Learning with constraints: Geometry and Sparsity

Ammar Mian

Réunion AfuTé - 05 November 2020

Joint work with:
Elias Raninen (Univ. of Aalto),
Esa Ollila (Univ. of Aalto)

Aalto University

LISTIC

UNIVERSITÉ SAVOIE MONT BLANC

Miscellaneous

- Works presented in this presentation:
  - **A. Mian**, E. Raninen, E. Ollila, "A Comparative Study of Supervised Learning Algorithms for Symmetric Positive Definite Features", in *IEEE 28th European Signal Processing Conference (EUSIPCO)*
  - E. Ollila, **A. Mian**, "Block-wise Minimization-Majorization Algorithm for Huber's Criterion: Sparse learning and Applications", in *IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*
  - F. Bouchard, **A. Mian**, J. Zhou, S. Said, G.Ginolhac, Y. Berthoumieu, "Riemannian geometry for compound Gaussian distributions: Application to recursive change detection", in *Elsevier Signal Processing*, 2020
- Papers and codes available at:

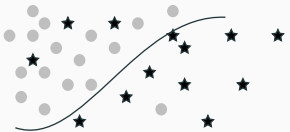
`https://ammarmian.github.io/`

Outline

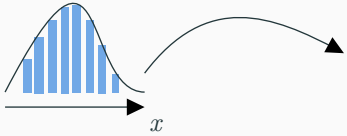
Outline

Learning from data

- Supervised and unsupervised classification:



- Parameter estimation:



$p_{x;\theta}(x;\theta)$

Estimateur: $\hat{\theta}$

- Denoising
- ...

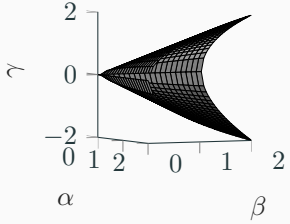
Objectives of this talk

Gives examples where a structure (constraint) in the data is taken into account

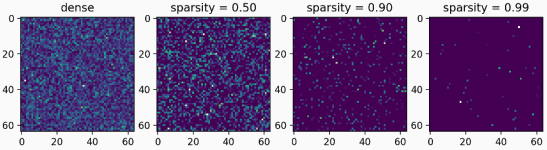
What type of structure are we talking about

- **Part I:** Geometric structure of the space of the data:

$$\mathbf{\Sigma} = \begin{bmatrix} \alpha & \gamma \\ \gamma & \beta \end{bmatrix}$$



- **Part II:** Sparsity



Outline

Non-Euclidean feature spaces

Many learning problems deal with data which is in a **non-Euclidean** space

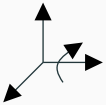
SPD matrices $\mathbb{S}_+^p$

fMRI
Computer vision
SAR images

Hypersphere S_p

Geography
PolSAR

Rotations $SO(3)$



Camera position
3D objects

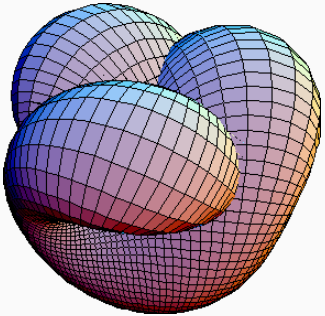
The data for some applications lie in non Eucldiean spaces!

Problems of Euclidean algorithms on non-Euclidean spaces

- The distances between points can be misleading:

- The mean of data points does not always belong to the input space:

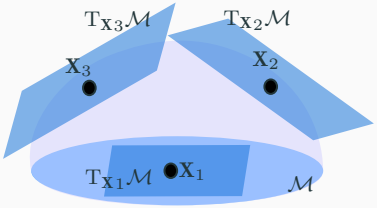
Manifolds



Manifold [AMS09]

A d-dimensional manifold can be informally defined as a set $\mathcal{M}$ which can be covered with a collection of coordinates patches that identify certain subsets of $\mathcal{M}$ with open subsets of $\mathbb{R}^d$.

