Structured Sparsity in Numerical Optimisation

Université Côte D’Azur, Inria, France

by

Matteo Frigo

Nice - December 4, 2017

Matteo Frigo

- PhD Student @ EDSTIC
- Supervisor: Rachid Deriche
- Inria - Athena Project Team
- ERC Computational Brain Connectivity Mapping

Contents

1. Mathematics
   Proximal Operator, FISTA, convex proper lower semi continuous
2. Regularisation theory
   Sparsity, Hierarchy, Lasso
3. Brain Imaging
   Tractography, dMRI, Connectomics

Mathematics

Numerical Optimisation

- $\mathcal{H}$ is a set
- $\Phi : \mathcal{H} \to \mathbb{R}$
- Find

$$x^* = \operatorname*{argmin}_{x \in \mathcal{H}} \Phi(x)$$

Numerical Optimisation

- $\mathcal{H}$ is a set
- $\Phi : \mathcal{H} \to \mathbb{R}$
- Find

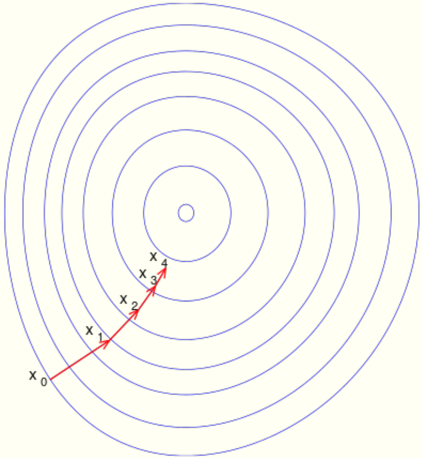
$$x^* = \operatorname*{argmin}_{x \in \mathcal{H}} \Phi(x)$$

Our setting:

- $\mathcal{H} = \mathbb{R}^d$
- $\Phi(x) := f(x) + g(x)$
  - $f(x)$ is convex and has L-Lipschitz continuous gradient
  - $g(x)$ is convex and lower semi-continuous
- Find

$$x^* = \operatorname*{argmin}_{x \in \mathbb{R}^d} f(x) + g(x)$$

Smooth case



Smooth case:

$$x^* = \operatorname*{argmin}_{x \in \mathbb{R}^d} f(x).$$

Iteration:

$$x^+ = x - \gamma \nabla f(x)$$

with $\gamma = \frac{1}{L}$

Non-smooth case

Let $\{C_j\}_j$ sequence of cvx subsets of $\mathbb{R}^d$ with non-empty intersection,

$$x^* = \operatorname*{argmin}_{x \in \mathbb{R}^d} \sum_j \iota_{C_j}(c)$$

Non-smooth case

Let $\{C_j\}_j$ sequence of cvx subsets of $\mathbb{R}^d$ with non-empty intersection,

$$x^* = \operatorname*{argmin}_{x \in \mathbb{R}^d} \sum_j \iota_{C_j}(c)$$

POCS: Projection Onto Convex Sets

$$x_{k+1} = \Pi_{C_1} \Pi_{C_2} \cdots \Pi_{C_N} x_k.$$

Non-smooth case

Let $\{C_j\}_j$ sequence of cvx subsets of $\mathbb{R}^d$ with non-empty intersection,

$$x^* = \operatorname*{argmin}_{x \in \mathbb{R}^d} \sum_j \iota_{C_j}(c)$$

POCS: Projection Onto Convex Sets

$$x_{k+1} = \Pi_{C_1} \Pi_{C_2} \cdots \Pi_{C_N} x_k.$$

Each projection is the solution of

$$\operatorname*{argmin}_{y \in \mathbb{R}^d} \frac{1}{2} \|x_k - y\|_2^2 + \iota_{C_j}(y).$$

Proximal operator

$\Gamma_0(\mathbb{R}^d) = \{g : \mathbb{R}^d \to \mathbb{R} \text{ l.s.c. and cvx with } \operatorname{dom}(g) \neq \emptyset\}$

Definition (Proximal Operator)

Let $g \in \Gamma_0(\mathbb{R}^d)$. For every $x \in \mathbb{R}^d$, the minimisation problem

$$\operatorname*{argmin}_{y \in \mathbb{R}^d} \frac{1}{2} \|x - y\|_2^2 + g(y)$$

admits a unique solution called $\operatorname{prox}_g(x)$.

