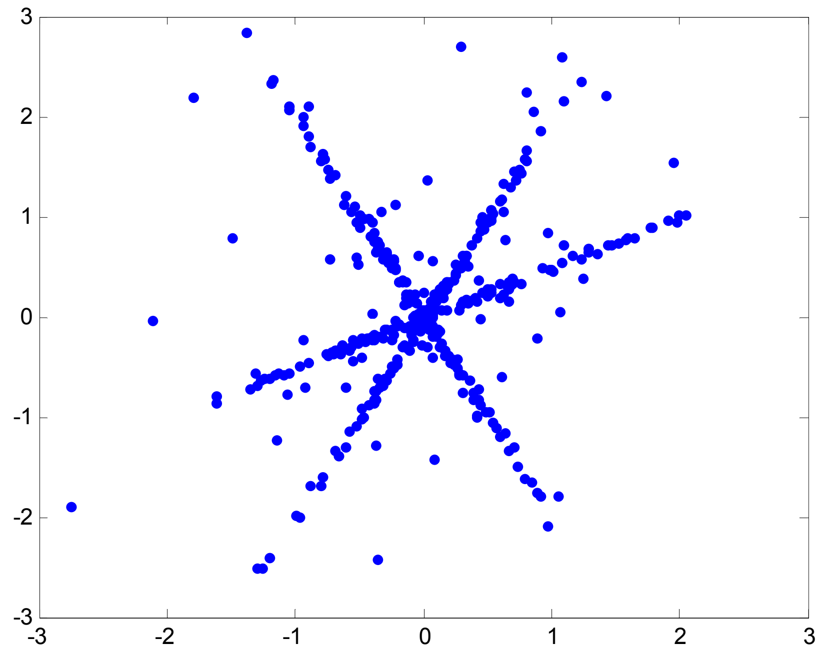
2 sources, 2 sensors:



Sparse sources (*cont.*)

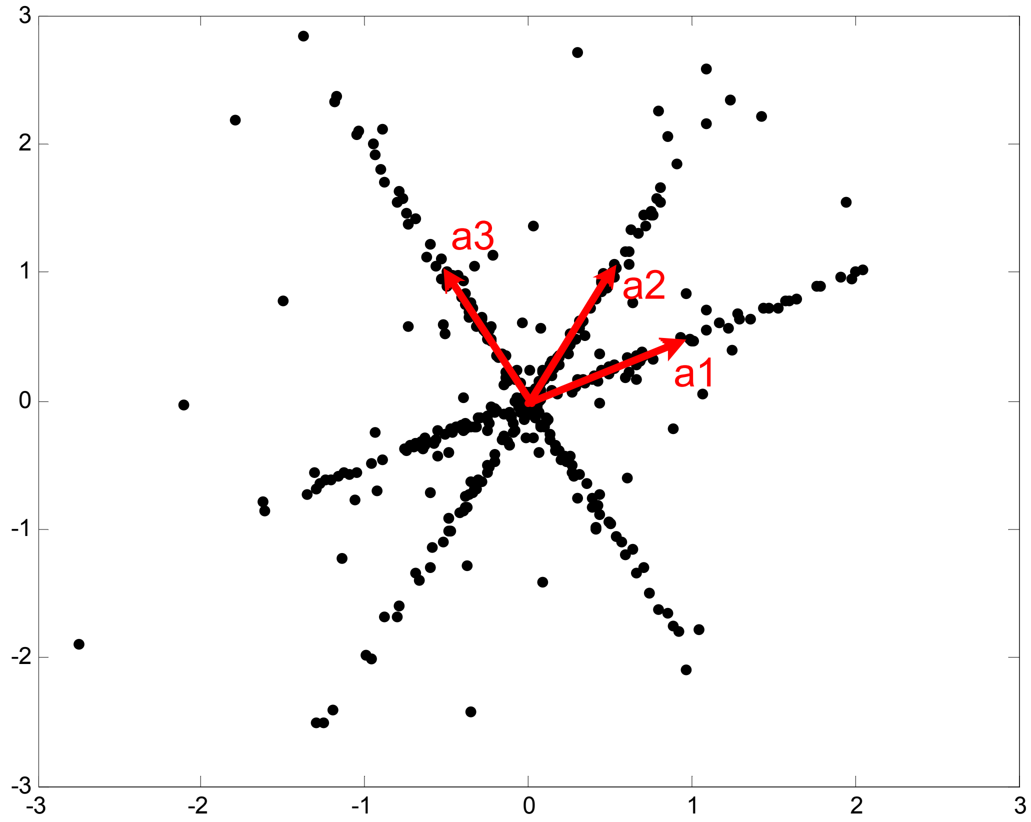
- 3 sparse sources, 2 sensors

Sparsity $\Rightarrow$ Source Separation, with more sensors than sources?



Estimating the mixing matrix

$\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3] \Rightarrow$



$\mathbf{x} = s_1\, \mathbf{a}_1 + s_2\, \mathbf{a}_2 + s_3\, \mathbf{a}_3$

$\Rightarrow$ Mixing matrix is easily identified for sparse sources

- Scale & Permutation indeterminacy
- $||\mathbf{a}_i||=1$

Restoration of the sources

- **A** known, how to find the sources?

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad or \quad \begin{cases} a_{11}s_1 + a_{12}s_2 + a_{13}s_3 = x_1 \\ a_{21}s_1 + a_{22}s_2 + a_{23}s_3 = x_2 \end{cases}$$

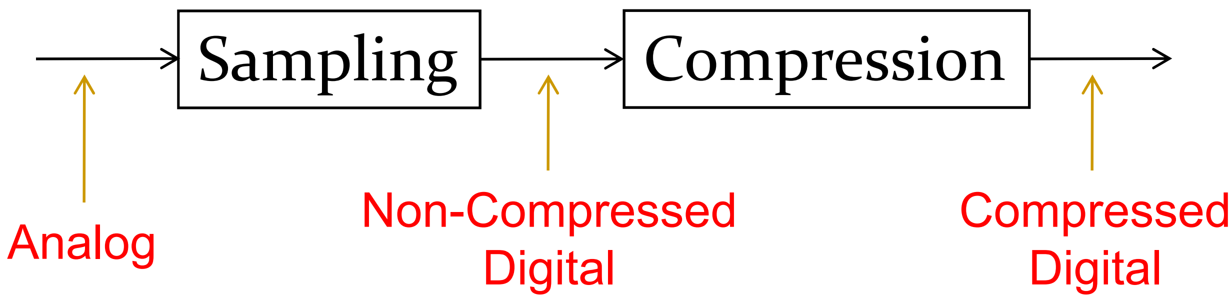
Underdertermined SCA

Application 3:

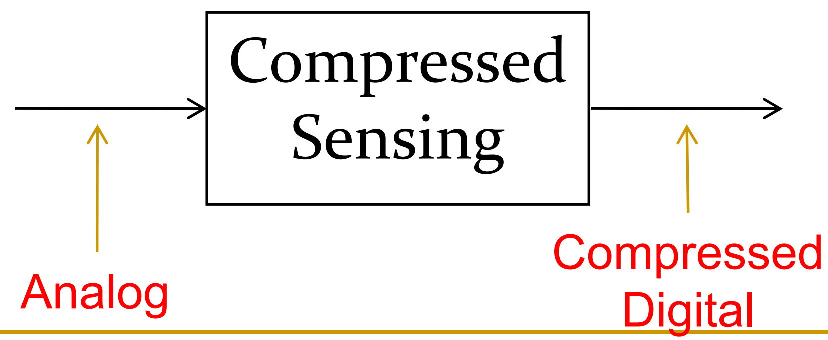
Compressed Sensing

Traditional Sampling vs. Compressed Sensing

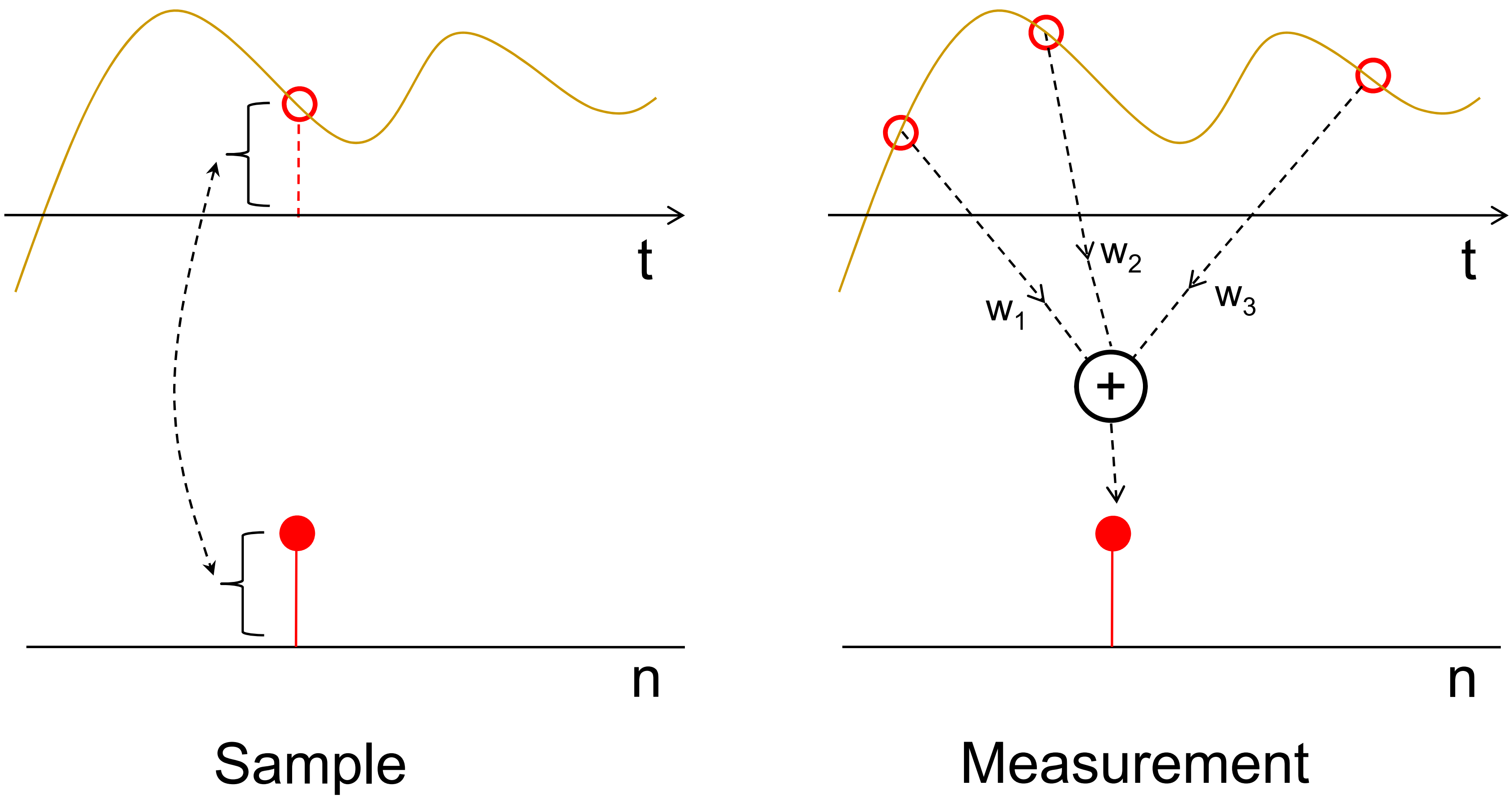
- Traditional Signal Acquisition:



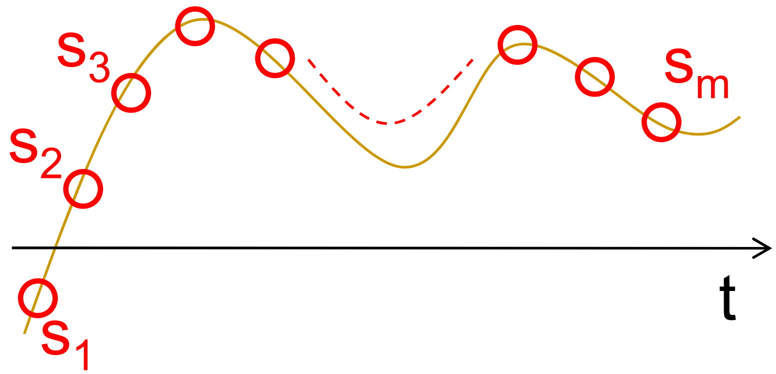
- Compressed Sensing (CS)



CS: Sample → Measurement



CS: A (smaller) set of random measurements



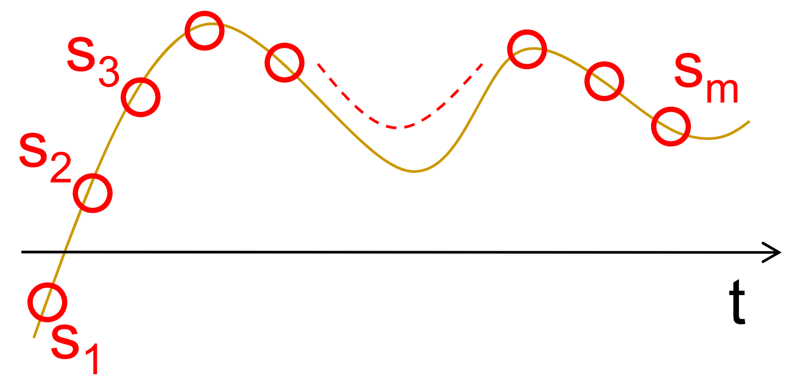
- 1^{st} measurement $\rightarrow$ $x_1 = \varphi_{11}\, s_1 + \varphi_{12}\, s_2 + \ldots + \varphi_{1n}\, s_m$
- 2^{nd} measurement $\rightarrow$ $x_2 = \varphi_{21}\, s_1 + \varphi_{22}\, s_2 + \ldots + \varphi_{2n}\, s_m$

:

- n^{th} measurement $\rightarrow$ $x_n = \varphi_{n1}\, s_1 + \varphi_{n2}\, s_2 + \ldots + \varphi_{nm}\, s_m$

$n < m \quad \Rightarrow$ USLE

CS: A (smaller) set of random measurements



$$\Phi \mathbf{s} = \mathbf{x}$$

Measurement matrix

?