Riemannian manifolds

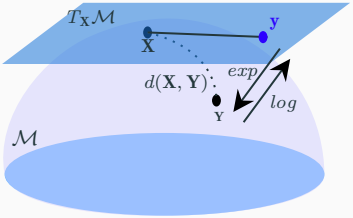


Riemannian Manifold [AMS09]

A Riemannian manifold $\mathcal{M}$ for which it is possible to define an inner product $\langle \cdot, \cdot \rangle_{\boldsymbol{X}}$ on the tangent space $\mathrm{T}_{\boldsymbol{X}}\mathcal{M}$ at each point $\mathbf{X}$ that varies "smoothly" in the manifold.

$\rightarrow$ We can define several useful tools on those spaces !

Riemannian manifolds tools



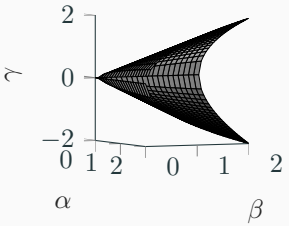
Geodesic distance

Length of the shortest curve between two points.

Tangent-space mappings

Exponential and logarithms mappings

Example of SPD matrices (1/2)



Several metrics can be defined: Log-Euclidean metric

Geodesic distance

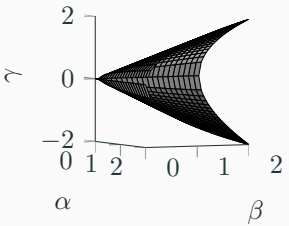
$$d_L(\mathbf{X}, \mathbf{Y}) = \| \log(\mathbf{X}) - \log(\mathbf{Y}) \|_{\mathrm{F}}, \tag{1}$$

where the $\log$ operator is defined for a SPD matrix $\mathbf{X}$ with eigenvalue decomposition $\mathbf{X} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top$ as:

$$\log(\mathbf{X}) = \boldsymbol{U} \log_{\odot}(\boldsymbol{\Lambda}) \boldsymbol{U}^\top, \tag{2}$$

with $\log_{\odot}$ being the point-wise logarithm function.

Example of SPD matrices (2/2)



Several metrics can be defined: Affine-invariant metric

Geodesic distance

$$d_A(\mathbf{X}, \mathbf{Y}) = \| \log(\mathbf{X}^{-1/2}\mathbf{Y}\mathbf{X}^{-1/2})\|_{\mathrm{F}}. \tag{3}$$

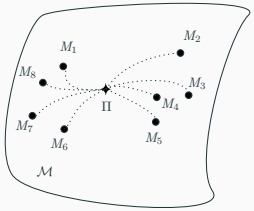
Logarithm mapping

$$\log_{\mathbf{X}}(\mathbf{Y}) = \mathbf{X}^{\frac{1}{2}} \log(\mathbf{X}^{-\frac{1}{2}}\mathbf{Y}\mathbf{X}^{-\frac{1}{2}})\mathbf{X}^{\frac{1}{2}}. \tag{4}$$

Exponential mapping

$$\exp_{\mathbf{X}}(\mathbf{y}) = \mathbf{X}^{\frac{1}{2}} \exp(\mathbf{X}^{-\frac{1}{2}}\mathbf{y}\mathbf{X}^{-\frac{1}{2}})\mathbf{X}^{\frac{1}{2}}. \tag{5}$$

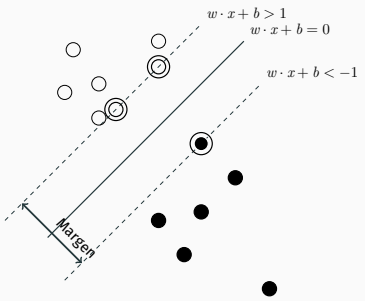
Riemannian mean



Riemannian mean

$$\mathbf{\Pi}\left(\{\alpha_i, \mathbf{M}_i\}_{1\leq i\leq N}\right) = \underset{\mathbf{M}\in\mathcal{M}}{\operatorname{argmin}} \sum_{i=1}^{N} \alpha_i d^2(\mathbf{M}, \mathbf{M}_i). \tag{6}$$