Proximal operator

$\Gamma_0(\mathbb{R}^d) = \{g : \mathbb{R}^d \to \mathbb{R}$ l.s.c. and cvx with $\operatorname{dom}(g) \neq \emptyset\}$

Definition (Proximal Operator)

Let $g \in \Gamma_0(\mathbb{R}^d)$. For every $x \in \mathbb{R}^d$, the minimisation problem

$$\operatorname*{argmin}_{y \in \mathbb{R}^d} \frac{1}{2} \|x - y\|_2^2 + g(y)$$

admits a unique solution called $\operatorname{prox}_g(x)$.

prox **is single valued**

Properties of prox

Separability:

$$g(x, y) = \varphi(x) + \psi(y) = \mathrm{prox}_{\varphi}(x) + \mathrm{prox}_{\psi}(y)$$

Firmly non-expansive:

$$\left\|\mathrm{prox}_g(x) - \mathrm{prox}_g(y)\right\|_2^2 \leq (x - y)^T \left(\mathrm{prox}_g(x) - \mathrm{prox}_g(y)\right)$$

Resolvent operator:

$$\mathrm{prox}_g(\cdot) = (I + \partial g)^{-1}(\cdot)$$

Moreau decomposition

Definition

The convex conjugate of $g : X \to \mathbb{R}$ is $g^* : X^* \to \mathbb{R}$

$$g^*(\xi) = \sup_{x \in X} \{\langle \xi, x \rangle - g(x)\} .$$

Moreau decomposition

Definition

The convex conjugate of $g : X \to \mathbb{R}$ is $g^* : X^* \to \mathbb{R}$

$$g^*(\xi) = \sup_{x \in X} \{\langle \xi, x \rangle - g(x)\}.$$

Moreau decomposition:

$$v = \mathrm{prox}_g(v) + \mathrm{prox}_{g^*}(v)$$
$$v = \Pi_L(v) + \Pi_{L^\perp}(v)$$
$$v = \Pi_K(v) + \Pi_{K^\circ}(v)$$

Proof.

$2 + 2 = 4 - 1 = 3$ quick mafhs. $\square$

Key property

A point $x^* \in \mathbb{R}^n$ is a minimiser of g if and only if

$$\operatorname{prox}_g(x^*) = x^*.$$

Greedy algorithm for non-smooth optimisation

$$x^* = \operatorname*{argmin}_{x \in \mathbb{R}^d} g(x)$$

Greedy algorithm for non-smooth optimisation

$$x^* = \operatorname*{argmin}_{x \in \mathbb{R}^d} g(x)$$

Considerations:

Greedy algorithm for non-smooth optimisation

$$x^* = \operatorname*{argmin}_{x \in \mathbb{R}^d} g(x)$$

Considerations:

- prox is non-expansive

Greedy algorithm for non-smooth optimisation

$$x^* = \operatorname*{argmin}_{x \in \mathbb{R}^d} g(x)$$

Considerations:

- prox is non-expansive
- The minimiser x^* is a fixed point of prox

Greedy algorithm for non-smooth optimisation

$$x^* = \operatorname*{argmin}_{x \in \mathbb{R}^d} g(x)$$

Considerations:

- prox is non-expansive
- The minimiser x^* is a fixed point of prox

Iteration:

$$x_{k+1} = \mathrm{prox}_g\,(x_k)$$

Greedy algorithm for non-smooth optimisation

$$x^* = \underset{x \in \mathbb{R}^d}{\operatorname{argmin}}\, g(x)$$

Considerations:

- prox is non-expansive
- The minimiser x^* is a fixed point of prox

Iteration:

$$x_{k+1} = \operatorname{prox}_g (x_k)$$

If we don't know anything about the Lipschitz constant of prox_g

$$x_{k+1} = [(1-\alpha)I + \alpha \operatorname{prox}_g](x_k)$$

Cominetti et al., On the rate of convergence of Krasnoselskii-Mann iterations and their connection with sums of Bernoullis, 2014

Splitting algorithms

The problem

Let $f \in C^1(\mathbb{R}^d)$ and $g \in \Gamma_0\left(\mathbb{R}^d\right)$ two convex, proper and l.s.c. functions.

$$x^* = \operatorname*{argmin}_{y \in \mathbb{R}^d} f(y) + g(y).$$

Splitting algorithms

The problem

Let $f \in C^1(\mathbb{R}^d)$ and $g \in \Gamma_0\left(\mathbb{R}^d\right)$ two convex, proper and l.s.c. functions.

$$x^* = \operatorname*{argmin}_{y \in \mathbb{R}^d} f(y) + g(y).$$

Idea

Gradient descent for the smooth part and proximal iteration for the non-smooth.

Forward-backward splitting

First order optimality condition

$$
\begin{aligned}
0 &\in \nabla f(x^*) + \partial g(x^*) \\
0 &\in \gamma \nabla f(x^*) + \gamma \partial g(x^*) \\
\gamma \nabla f(x^*) &\in \gamma \partial g(x^*) \\
(I - \gamma \nabla f)(x^*) &\in (I + \gamma \partial g)(x^*) \\
(I + \gamma \partial g)^{-1}(I - \gamma \nabla f)(x^*) &= x^* \\
\mathrm{prox}_{\gamma g}(x^* - \gamma \nabla f(x^*)) &= x^*
\end{aligned}
$$

Forward-backward splitting

First order optimality condition

$$
\begin{aligned}
0 &\in \nabla f(x^*) + \partial g(x^*) \\
0 &\in \gamma \nabla f(x^*) + \gamma \partial g(x^*) \\
\gamma \nabla f(x^*) &\in \gamma \partial g(x^*) \\
(I - \gamma \nabla f)(x^*) &\in (I + \gamma \partial g)(x^*) \\
(I + \gamma \partial g)^{-1} (I - \gamma \nabla f)(x^*) &= x^* \\
\operatorname{prox}_{\gamma g}(x^* - \gamma \nabla f(x^*)) &= x^*
\end{aligned}
$$

$$
x_{k+1} = \operatorname{prox}_{\gamma g}(x_k - \gamma \nabla f(x_k))
$$

FISTA

Fast Iterative Shrinkage Thresholding Algorithm (2009)

Objective:

$$x^* = \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \overbrace{f(x)}^{\text{s}} + \overbrace{g(x)}^{\text{ns}}$$

Algorithm ($t_0 = 1$):

$$x_k = \operatorname{prox}_{t_k g}\left(x_{k-1} - t_k \nabla f(x_{k-1})\right)$$

$$t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2}$$

$$y_{k+1} = x_k + \left(\frac{t_k - 1}{t_{k+1}}\right)(x_k - x_{k-1})$$

Rate of convergence:

$$\mathcal{O}\left(1/k^2\right)$$

Sum up...

- We are able to solve smooth + non smooth minimisation problems
- We need L-Lipschitz gradients for the smooth term
- We need to be able to compute prox of the non smooth term

Regularisation theory

Forward/Inverse problems

Forward:

- Input
  - Linear model $A : \mathbb{R}^d \to \mathbb{R}^n$
  - Set of weights $x \in \mathbb{R}^d$
  - Noise ε
- Output:
  - Data $y = Ax + \varepsilon$

Inverse:

- Input
  - Data $y \in \mathbb{R}^n$
  - Linear model $A : \mathbb{R}^d \to \mathbb{R}^n$
  - Desire to recover $x \in \mathbb{R}^d$
- Output:
  - Set of weights $x \in \mathbb{R}^d$

