Inverse problems and sparse models (3/6)

Rémi Gribonval
INRIA Rennes - Bretagne Atlantique, France
remi.gribonval@inria.fr

Reminder of last sessions

- **Introduction**
  - ✓ sparsity & data compression
  - ✓ inverse problems in signal and image processing
    - ✦ image deblurring, image inpainting,
    - ✦ channel equalization, signal separation,
    - ✦ tomography, MRI
  - ✓ sparsity & under-determined inverse problems
    - ✦ relation to subset selection problem
- **Pursuit Algorithms**
  - ✓ Greedy algorithms: Matching Pursuit & al
  - ✓ L1 minimization principles
  - ✓ L1 minimization algorithms
  - ✓ Complexity of Pursuit Algorithms

Sparsity: definition

- A vector is
  - ✓ **sparse** if it has (many) zero coefficients
  - ✓ ***k*-sparse** if it has *at most k* nonzero coefficients
- Symbolic representation as column vector
- **Support** = indices of nonzero components
- Sparsity measured with **L0 pseudo-norm**

$$\|x\|_0 := \sharp\{n,\ x_n \neq 0\} = \sum_n |x_n|^0$$

Convention here
$a^0 = 1(a > 0); 0^0 = 0$

- *In french:*
  - ✦ sparse -> *«creux», «parcimonieux»*
  - ✦ sparsity, sparseness -> *«parcimonie»,* ~~*«sparsité»*~~

Linear inverse problems: definition

- **Definition**: a problem where a high-dimensional vector must be estimated from its low dimensional projection
- **Generic form:**