Learning procedure



Supervised learning: Given data $\{(\mathbf{x}_i, y_i) \in \mathbb{R}^d \times \mathcal{C} : 1 \leq i \leq N\}$, where $\mathcal{C}$ is a discrete set of class labels. Find a mapping

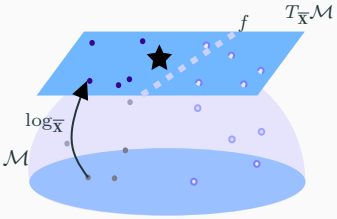
$$f : \mathbf{x} \in \mathbb{R}^d \to \mathcal{C},$$

that minimizes the empirical risk

$$R = N^{-1} \sum_i (f(\mathbf{x}_i) == y_i).$$

$\to$ How to adapt to Riemannian manifolds ?

1st solution: Map the data to a tangent space



Consider the following procedure:

$$f \circ \log_{\mathbf{\Pi}_A\left(\{1/N, \mathbf{X}_i\}_{1\leq i\leq N}\right)} : \mathcal{M} \to \mathcal{C}. \qquad (7)$$

Reference point $\mathbf{X}$: Riemannian mean $\mathbf{\Pi}_A\left(\{\alpha_i, \mathbf{X}_i\}_{1\leq i\leq N}\right)$ with weights $\alpha_i = 1/N$.

2nd solution: Use geodesic distances

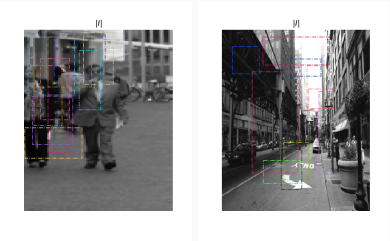
For algorithms based on distances:

- Minimum distance to mean:
  $\rightarrow$ Compute Riemannian mean of each class
  $\rightarrow$ Assign each test data to the closest mean (in the Riemannian sense)
- K-nearest neighbors::
  $\rightarrow$ For each test data compute distance to all training data
  $\rightarrow$ Assign class by majority of the K-nearest points

Works also for non-supervised classification:

- K-means
- Spectral clustering

Application: Pedestrian detection



Features:

$$\mathbf{z}(x,y) = \left[x, y, |I_x|, |I_y|, \sqrt{I_x^2 + I_y^2}, |I_{xx}|, |I_{yy}|, \arctan \frac{|I_x|}{|I_y|}\right].$$

The *covariance descriptor* of an arbitrary window W is a SPD matrix of the feature vectors $\mathbf{z}(x,y)$

$$\mathbf{C}_W = \frac{1}{n_x n_y - 1} \sum_{x,y \in W} (\mathbf{z}(x,y) - \bar{\mathbf{z}})(\mathbf{z}(x,y) - \bar{\mathbf{z}})^\top,$$

where $\bar{\mathbf{z}} = \frac{1}{n_x n_y} \sum_{x,y} \mathbf{z}(x,y)$ is the sample mean of $\mathbf{z}(x,y)$