Measurement vector

CS: A (smaller) set of random measurements

$$\Phi \mathbf{s} = \mathbf{x}$$

$$\downarrow$$

?

- $\Psi_{m \times m} \rightarrow$ sparsifying transform:

$$\mathbf{s} = \Psi\,\theta,$$

where θ is sparse

$$\Downarrow$$

$$(\Phi\,\Psi)\,\theta = \mathbf{x}$$

(USLE with sparsity)

Application 4:

Error Correcting Codes

(Real-field coding)

Coding Terminology

- **u**=(u_1,…, u_k) → the message to be sent (k symbols)
- **G** → Code Generator matrix ($n \times k$, $n>k$)
- **v** =(v_1,…, v_n) → Codeword:

$$\mathbf{v}=\mathbf{G}.\mathbf{u}$$

(adding n-k "parity" symbols)

- **H** → Parity check matrix ($(n-k)\times n$):

$$\mathbf{HG=0}$$

- **v** is a codeword if and only if: **H.v=0**

Error Correction

$$\mathbf{v} \longrightarrow \boxed{\text{Error-prone channel}} \longrightarrow \mathbf{r} = \mathbf{v} + \mathbf{e}$$

- **v** sent, **r = v + e** received

  (**e** is the error → assumed sparse)

- Syndrome of **r** → **s = H.r**

  ⇒ **s = H.(v+e) = H.e**

$$\underbrace{\mathbf{H}}_{(n-k)\times n}.\underbrace{\mathbf{e}}_{n} = \underbrace{\mathbf{s}}_{n-k}$$

→ **USLE**

Error Correction

The receiver:

- Receives **r** = **v**+**e**
- Computes **s** = **H**.**r**
- Finds sparse solution of USLE **H**.**e**=**s**
- $\Rightarrow$ Error Correction

Sparsity of **e**?

$$\underbrace{\mathbf{H}}_{(n-k)\times n}.\underbrace{\mathbf{e}}_{n} = \underbrace{\mathbf{s}}_{n-k}$$

- Galois fields (binary) codes $\Leftrightarrow$ small probability of error
- Real-field codes $\Leftrightarrow$ Impulsive noise, Laplace noise

Summary of Part II

Atomic Decomposition
on over-complete dictionaries

Underdetermined SCA

Compressed sensing

Real field
codes

Pattern recognition
and classification
(OCR)

Image Denoising

Image Compression

.
.
.

Finding sparsest solution of
an USLE

Part III

HOW

to find the

Sparsest Solution?

How to find the sparsest solution

- **A.s** = **x**, n equations, m unknowns, m>n
- Goal: Finding the sparsest solution
- Note: at least m-n unknown are zero.

- Direct method:
  - Set m-n (arbitrary) unknowns equal to zero
  - Solve the remaining system of n equations and n unknowns
  - Do above for all possible choices, and take the sparsest answer.

- Another name: Minimum L^0 norm method
  - L^0 norm of s = number of non-zero components = $\Sigma|s_i|^0$

Example

$$\begin{bmatrix} 1 & 2 & 1 & 1 \\ 1 & -1 & 2 & -2 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix} = \begin{bmatrix} 4 \\ -2 \end{bmatrix}$$

$$\binom{4}{2} = 6 \text{ different answers to be tested}$$

- s1=s2=0 $\Rightarrow$ **s**=(0, 0, 1.5, 2.5)^T $\Rightarrow$ L^0=2
- s1=s3=0 $\Rightarrow$ **s**=(0, 2, 0, 0)^T $\Rightarrow$ L^0=1
- s1=s4=0 $\Rightarrow$ **s**=(0, 2, 0, 0)^T $\Rightarrow$ L^0=1
- s2=s3=0 $\Rightarrow$ **s**=(2, 0, 0, 2)^T $\Rightarrow$ L^0=2
- s2=s4=0 $\Rightarrow$ **s**=(10, 0, -6, 0)^T $\Rightarrow$ L^0=2
- s3=s4=0 $\Rightarrow$ **s**=(0, 2, 0, 0)^T $\Rightarrow$ L^0=2
- $\Rightarrow$ Minimum L^0 norm solution $\rightarrow$ **s**=(0, 2, 0, 0)^T

Drawbacks of minimal norm L^0

$$(P_0)\ \text{Minimize } \|\mathbf{s}\|_0 = \sum_i |s_i|^0 \quad \text{s.t. } \mathbf{x} = \mathbf{A}\mathbf{s}$$

- Highly (unacceptably) sensitive to noise
- Need for a combinatorial search:

$$\binom{m}{n} \text{ diffetent cases should be tested separately}$$

- Example. m=50, n=30,

$$\binom{50}{30} \approx 5\times 10^{13} \text{ cases should be tested.}$$

On our computer: Time for solving a 30 by 30 system of equation=$2\text{x}10^{-4}$

Total time $\approx (5\text{x}10^{13})(2\text{x}10^{-4}) \approx$ 300 years! $\rightarrow$ Non-tractable

Some ideas for solving the problem

- Method of Frames (MoF) [Daubechies, 1989]
- Matching Pursuit [Mallat & Zhang, 1993]
- Basis Pursuit (minimal L1 norm $\rightarrow$ Linear Programming) [Chen, Donoho, Saunders, 1995]
- SL0

Idea 1 (obsolete):

Pseudo-inverse

[Daubechies, 1989]

Method of Frames (Daubechies, 1989)

- Use pseudo-inverse:

$$\hat{\mathbf{s}}_{MoF} = \mathbf{A}^T \left(\mathbf{A}\mathbf{A}^T \right)^{-1} \mathbf{x}$$