Forward/Inverse problems

Forward:

- Input
  - Linear model $A : \mathbb{R}^d \to \mathbb{R}^n$
  - Set of weights $x \in \mathbb{R}^d$
  - Noise ε
- Output:
  - Data $y = Ax + \varepsilon$

Inverse:

- Input
  - Data $y \in \mathbb{R}^n$
  - Linear model $A : \mathbb{R}^d \to \mathbb{R}^n$
  - Desire to recover $x \in \mathbb{R}^d$
- Output:
  - Set of weights $x \in \mathbb{R}^d$

NB: the noise is missing in the inverse problem

Inverse problem

$$x^* = \operatorname*{argmin}_{x \in \mathbb{R}^d} \underbrace{\frac{1}{2} \|Ax - y\|_2^2}_{\text{smooth}} + \underbrace{\lambda \Omega(x)}_{?}$$

with $\Omega : \mathbb{R}^d \to \mathbb{R}$ convex, proper and l.s.c.

Tikonov regularisation

$$x^* = \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \frac{1}{2} \|Ax - y\|_2^2 + \lambda \|x\|_2^2$$

- Everything is smooth with L-Lipschitz gradient
- No need of proximal stuff

Smoothing splines

$$x^* = \operatorname*{argmin}_{x \in \mathbb{R}^d} \frac{1}{2} \|Ax - y\|_2^2 + \lambda \|\Delta x\|_2^2$$

- $\lambda \to 0$: interpolating spline
- $\lambda \to \infty$: least square estimate

Sparse recovery

Goal: extract the **few relevant features** of x that let us reconstruct y.

Sparse recovery

Goal: extract the **few relevant features** of x that let us reconstruct y.

Idea

Penalise with the ℓ_0 "norm" $\|x\|_0 = \sum x_j^0$

Sparse recovery

Goal: extract the **few relevant features** of x that let us reconstruct y.

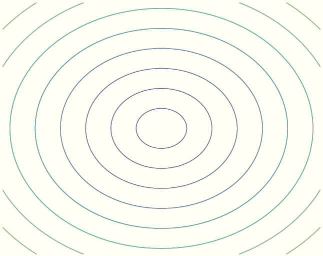
Idea

Penalise with the ℓ_0 "norm" $\|x\|_0 = \sum x_j^0$

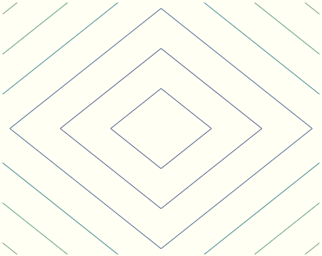
Problem

Combinatorial complexity

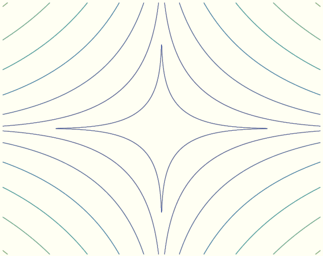
Shape of ℓ_p balls



ℓ_2

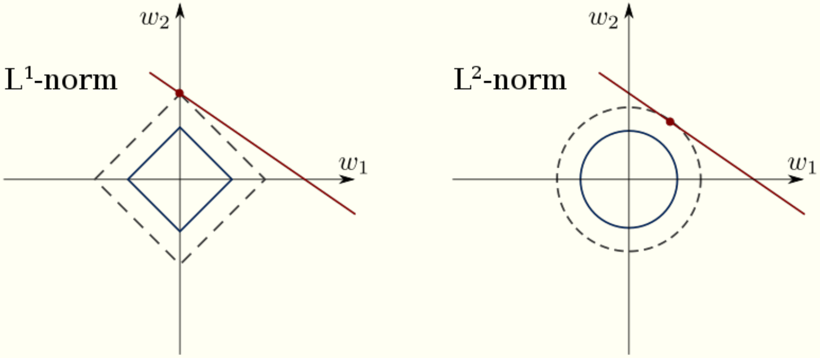


ℓ_1



$\ell_{0.4}$

Sparse recovery



Sparse recovery (Lasso)