Results on INRIA dataset

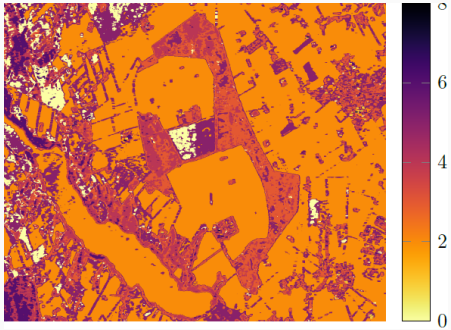
		Fold 1	Fold 2	Fold 3	Fold 4	mean
Euclidean	RBF SVM	0.819	0.823	0.819	0.820	0.820
	Logitboost	0.934	0.931	0.933	0.935	0.933
	KNN	0.780	0.781	0.780	0.783	0.781
	MDM	0.597	0.595	0.592	0.595	0.595
	LogisticRegression	0.831	0.831	0.832	0.831	0.831
Riemannian	RBF SVM	0.892	0.892	0.892	0.894	0.892
	Logitboost	**0.948**	**0.947**	**0.946**	**0.950**	**0.948**
	KNN	0.827	0.825	0.826	0.825	0.826
	MDM	0.692	0.698	0.701	0.699	0.697
	LogisticRegression	0.741	0.709	0.719	0.685	0.714

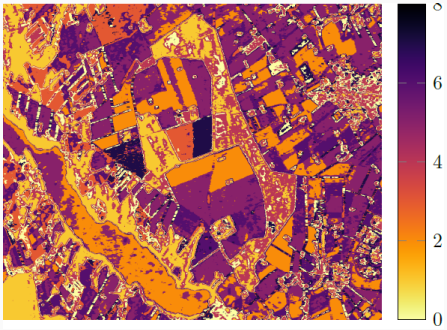
Application: H-α clustering of SAR images

- Data: Emisar Foulom, polarimetric data $p = 3$
- Image size: 1750 rows x 997 columns
- Features: Covariances computed on 7×7 patchs

Results

Left: K-means with Wishart distance and Euclidean mean
Right: K-means with Riemannian distance and mean





Outline

Impulsive noise denoising

(a) Original image

(b) Noisy image

Sparse dictionary model

On a patch of size $N = N_x \times N_y$ pixels, we consider the following model:

$$\mathbf{y} = \mathbf{X}\beta + \mathbf{e}, \tag{8}$$

where:

- $\mathbf{X}$ is an overcomplete dictionary (wavelets, 2D-DCT, etc)
- β is a sparse vector of K non-zero elements
- $\mathbf{e}$ accounts for the noise

Denoised image patch

$\hat{\mathbf{u}} = \mathbf{X}\hat{\beta}$

Link with linear regression

Suppose N sets $\{(x_i^{\mathrm{T}}, y_i) : 1 \leq i \leq N\}$ where:

- $y_i \in \mathbb{R}$, are the inputs (or predictors)
- $x_i^{\mathrm{T}} = (x_{i1}, \ldots, x_{ip}) \in \mathbb{R}^p$ are the outputs (or responses)

A linear model between the input and outputs is:

$$\begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix} = \begin{pmatrix} x_1^\top \\ \vdots \\ x_N^\top \end{pmatrix} \beta + \begin{pmatrix} e_1 \\ \vdots \\ e_N \end{pmatrix} \tag{9}$$

$$\mathbf{y} = \mathbf{X}\beta + \mathbf{e},$$

where the error terms e_i are i.i.d with p.d.f $f(e) = (1/\sigma) f_0(e/\sigma)$.

Goal

Estimate the regression coefficients $\beta = (\beta_1, \ldots, \beta_p) \in \mathbb{R}^p$ and the scale $\sigma > 0$.

Robust linear regression

Introduced in [Hub64], the Least Favorable Distribution (LFD):

Huber's unit scale ($\sigma = 1$) LFD

$$f_0(e) \propto \exp\left(-\rho_c(e)\right), \tag{10}$$

where:

$$\rho_c(x) = \frac{1}{2} \times \begin{cases} |x|^2, & \text{for } |x| \leq c \\ 2c|x| - c^2, & \text{for } |x| > c, \end{cases}, \quad x \in \mathbb{R},$$

is called as the **Huber's loss function** and c is a user-defined *threshold*.