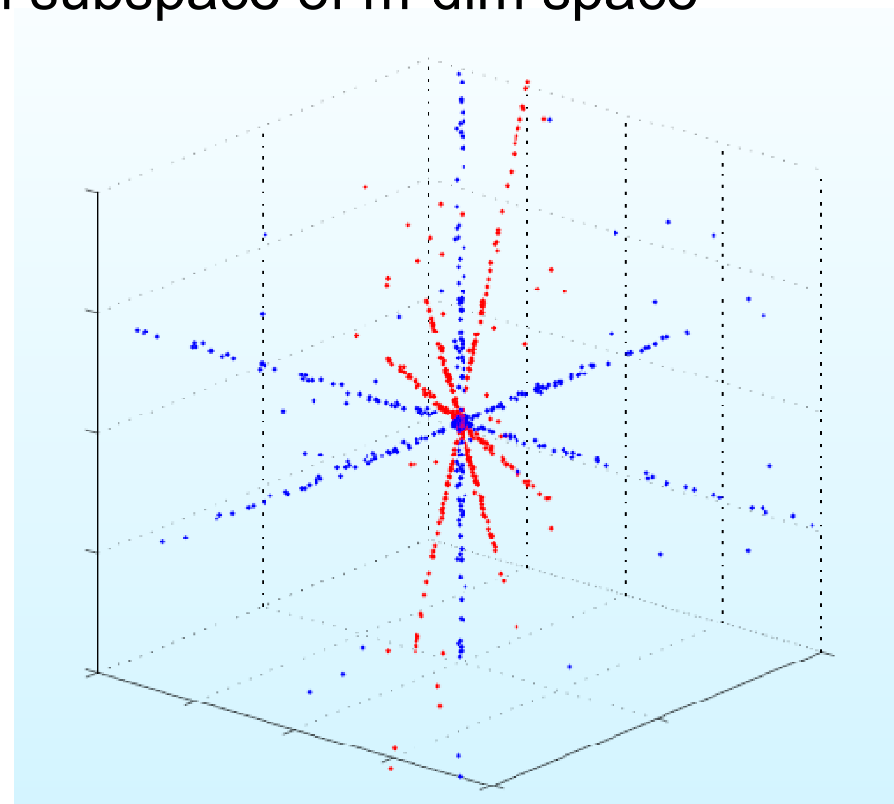
- It is equivalent to minimizing the L2 (energy) solution:

$$(P_2) \;\; \text{Minimize} \; \left\| \mathbf{s} \right\|_2 = \sum_i \left| s_i \right|^2 \quad \text{s.t.} \;\; \mathbf{x} = \mathbf{A}\mathbf{s}$$

- Different view points resulting in the same answer:
  - Linear LS inverse $\quad \hat{\mathbf{s}} = \mathbf{B}\mathbf{x}, \;\; \mathbf{B}\mathbf{A} \overset{LS}{\approx} \mathbf{I}$
  - Linear MMSE Estimator
  - MAP estimator under a Gaussian prior $\quad \mathbf{s} \sim N\left(0, \sigma_s^2 \mathbf{I}\right)$

Drawback of MoF

- It is a 'linear' method: **s=Bx**
  - $\Rightarrow$ s will be an n-dim subspace of m-dim space
- Example:
  3 sources, 2 sensors:



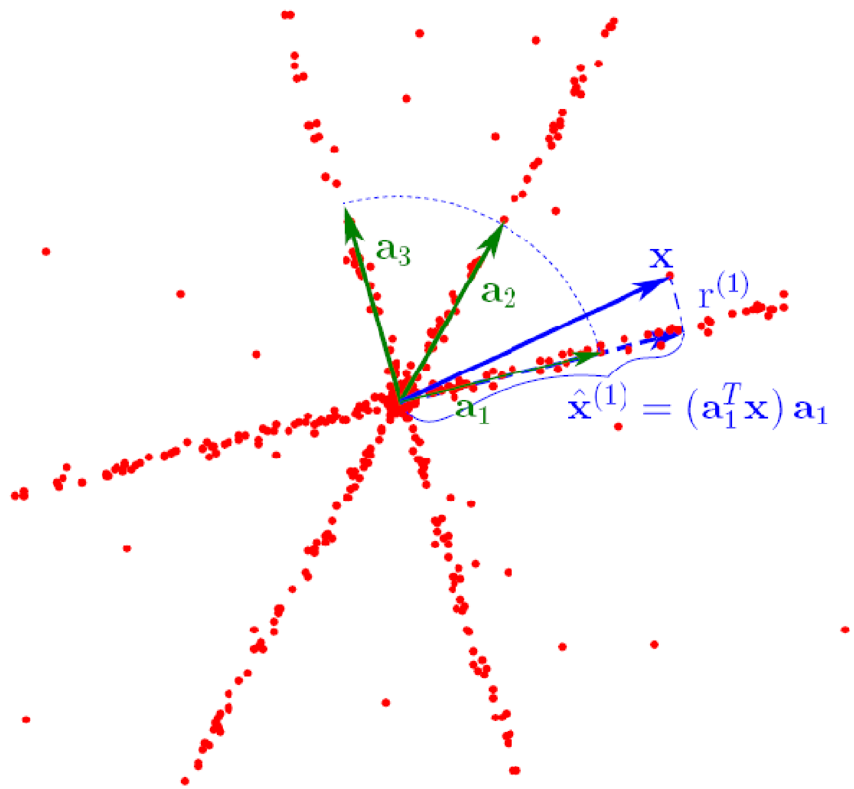
- $\Rightarrow$ Never can produce original sources

Idea 2:

Matching Pursuit

[Mallat & Zhang, 1993]

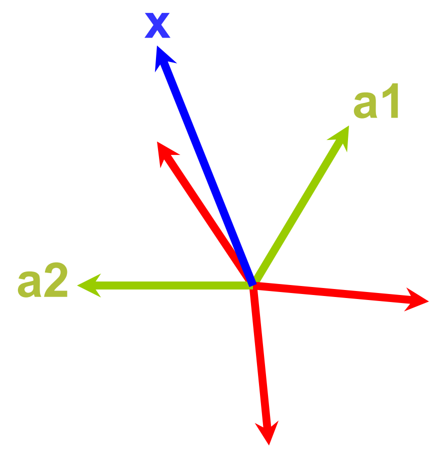
Matching Pursuit (MP) [Mallat & Zhang, 1993]



Properties of MP

- Advantage:
  - Very Fast

- Drawback
  - A very 'greedy' algorithm → Error in a stage, can never be corrected → Not necessarily a sparse solution



Variants

- OMP: Orthogonal MP [Tropp&Gilbert, IEEE Tr. On IT, 2007]
- StOMP: Stagewise MP [Donoho et. al., TechReport, 2006]
- CoSaMP: Compressive Sampling Matching Pursuit [Needell&Tropp, *Appl. Comp. Harmonic Anal., 2008*]
- …

Idea 3:

Minimizing L1 norm

[Chen, Donoho, Saunders, 1995]

Minimum L^1 norm or Basis Pursuit [Chen, Donoho, Saunders, 1995]

- Minimum norm L1 solution:

$$(P_1) \ \ \text{Minimize } \|\mathbf{s}\|_1 = \sum_i |s_i| \quad \text{s.t. } \ \mathbf{x} = \mathbf{A}\mathbf{s}$$

- MAP estimator under a Laplacian prior

Minimal L^1 norm (*cont.*)

$$(P_1) \ \text{Minimize } \|\mathbf{s}\|_1 = \sum_i |s_i| \quad \text{s.t. } \mathbf{x} = \mathbf{A}\mathbf{s}$$

- Minimal L^1 norm solution may be found by Linear Programming (LP)
- Fast algorithms for LP:
  - Simplex
  - Interior Point method
- A theoretical guarantee for finding the sparse solution, under some limiting conditions

Theoretical Support for BP: Mutual Coherence

- **Mutual Coherence** [Gribonval&Nielsen2003, Donoho&Elad2003]: of the matrix **A** is the maximum correlation between its columns

$$M = \max_{i \neq j} < \mathbf{a}_i, \mathbf{a}_j > = \max_{i \neq j} \ \mathbf{a}_i^T \mathbf{a}_j$$

- For an **A** (n $\times$ m) with normalized columns:

$$M \geq \frac{1}{\sqrt{n}}$$

Theoretical Support for BP: Theorem

- Theorem [Gribonval&Nielsen2003, Donoho&Elad2003]:
  If the USLE **As**=**x** has a sparse solution **s** such that

$$\|\mathbf{s}\|_0 < \frac{1+M^{-1}}{2}$$

  then it is guaranteed that BP finds this solution.

- Loosely speaking: BP is guaranteed to work were there is a "very very" sparse solution.

Example

- m=1000 unknowns, n=500 equations

- Uniqueness: a sparse solution with at most $||s||_0 \leq n/2=250$ is the unique sparsest solution.
- BP: $M^{-1} <$ sqrt(500)=22.36 $\Rightarrow (1+M^{-1})/2<11.68$

- So:
  - If there is a sparse solution with 250 out of 1000 non-zero entries, it is the unique sparse solution.
  - If there is a sparse solution with 11 out of 1000 non-zero entries, it is guaranteed that it can be found by BP.

Summary of minimal L^1 norm method

- Advantages:
  - Good practical results
  - Existence of a theoretical support
- Drawbacks:
  - Theoretical support is limited to very sparse solutions
  - Tractable, but still very time-consuming

Part IV

Smoothed L0 (SL0) Approach

References

- Developed mainly in 2006 by:
  - Hossein Mohimani,
  - Massoud Babaie-Zadeh,
  - Christian Jutten

- Papers on SL0:
  - Conference ICA2007 (London).
  - Journal: IEEE Transactions on Signal Processing, January 2009 (>50 citations till now).
  - Complex-valued version: ICASSP2008.
  - Convergence analysis: arXiv (co-authored with I.Gorodnitsky).

- Extentions
  - Robust-SL0 [Eftekhari et.al., ICASSP 2009]
  - Two-dimensional signals [Ghaffari et. al., ICASSP2009],[Eftekhari et. al., Signal Processing, accepted]

Smoothed L0 Norm: The main idea

$$(P_0) \ \ \text{Minimize } \|\mathbf{s}\|_0 = \sum_i |s_i|^0 \quad \text{s.t. } \ \mathbf{x} = \mathbf{A}\mathbf{s}$$

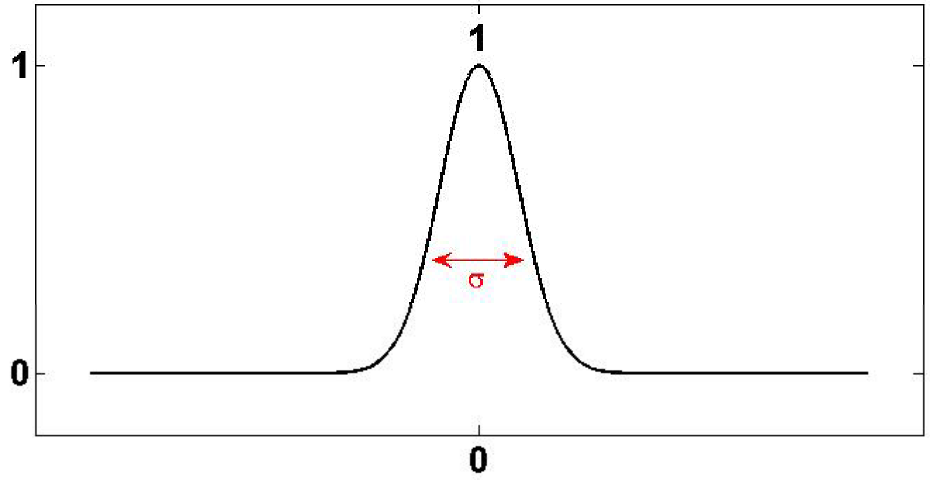
- Note: Problems of the L0 norm:
  - Computational load (combinatorial search)
  - Sensitivity to noise
- Both due to discontinuity of the L0 norm
- Main Idea: Use a smoothed L0 norm (continuous)

Smoothed L0 (SL0): Smoothing function

$$f_\sigma(s) \triangleq \exp\left(-s^2/2\sigma^2\right),$$

$$\Rightarrow \qquad \lim_{\sigma\to 0} f_\sigma(s) = \begin{cases} 1 & ;\text{if } s = 0 \\ 0 & ;\text{if } s \neq 0 \end{cases}$$

$F_\sigma(\mathbf{s})$



SL0: Finding the sparse solution

- Goal: For a small σ

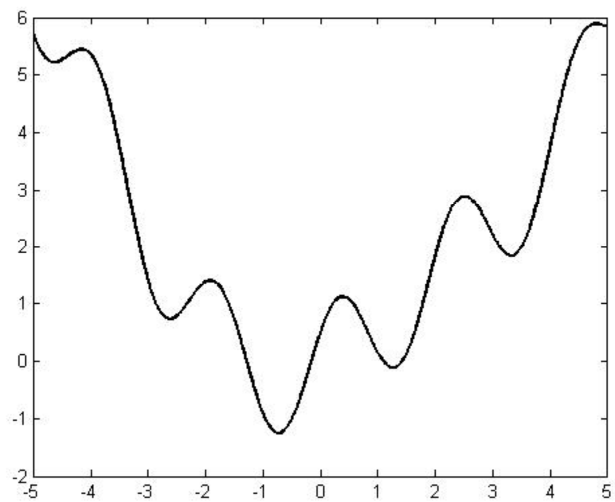
  Maximize $F_\sigma(s)$ s.t. **As=x**

- Problem: Small σ $\rightarrow$ lots of local maxima

- Idea: Use Graduated Non-Convexity (GNC)