$$\mathbf{b} = \mathbf{A}\mathbf{y} + \mathbf{e}$$

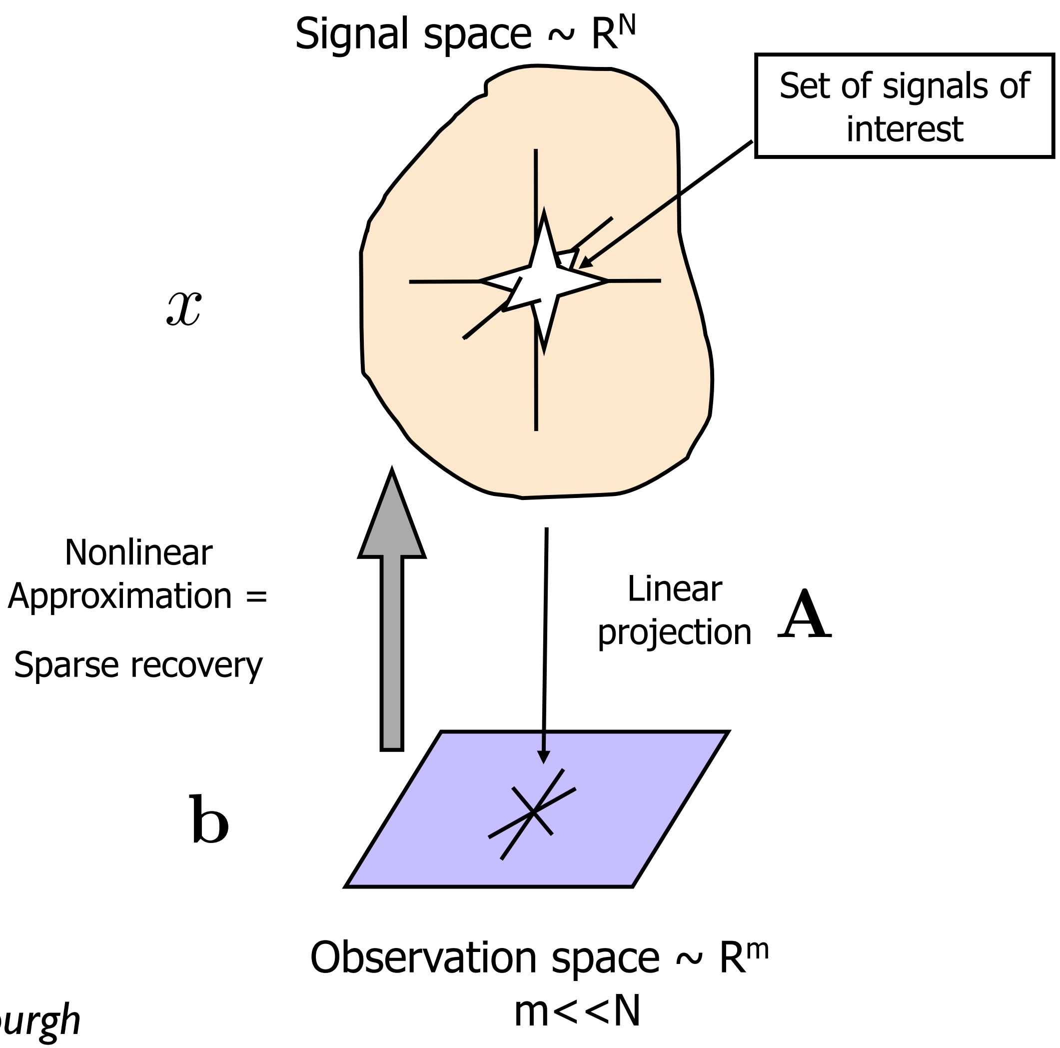
observation/measure unknown noise

projection matrix

✓ m observations / measures $\mathbf{b} \in \mathbb{R}^m$ $\mathbf{A} \in \mathbb{R}^{m \times N}$

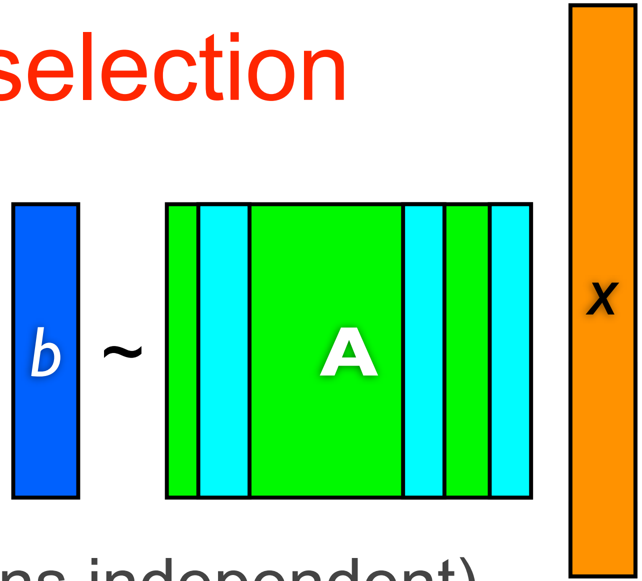
✓ N unknowns $\mathbf{y} \in \mathbb{R}^N$

Inverse problems

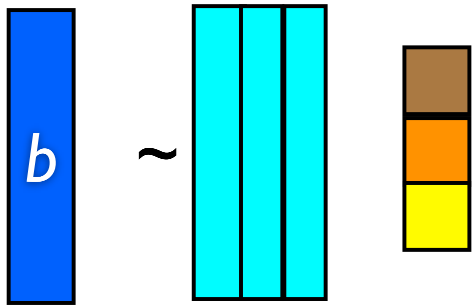


Courtesy: M. Davies, U. Edinburgh

Sparsity and subset selection



- Under-determined system
  - ✓ Infinitely many solutions
- If vector is sparse:
  - ✓ If support is known (and columns independent)
    - ✦ nonzero values characterized by (over)determined linear problem
  - ✓ **If support is unknown**
    - ✦ Main issue = finding the support!
    - ✦ This is the **subset selection problem**