Robust linear regression with Maximum likelihood criterion

- The Maximum Likelihood (ML) criterion for this yields:

$$L_{\mathrm{ML}}(\beta, \sigma) = -\sum_{i=1}^{N} \ln \left\{ \frac{1}{\sigma} f_0 \left(\frac{y_i - x_i^\top \beta}{\sigma} \right) \right\}$$
$$= N \ln \sigma + \sum_{i=1}^{N} \rho_c \left(\frac{y_i - x_i^\top \beta}{\sigma} \right).$$

It fails to:

Robust linear regression with Maximum likelihood criterion

- The Maximum Likelihood (ML) criterion for this yields:

$$L_{\mathrm{ML}}(\beta,\sigma) = -\sum_{i=1}^{N} \ln\left\{\frac{1}{\sigma} f_0\left(\frac{y_i - x_i^\top \beta}{\sigma}\right)\right\}$$
$$= N\ln\sigma + \sum_{i=1}^{N} \rho_c\left(\frac{y_i - x_i^\top \beta}{\sigma}\right).$$

It fails to:

- be convex in (β,σ),

Robust linear regression with Maximum likelihood criterion

- The Maximum Likelihood (ML) criterion for this yields:

$$L_{\mathrm{ML}}(\beta,\sigma) = -\sum_{i=1}^{N} \ln\left\{\frac{1}{\sigma} f_0\left(\frac{y_i - x_i^\top \beta}{\sigma}\right)\right\}$$
$$= N\ln\sigma + \sum_{i=1}^{N} \rho_c\left(\frac{y_i - x_i^\top \beta}{\sigma}\right).$$

It fails to:

- be convex in (β,σ),
- provide robust estimates (bounded influence functions).

Robust linear regression with Huber's criterion

- Huber proposed to minimize the following criterion[1]:

$$L(\beta, \sigma) = N(\alpha\sigma) + \sum_{i=1}^{N} \rho_c \left(\frac{y_i - x_i^\top \beta}{\sigma} \right) \sigma. \tag{11}$$

The criterion [Hub64]:

- is convex in (β, σ),
- provides robust estimates (bounded influence functions).

[1] α is a scaling factor chosen to ensure Fisher-consistency to the scale of a distribution of interest. For Gaussian consistency, we chose $\alpha = \frac{1}{2}$.

Minimizing Huber's criterion

We want to solve the following:

Optimization problem

$$(\hat{\beta}, \hat{\sigma}) = \operatorname{argmin}_{\beta,\sigma} L(\beta, \sigma)$$

No close-form solution but solution can be found:

- Gradient descent approach (convex),
- Majorization-minimization (MM) algorithm (our contribution [OM20])

Majorization-minimization procedure

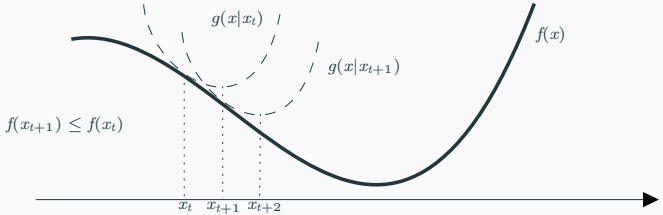


Figure 2: Principle of the algorithm

Interest of this approach

- Can allow to minimize locally heavily non-convex functions
- When surrogate functions are simple, closed-form updates are possible

Numerical examples: regression results

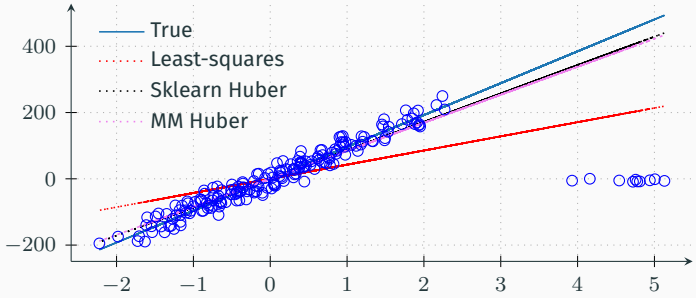


Figure 3: Comparison of regression results with two implementation of Huber's criterion. $p = 1$, $N = 200$, $c = 1.345$, $\sigma = 20$, $\beta = 96.19$.