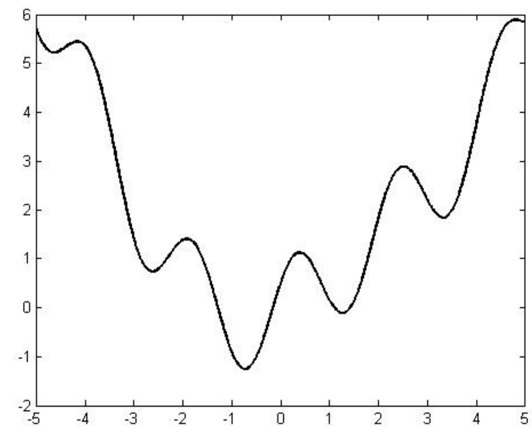
Graduated Non-Convexity (GNC)

- Global minimization of a non-convex $f(\cdot)$

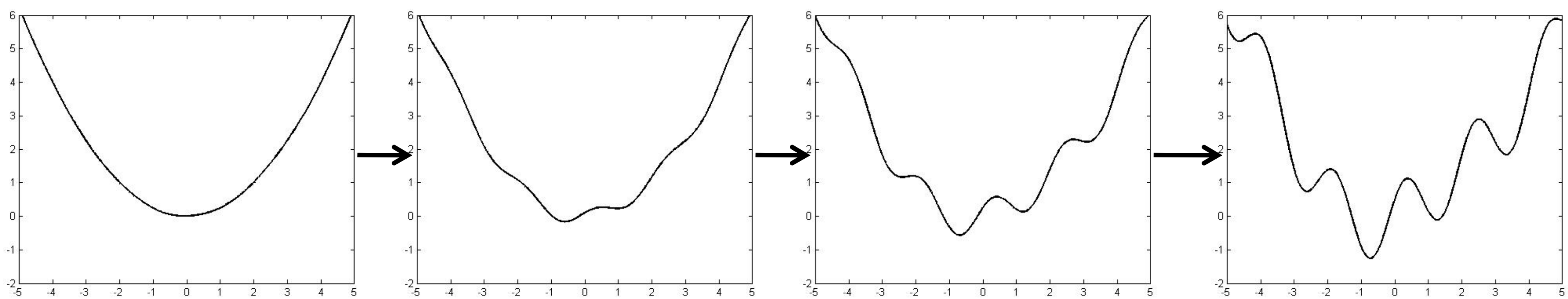


GNC: Example

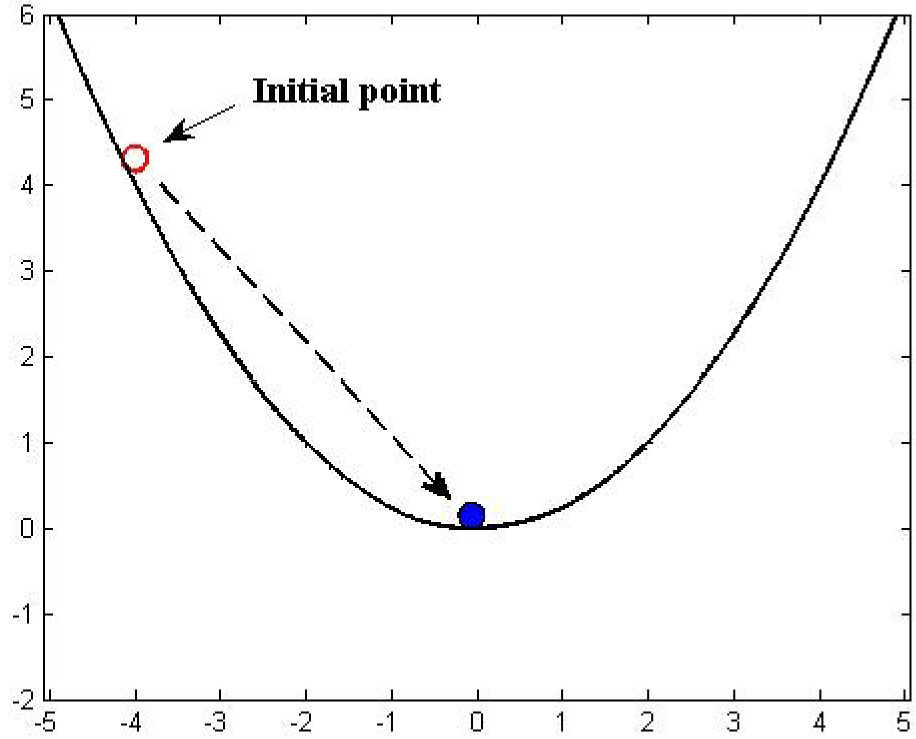
The function to be minimized
(many local minima)



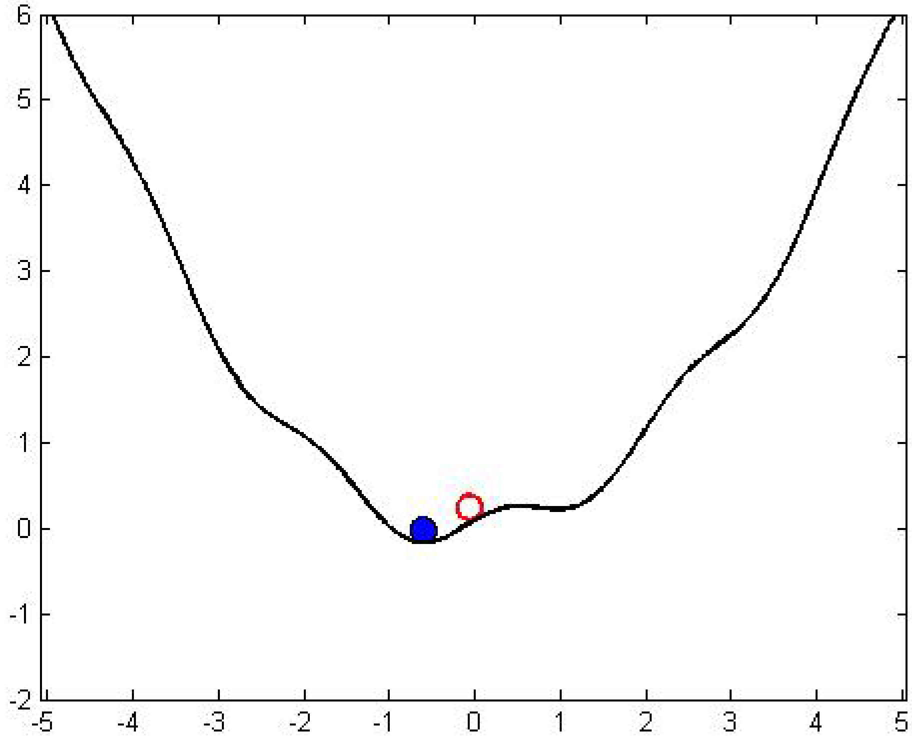
Sequence of functions converging to the original function:



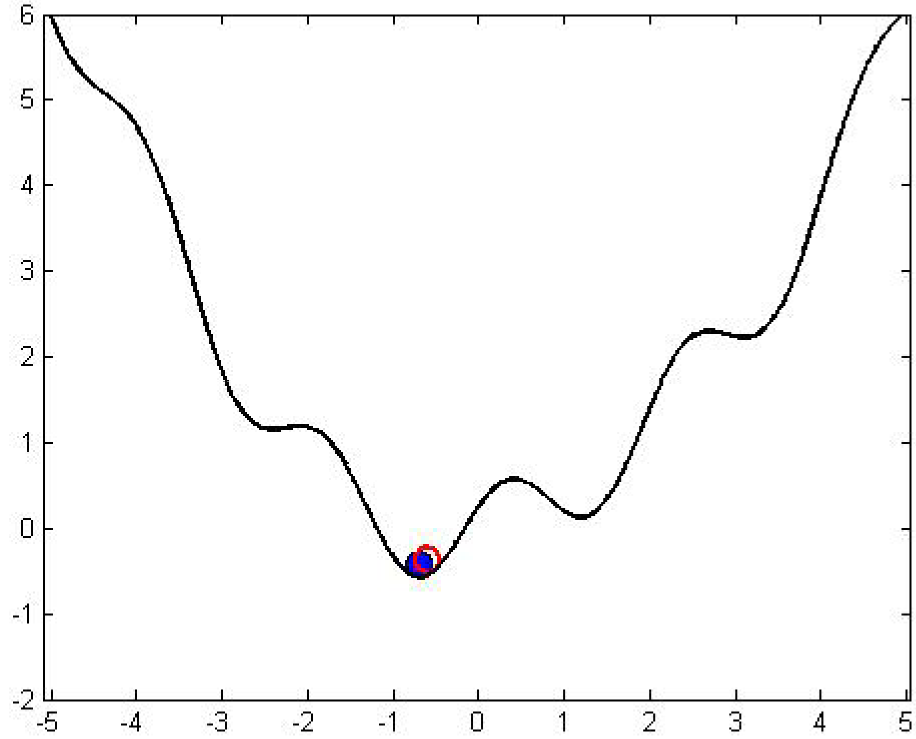
GNC: Example (*cont.*)



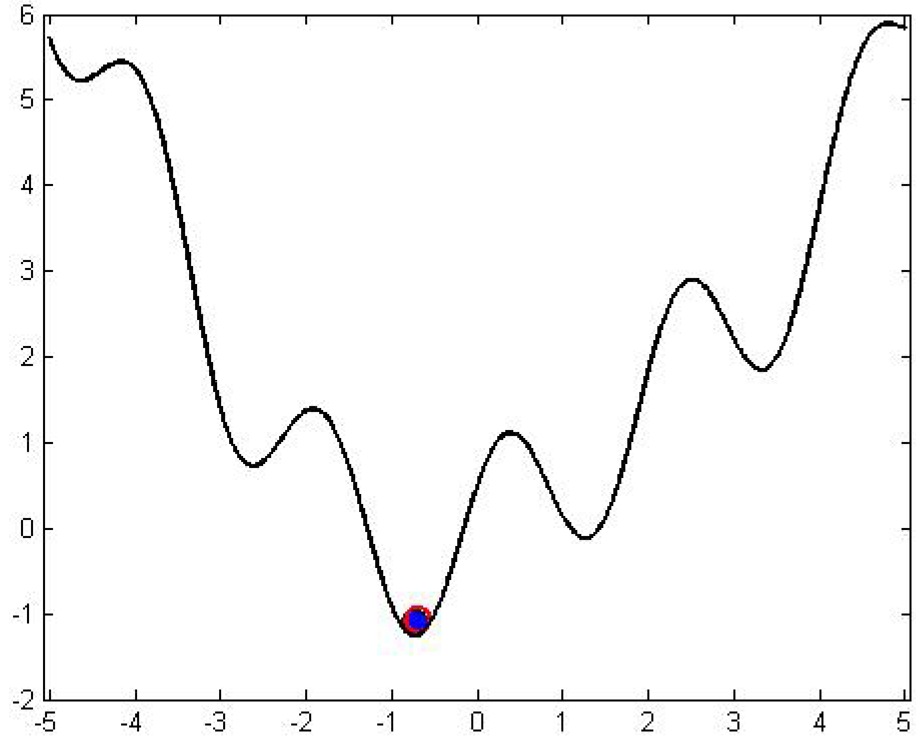
GNC: Example (*cont.*)



GNC: Example (*cont.*)



GNC: Example (*cont.*)



GNC

- Global minimization of a non-convex $f(\cdot)$
  - Use a sequence of functions $f_\sigma(\cdot)$, $\sigma = \overrightarrow{\sigma_1, \sigma_2, \sigma_3, \ldots}^{\text{decreasing}}$, converging to $f(\cdot)$:

$$\lim_{\sigma \to 0} f_\sigma(\cdot) = f(\cdot)$$

  - For each σ, minimize $f_\sigma(\cdot)$, by starting the search from the minimizer for the previous σ

SL0:

- Goal: For a small σ

  Maximize $F_\sigma(s)$ s.t. $\mathbf{As=x}$

- Use the GNC idea:
  - Start with large σ, and decrease it gradually.
  - For each σ, maximize $F_\sigma(s)$ by starting the search from the maximizer of the previous $F_\sigma(s)$ (which had a larger σ).

- Starting point? (corresponding to $\sigma\to\infty$)?

Initialization

- Theorem: For very large σ:

  $$\text{Maximize } F_\sigma(\mathrm{s}) \quad \text{s.t.} \quad \mathbf{As=x}$$

  has no local maxima, and its unique solution is the minimum L2 norm solution of **As=x** (given by pseudo-inverse)

- $\Rightarrow$ $\boxed{\text{starting point of SL0: min L2 norm solution}}$

Constraints?

- Goal: For a small σ

  Maximize $F_\sigma(s)$ s.t. **As=x**

- Use a Gradient-Projection approach.