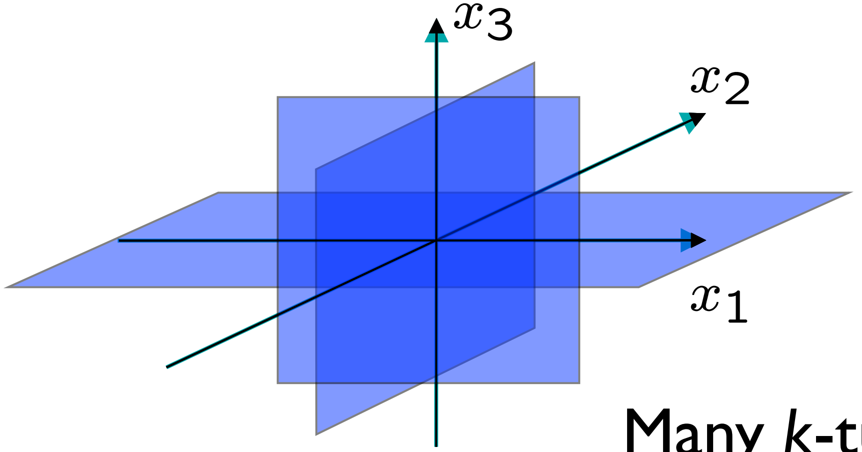


- Objectives of the course
    - ✦ **Well-posedness** of subset selection
    - ✦ Efficient subset selection algorithms = **pursuit algorithms**
    - ✦ **Stability guarantees** of pursuits

Complexity of Ideal Sparse Approximation

- Naive: Brute force search

$$\min_x \|\mathbf{b} - \mathbf{A}x\|_2 \text{ s.t. support}(x) = I$$



$$\begin{bmatrix} a_{11} & a_{13} \\ a_{21} & a_{23} \end{bmatrix} \cdot \begin{pmatrix} x_1 \\ x_3 \end{pmatrix} = \mathbf{b}$$

$$\begin{pmatrix} x_1 \\ x_3 \end{pmatrix} = \begin{bmatrix} a_{11} & a_{13} \\ a_{21} & a_{23} \end{bmatrix}^{-1} \cdot \mathbf{b}$$

Many *k*-tuples to try!

- **Theorem** *(Davies et al, Natarajan)*

  Solving the L0 optimization problem is **NP-complete**

Overview of greedy algorithms

$$\mathbf{b} = \mathbf{A}x_i + \mathbf{r}_i \qquad\qquad \mathbf{A} = [\mathbf{A}_1, \ldots \mathbf{A}_N]$$

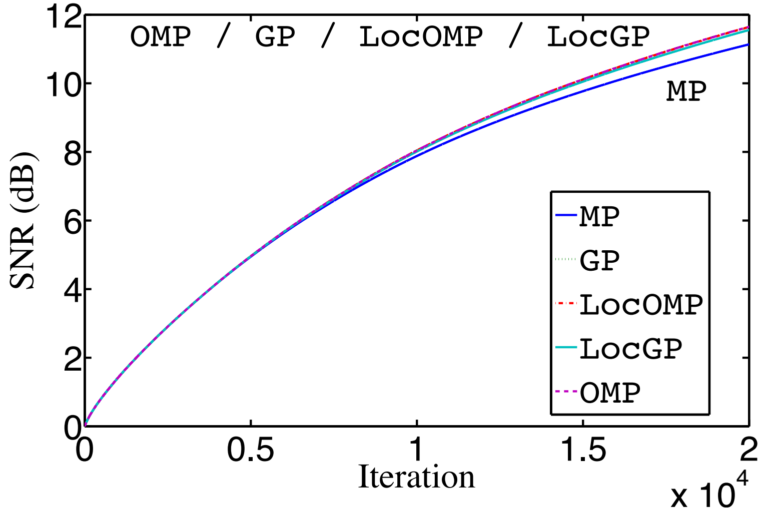
<table>
<tr><th></th><th>Matching Pursuit</th><th>OMP</th><th>Stagewise OMP</th></tr>
<tr><td>Selection</td><td colspan="2">$\Gamma_i := \arg\max_n |\mathbf{A}_n^T \mathbf{r}_{i-1}|$</td><td>$\Gamma_i := \{n \mid |\mathbf{A}_n^T \mathbf{r}_{i-1}| > \theta_i\}$</td></tr>
<tr><td>Update</td><td>$\Lambda_i = \Lambda_{i-1} \cup \Gamma_i$
$x_i = x_{i-1} + \mathbf{A}_{\Gamma_i}^+ \mathbf{r}_{i-1}$
$\mathbf{r}_i = \mathbf{r}_{i-1} - \mathbf{A}_{\Gamma_i} \mathbf{A}_{\Gamma_i}^+ \mathbf{r}_{i-1}$</td><td colspan="2">$\Lambda_i = \Lambda_{i-1} \cup \Gamma_i$
$x_i = \mathbf{A}_{\Lambda_i}^+ \mathbf{b}$
$\mathbf{r}_i = \mathbf{b} - \mathbf{A}_{\Lambda_i} x_i$</td></tr>
</table>