$$x^* = \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \frac{1}{2} \|Ax - y\|_2^2 + \lambda \|x\|_1$$

$$x^* = \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \|x\|_1 \qquad \text{s.t. } \|Ax - y\|_2^2 \leq \varepsilon$$

prox of norms

Consider $g = \|\cdot\|$ and $\mathcal{B} = \{x : \|x\|_* \leq 1\}$, then

$$g^*(\xi) = \iota_{\mathcal{B}}(\xi)$$

By Moreau decomposition:

$$v = \operatorname{prox}_{\|\cdot\|}(v) + \Pi_{\mathcal{B}}(v)$$
$$\operatorname{prox}_{\|\cdot\|}(v) = v - \Pi_{\mathcal{B}}(v)$$

prox of norms

Consider $g = \|\cdot\|$ and $\mathcal{B} = \{x : \|x\|_* \leq 1\}$, then

$$g^* (\xi) = \iota_{\mathcal{B}} (\xi)$$

By Moreau decomposition:

$$v = \operatorname{prox}_{\|\cdot\|} (v) + \Pi_{\mathcal{B}}(v)$$
$$\operatorname{prox}_{\|\cdot\|} (v) = v - \Pi_{\mathcal{B}}(v)$$

We have the proximal operator of norms!

Sparse recovery can be solved

Structured sparsity

Necessity to *exclude* blocks of x

- Non-Overlapping Group Lasso
- Overlapping Group Lasso
- Hierarchical Lasso

Non Overlapping Group Lasso

$$\Omega(x) = \|x_{\mathcal{G}}\|_1 = \sum_{g \in \mathcal{G}} w_g \left\|x_{|g}\right\|_2$$

Non Overlapping Group Lasso

$$\Omega(x) = \|X_{\mathcal{G}}\|_1 = \sum_{g \in \mathcal{G}} w_g \left\|x_{|g}\right\|_2$$

By separability of prox we have

$$\mathrm{prox}_{\Omega}(x) = \mathrm{prox}_{g_1}\left(\mathrm{prox}_{g_2}\left(\ldots \mathrm{prox}_{g_k}(x)\right)\right)$$

Overlapping Group Lasso

$$\Omega(x) = \sum_{g \in \mathcal{G}} w_g \left\| x_{|g} \right\|_2$$

Overlapping Group Lasso

$$\Omega(x) = \sum_{g \in \mathcal{G}} w_g \left\| x_{|g} \right\|_2$$

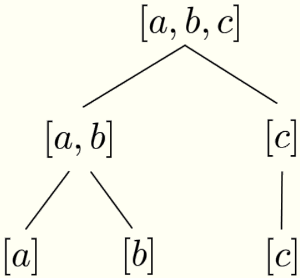
Much more complicated

Overlapping Group Lasso

$$\Omega(x) = \sum_{g \in \mathcal{G}} w_g \left\| x_{|g} \right\|_2$$

Much more complicated but feasible

Hierarchical Sparsity



$$\mathrm{prox}_{\Omega}\left(x\right) = \mathrm{prox}_{g_1}\left(\mathrm{prox}_{g_2}\left(\ldots \mathrm{prox}_{g_k}\left(x\right)\right)\right)$$

Proximal of Hierarchical Sparsity

Let $(\mathcal{G}, \preceq)$ be ad ordered tree structure

1. Set $v = x$.
2. For $g \in \mathcal{G}$ following $\preceq$ do

$$v_{|g} \longleftarrow v_{|g} - \Pi_{\|\cdot\|_* \leq \omega_g} \left(v_{|g} \right) .$$

Non-Negativity constraint

$$x^* = \operatorname*{argmin}_{x \in \mathbb{R}^d} \underbrace{f(x)}_{\text{s}} + \underbrace{g(x) + \iota_{\geq 0}(x)}_{\text{ns}}$$

Non-Negativity constraint

$$x^* = \operatorname*{argmin}_{x \in \mathbb{R}^d} \underbrace{f(x)}_{\text{s}} + \underbrace{g(x) + \iota_{\geq 0}(x)}_{\text{ns}}$$

Definition (Absolute norm)

A norm $\Omega : X \to \mathbb{R}$ is called *absolute* if $\forall u, v \in \mathbb{R}^N$ such that $|u_j| \leq |v_j|$ for all j implies $\Omega(u) \leq \Omega(v)$.