- Each iteration:
  - Gradient: $\mathbf{s} \leftarrow \mathbf{s} + \mu_\sigma \nabla F_\sigma(\mathbf{s})$
  - Projection onto $\{\mathbf{s}|\mathbf{As}=\mathbf{x}\}$

- Decreasing step-size: $\mu_\sigma = \mu_0\, \sigma^2$

Final Algorithm

- Initialization: Set $\hat{\mathbf{s}}_0 = \mathbf{A}^\dagger \mathbf{x}$. Choose a suitable decreasing sequence for σ: $[\sigma_1 \ldots \sigma_J]$.
- For $j = 1, \ldots, J$:
  1) Let $\sigma = \sigma_i$.
  2) Maximize $F_\sigma(\mathbf{s})$ subject to $\mathbf{A}\mathbf{s} = \mathbf{x}$, using L iterations of steepest ascent:
     - Initialization: $\mathbf{s} = \hat{\mathbf{s}}_{j-1}$.
     - For $\ell = 1, 2, \ldots, L$
       a) Let $\mathbf{s} \leftarrow \mathbf{s} + (\mu\sigma^2)\nabla F_\sigma(\mathbf{s})$.
       b) Project $\mathbf{s}$ back onto the feasible set $\{\mathbf{s} | \mathbf{A}\mathbf{s} = \mathbf{x}\}$:
       $$\mathbf{s} \leftarrow \mathbf{s} - \mathbf{A}^\dagger(\mathbf{A}\mathbf{s} - \mathbf{x}).$$
  3) Set $\hat{\mathbf{s}}_j = \mathbf{s}$.
- Final answer is $\hat{\mathbf{s}} = \hat{\mathbf{s}}_J$.

Simulation result

$$\mathbf{A}\,\mathbf{s} = \mathbf{x}$$

$n \times m$

$m \times 1$

$n \times 1$

n equations
m unknowns
m>*n*

- m = 1000
- n = 400
- About 100 non-zero entries in **s**

Experimental Result (*cont.*)

TABLE I

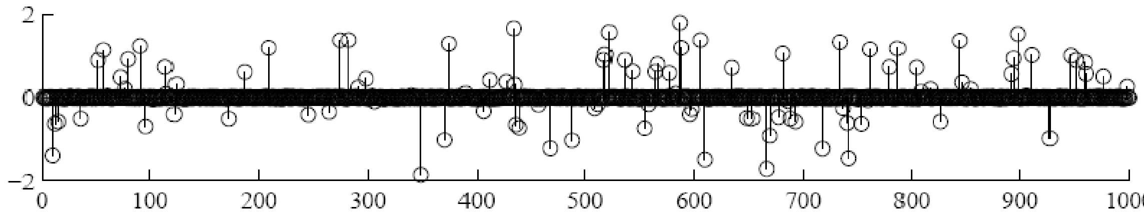
PROGRESS OF SL0 FOR A PROBLEM WITH $m = 1000$, $n = 400$ AND $k = 100$ $(p = 0.1)$.

itr. #	σ	MSE	SNR (dB)
1	1	$4.84\,e-2$	2.82
2	0.5	$2.02\,e-2$	5.19
3	0.2	$4.96\,e-3$	11.59
4	0.1	$2.30\,e-3$	16.44
5	0.05	$5.83\,e-4$	20.69
6	0.02	$1.17\,e-4$	28.62
7	0.01	$5.53\,e-5$	30.85

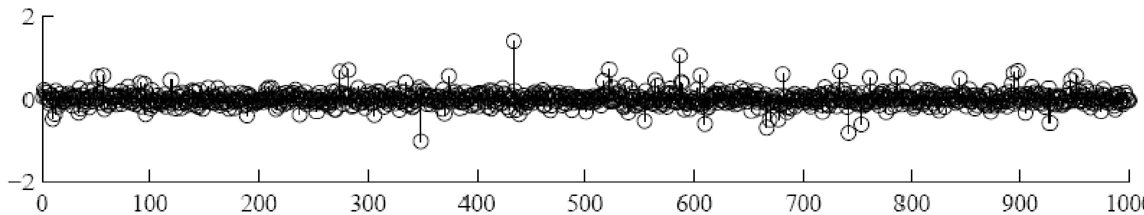
algorithm	total time	MSE	SNR (dB)
SL0	0.227 seconds	$5.53\,e-5$	30.85
LP	30.1 seconds	$2.31\,e-4$	25.65

Experimental Result (*cont.*)

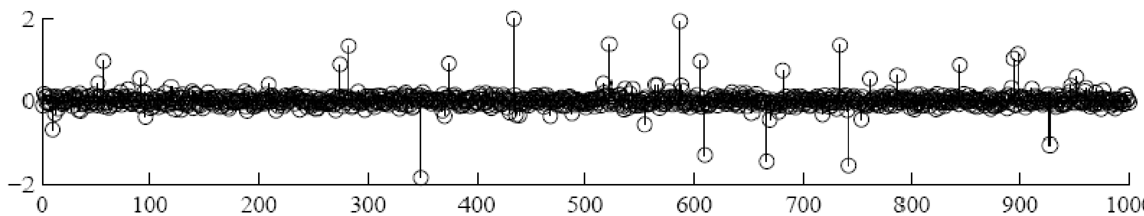
Original:



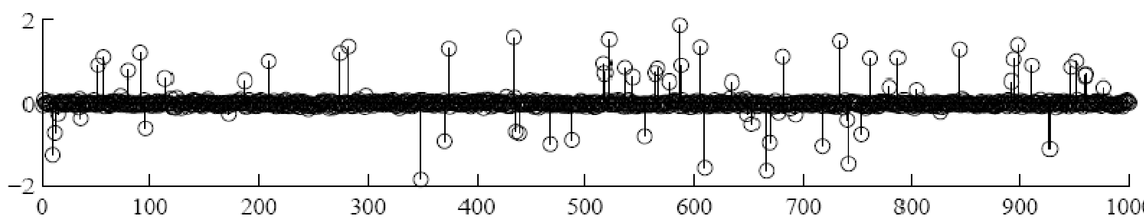
σ=1



σ=0.5



σ=0.2



Comparisons

- SL0 versus L0:
  - No need for combinatorial search (Fast)
  - Not sensitive to noise (Accurate)

- SL0 versus L1:
  - ☺ Highly faster
  - ☺ Better accuracy
  - ☹ Non-convex (need for gradual decreasing σ)

Conclusions

- L0 intractable and sensitive to noise? Use its smoothed version!
- $\Rightarrow$ A highly faster algorithm compared to L1 minimization approach.
- Try it yourself!

  http://ee.sharif.edu/~SLzero or google "SL0 algorithm".

Conclusions (*cont.*)

- We have used it in many applications, including:
  - Two dimensional compressive classifiers (ICIP2009)
  - Two dimensional random projections (to appear in Signal Processing)
  - Image inpainting (MLSP2009)
  - Image denoising (MLSP2009)
  - Image compression (ICA2009)
  - Dictionary learning (ICASSP2009)
  - …
- Not yet enough fast to solve n=8000, m=200000

Thank you very much for your attention