MP & OMP: *Mallat & Zhang 1993*
StOMP: *Donoho & al 2006* (similar to MCA, *Bobin & al 2006*)

OMP versus MP

- SNR as a function of iteration number

$$\mathrm{SNR} = 10 \log_{10} \frac{\|\mathbf{b}\|_2^2}{\|\mathbf{r}_i\|_2^2}$$



Overview of the course

- Session 1: Introduction
- Session 2: Complexity & Feasibility
  - ✓ Difficulty of ideal sparse approximation
  - ✓ Greedy algorithms
- **Session 3: Convex Pursuit Algorithms,**
- Session 4-6: Recovery Guarantees, Dictionaries & Compressive Sensing, Beyond sparsity

Convex Pursuit Algorithms

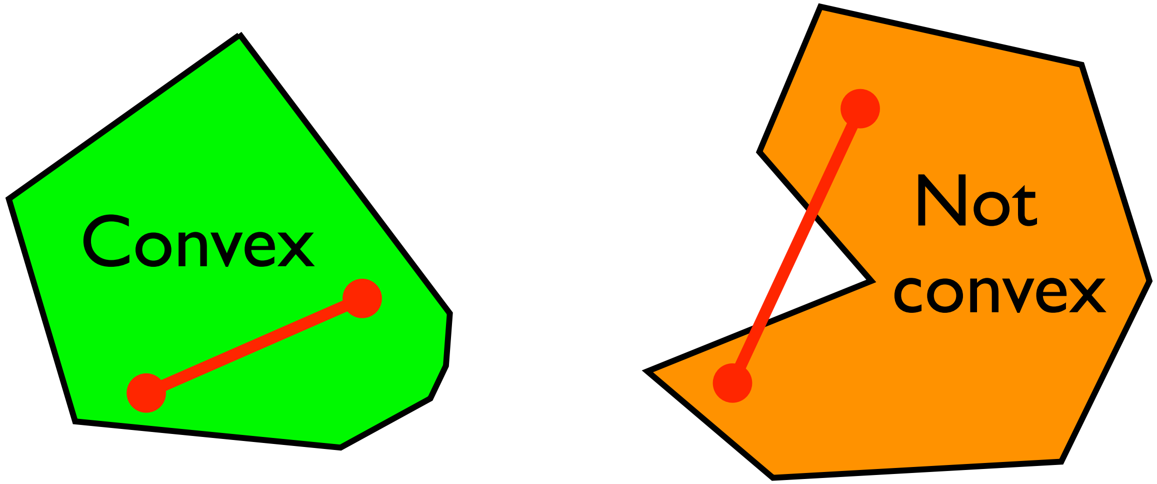
Sparse optimization *principles*

L1 minimization *induces* sparsity

Algorithms for L1 minimization

Convex sets

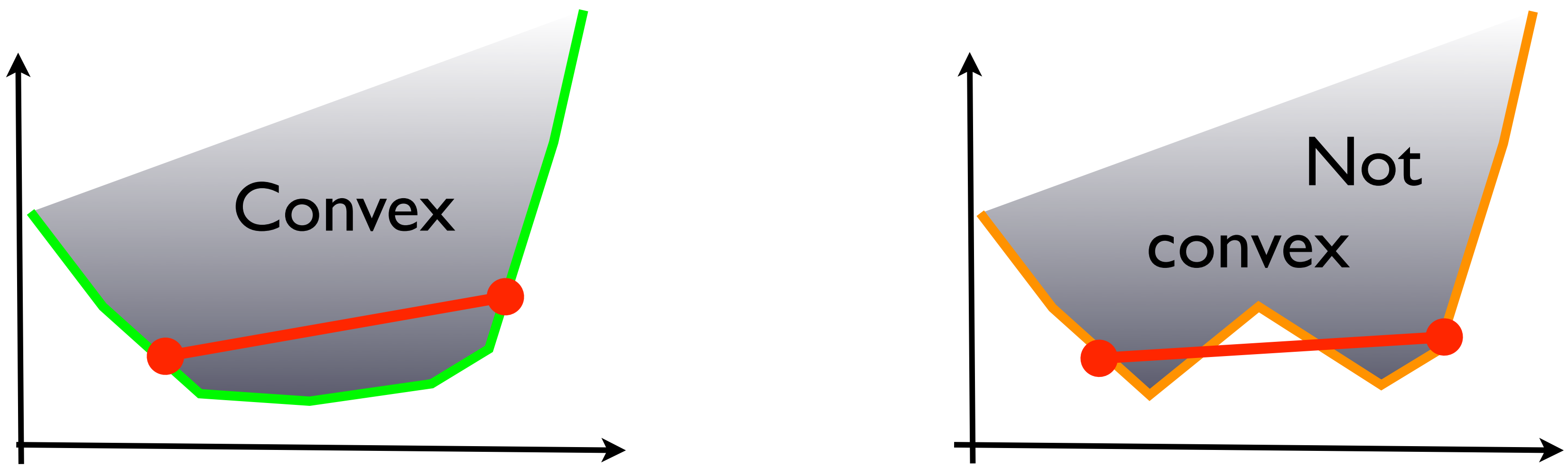
- Definition: $\theta \mathbf{u} + (1-\theta)\mathbf{v} \in X, \quad \forall \mathbf{u}, \mathbf{v} \in X, 0 \leq \theta \leq 1$



Convex functions

- Definition: f is convex if $\forall \mathbf{u}, \mathbf{v}, 0 \leq \theta \leq 1$

$$f(\theta \mathbf{u} + (1-\theta)\mathbf{v}) \leq \theta f(\mathbf{u}) + (1-\theta) f(\mathbf{v})$$



Overall compromise

- Approximation quality

$$\|\mathbf{A}x - \mathbf{b}\|_2$$

- Ideal sparsity measure : ℓ^0 "norm"