Theorem (Proximal operator of absolute norms)

Let $w \in \mathbb{R}^n$ and $\lambda > 0$. Consider an absolute norm Ω. We have

$$\operatorname*{argmin}_{z \in \mathbb{R}^n} \left[\frac{1}{2} \|[w]_+ - z\|_2^2 + \lambda \Omega(z) \right] = \operatorname*{argmin}_{z \in \mathbb{R}^n_+} \left[\frac{1}{2} \|w - z\|_2^2 + \lambda \Omega(z) \right].$$

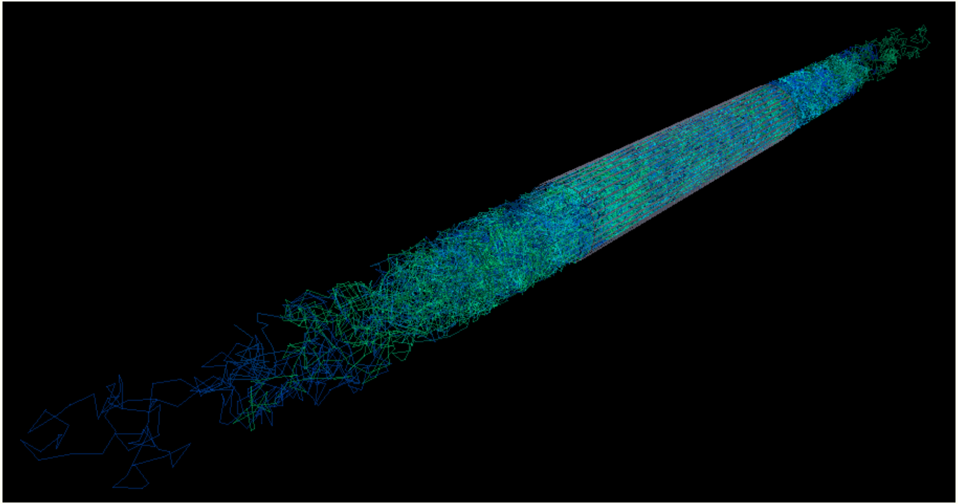
$$\operatorname{prox}_{\lambda\Omega}([w]_+) = \operatorname{prox}_{\lambda\Omega + \iota_{\geq 0}}(w)$$

Sum up...

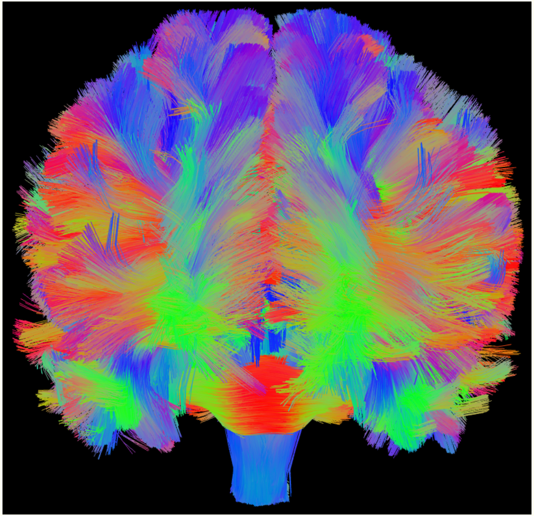
- We can recover sparse variables/signals/codes/...
- Intrinsic structures can be exploited
- Proximal methods are the correct tools for sparse reconstruction
- Non-Negativity constraint can be embedded almost for free