$$\|x\|_0 := \sharp\{n,\ x_n \neq 0\} = \sum_n |x_n|^0$$

- "Relaxed" sparsity measures

$$0 < p < \infty,\ \|x\|_p := \Big(\sum_n |x_n|^p\Big)^{1/p}$$

L*p* norms / quasi-norms

- **Norms** when $1 \leq p < \infty$ = convex

$$\|x\|_p = 0 \Leftrightarrow x = 0$$

$$\|\lambda x\|_p = |\lambda| \|x\|_p, \forall \lambda, x$$

Triangle inequality $$\|x + y\|_p \leq \|x\|_p + \|y\|_p, \forall x, y$$

- **Quasi-norms** when $0 < p < 1$ = nonconvex

Quasi-triangle inequality

$$\|x + y\|_p \leq 2^{1/p} \big(\|x\|_p + \|y\|_p\big), \forall x, y$$

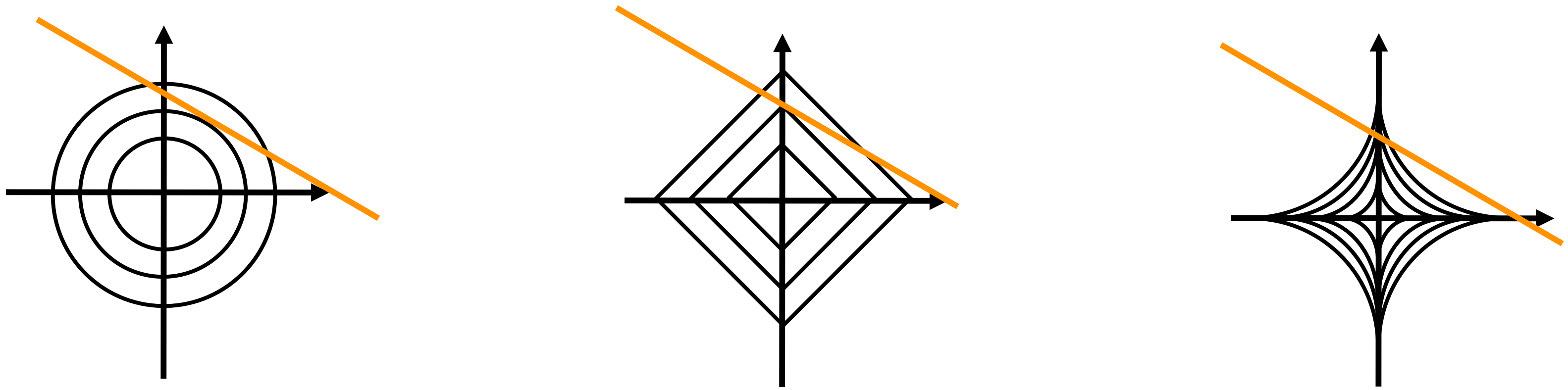
$$\|x + y\|_p^p \leq \|x\|_p^p + \|y\|_p^p, \forall x, y$$

- "*Pseudo*"-norm for p=0

$$\|x + y\|_0 \leq \|x\|_0 + \|y\|_0, \forall x, y$$

L*p* “norms” level sets

- Strictly convex when *p*>1
- Convex *p*=1
- Nonconvex *p*<1



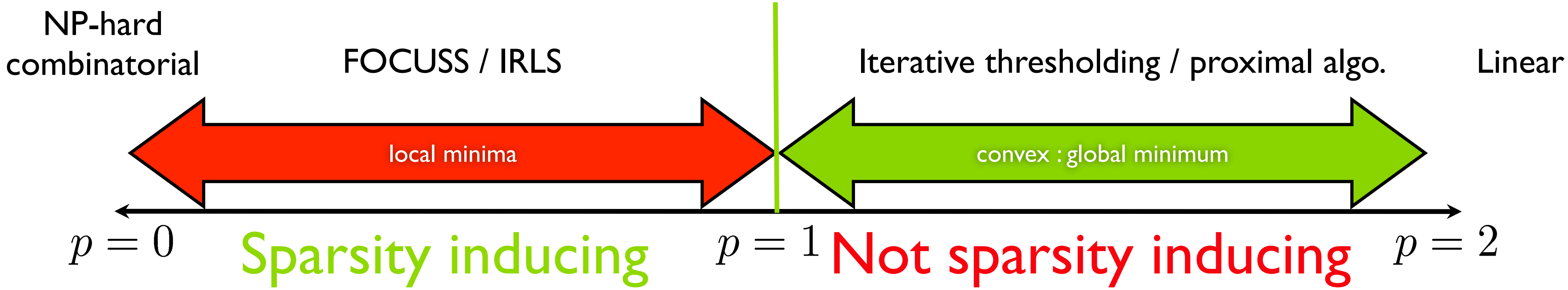
Observation: *the minimizer is sparse when p<=1*

$\{x \text{ s.t.} \mathbf{b} = \mathbf{A}x\}$

Global Optimization : from Principles to Algorithms

- Optimization principle
$$\min_x \frac{1}{2}\|\mathbf{A}x - \mathbf{b}\|_2^2 + \lambda \|x\|_p^p$$

  - ✓ Sparse representation $\quad \lambda \to 0 \quad \mathbf{A}x = \mathbf{b}$
  - ✓ Sparse approximation $\quad \lambda > 0 \quad \mathbf{A}x \approx \mathbf{b}$



Lasso [Tibshirani 1996], *Basis Pursuit (Denoising)* [Chen, Donoho & Saunders, 1999]
Linear/Quadratic programming (interior point, etc.)
Homotopy method [Osborne 2000] / Least Angle Regression [Efron &al 2002]
Iterative / proximal algorithms [Daubechies, de Frise, de Mol 2004, Combettes & Pesquet 2008, Beck & Teboulle 2009 ...]

Summary

	Global optimization	Iterative greedy algorithms
Principle	$\min_x \frac{1}{2}\|\mathbf{A}x - \mathbf{b}\|_2^2 + \lambda\|x\|_p^p$	iterative decomposition $\mathbf{r}_i = \mathbf{b} - \mathbf{A}x_i$ • select new components • update residual
Tuning quality/sparsity	regularization parameter λ	stopping criterion (nb of iterations, error level, ...) $\|x_i\|_0 \geq k$ $\|\mathbf{r}_i\| \leq \epsilon$
Variants	• choice of sparsity measure p • **optimization algorithm** • initialization	•selection criterion (weak, stagewise ...) •update strategy (orthogonal ...)

Convex Pursuit Algorithms

Sparse optimization *principles*

L1 minimization *induces* sparsity

Algorithms for L1 minimization

L1 *induces* sparsity (1)

- Real-valued case
  - ✓ **A** = an *m* x *N* real-valued matrix, where m < N
  - ✓ **b** = an *m*-dimensional real-valued vector
  - ✓ *X* = set of all minimum L*1* norm solutions to $\mathbf{A}x = \mathbf{b}$

$$\tilde{x} \in X \Leftrightarrow \|\tilde{x}\|_1 = \min_x \|x\|_1 \text{ s.t. } \mathbf{A}x = \mathbf{b}$$

- **Fact 1**: *X* is convex and contains a "sparse" solution

$$\exists x_0 \in X \subset \mathbb{R}^N, \|x_0\|_0 \leq m < N$$

Proof ? Exercice!

Proof ? Exercice!

- **Convexity of the set of solutions *X*:**
  - ✓ let $x, x' \in X,\ 0 \leq \theta \leq 1$
  - ✓ convexity of constraint

$$\mathbf{A}x = \mathbf{A}x' = \mathbf{A}(\theta x + (1-\theta)x') = \mathbf{b}$$

  - ✓ by definition $\|x\|_1 = \|x'\|_1 = \min \|\tilde{x}\|_1 \text{ s.t. } \mathbf{A}\tilde{x} = \mathbf{b}$
  - ✓ convexity of objective function

$$\|(\theta x + (1-\theta)x')\|_1 \leq \theta\|x\|_1 + (1-\theta)\|x'\|_1 = \|x\|_1$$

  - ✓ hence

$$\theta x + (1-\theta)x' \in X$$

Exercice: Matlab code for (O)MP

- Full clean code would include some checking (column normalization, dimension checking, etc.)

```
function [x res] = mp(b,A,k)
% explain here what the function should do
....
end

function [x res] = omp(b,A,k)
% explain here what the function should do
....
end
```