Brain Imaging

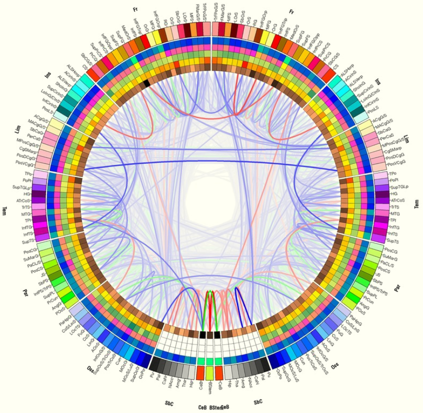
Diffusion MRI

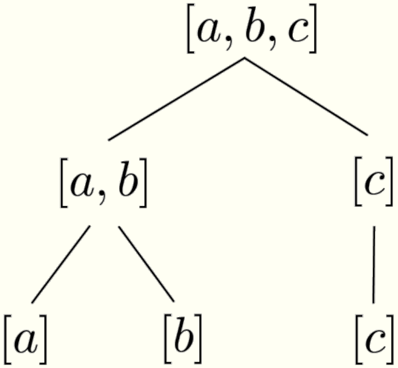


Tractography



Brain hierarchy





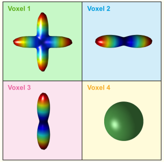
The problem: false positives

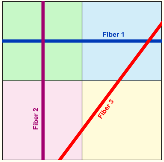
The challenge of mapping the human connectome based on diffusion tractography

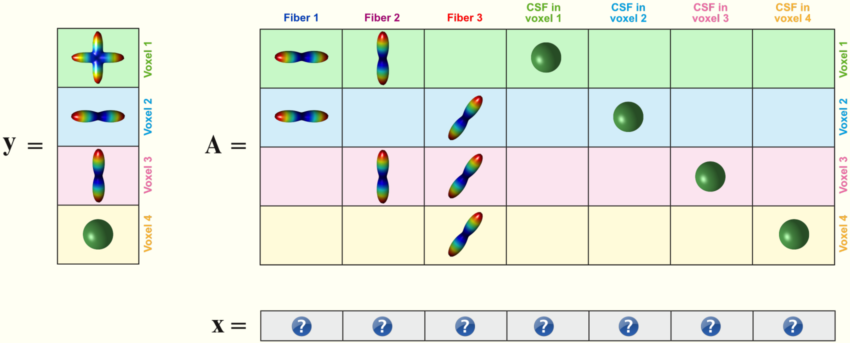
Maier-Hein et al., 2017 (Nature)

- 96 distinct tractography pipelines
- *“most algorithms routinely extracted many false positive bundles”*
- *“Tractography identifies more invalid than valid bundles”*
- **“Tractography is fundamentally ill-posed”**

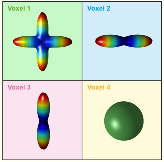
Forward model: COMMIT

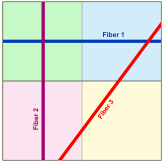


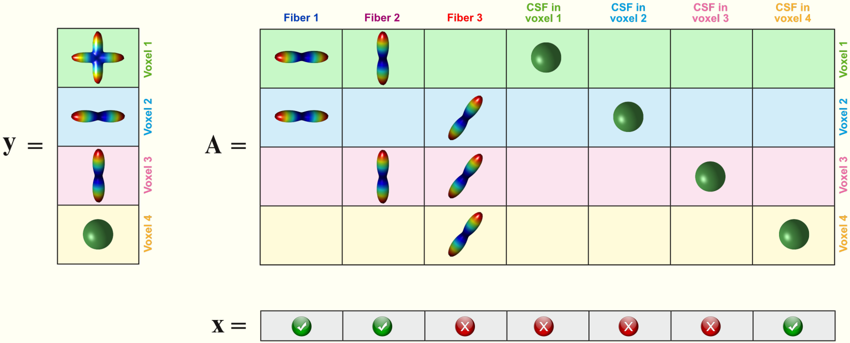




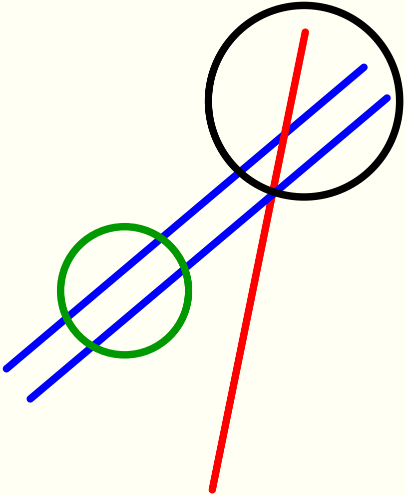
Forward model: COMMIT

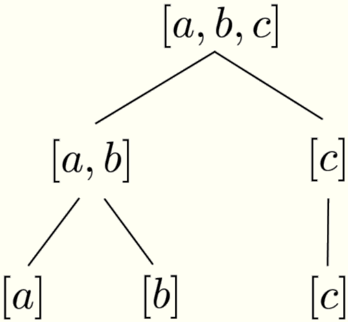






Hierarchical pattern





$$\mathcal{G} = \Big(\big[[a], [b], [a, b], [c], [a, b, c] \big], \preceq \Big)$$

NB:

$$[a] \preceq [a, b] \preceq [c] \qquad [a, b, c] \not\preceq [c]$$

False positives detection

- Build the dictionary/matrix A
- Fit the acquired dMRI signal
- Force hierarchical sparsity

False positives detection

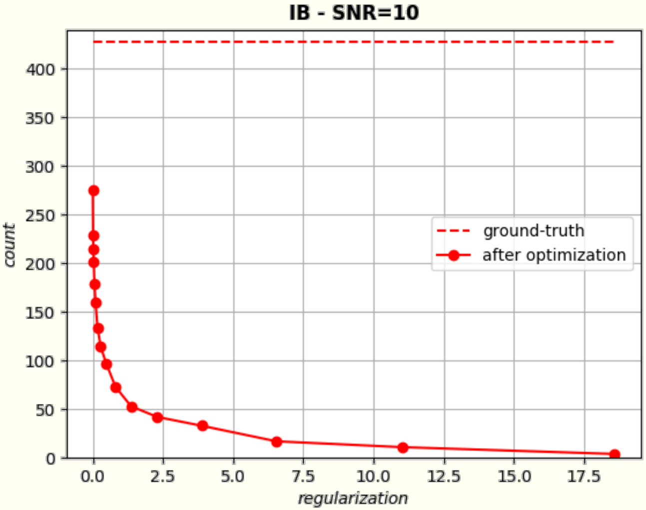
- Build the dictionary/matrix A
- Fit the acquired dMRI signal
- Force hierarchical sparsity
- **Impose non-negativity of** x

False positives detection

- Build the dictionary/matrix A
- Fit the acquired dMRI signal
- Force hierarchical sparsity
- **Impose non-negativity of** x

$$x^* = \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \frac{1}{2} \|Ax - y\|_2^2 + \lambda \sum_{g \in \mathcal{G}} w_g \left\|x_{|g}\right\|_2 + \iota_{\geq 0}(x)$$

Results


IB - SNR=10
count
regularization
0
50
100
150
200
250
300
350
400
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
ground-truth
after optimization

Sum up...

- Brain connections are organised with an hierarchical pattern
- False positives can be detected with structured sparsity
- The solution can be obtained via proximal splitting methods

Sum up...

- Brain connections are organised with an hierarchical pattern
- False positives can be detected with structured sparsity
- The solution can be obtained via proximal splitting methods
- It works :)

Sum up...

- Brain connections are organised with an hierarchical pattern
- False positives can be detected with structured sparsity
- The solution can be obtained via proximal splitting methods
- It works :)

☺Thank you ☺