Sparse dictionary model for denoising

On a patch of size $N = N_x \times N_y$ pixels, we consider the following model:

$$\mathbf{y} = \mathbf{X}\beta + \mathbf{e}, \tag{12}$$

where:

- $\mathbf{X}$ is an overcomplete dictionary (wavelets, 2D-DCT, etc)
- β is a sparse vector of K non-zero elements
- $\mathbf{e}$ accounts for the noise

Denoised image patch

We want to solve: $\hat{\beta} = \underset{\|\beta\|_0 \leq K}{\operatorname{argmin}} L(\mathbf{y}, \beta)$

$\rightarrow$ In non-robust approaches: $L(\mathbf{y}, \beta) = \|\mathbf{y} - \mathbf{X}\beta\|_2$.

$\rightarrow$ We can minimize Huber's criterion (our approach)

Robust sparse coding with Huber's criterion

Proposed approach: adapt the Normalized Iterative Hard Tresholding (NIHT)[BD10].

- We introduce the hard-thresholding operator:

$$H_K(\beta) = \begin{cases} \beta_i & \text{if } i \leq K \\ 0 & \text{otherwhise} \end{cases} \tag{13}$$

- At each iteration $\tilde{\beta}^{(n+1)}$ of the MM-algorithm optimization we add a final step

$$\beta^{(n+1)} = H_K(\tilde{\beta}^{(n+1)}).$$

$\rightarrow$ Convergence of the algorithm is noticed in practice. Theoretical study is under work (adapting results from [BD10].).

Examples of denoising and comparison: data

(a) Original image

(b) Noisy image

Parameters: $N = 81$, redundant 2D-DCT dictionary of size $N \times 256$ atoms, $c = 0.3529$.

Examples of denoising and comparison: results

(a) K=3, PSNR= 24.1285dB

(b) Noisy image

Examples of denoising and comparison: results

(a) K=4, PSNR= 25.1394dB

(b) Noisy image

Examples of denoising and comparison: results

(a) K=5, PSNR= 25.9906dB

(b) Noisy image

Examples of denoising and comparison: results

(a) K=6, PSNR= 26.7114dB

(b) Noisy image

Examples of denoising and comparison: results

(a) K=7, PSNR= 27.3306dB

(b) Noisy image

Examples of denoising and comparison: results

(a) K=8, PSNR= 27.8512dB

(b) Noisy image

Examples of denoising and comparison: results

(a) K=9, PSNR= 28.981dB

(b) Noisy image

Examples of denoising and comparison: results

(a) K=10, PSNR= 28.638dB

(b) Noisy image

Examples of denoising and comparison: results

(a) K=11, PSNR= 28.8801dB

(b) Noisy image

Examples of denoising and comparison: results

(a) K=12, PSNR= 28.9631dB

(b) Noisy image

Examples of denoising and comparison: results

(a) K=13, PSNR= 28.9572dB

(b) Noisy image

Examples of denoising and comparison: results

(a) K=14, PSNR= 28.8543dB

(b) Noisy image

Examples of denoising and comparison: results

(a) K=15, PSNR= 28.5781dB

(b) Noisy image

Examples of denoising and comparison: results

(a) K=17, PSNR= 27.894dB

(b) Noisy image

Examples of denoising and comparison: results

(a) K=20, PSNR= 26.3219dB

(b) Noisy image

Comparison to state other denoising approaches

(a) Noisy image
PSNR = 14,95 dB

(b) OMP
PSNR= 22.22 dB

(c) Proposed
PSNR = 28.96 dB

(d) Median 3×3
PSNR = 26.27 dB

(e) K-SVD
PSNR = 21.58 dB

(f) BM3D
PSNR = 24.17 dB

What's next?

A journal paper is under work. It will include:

- More examples ans applications (remote sensing).
- Tuning of parameters (c and K).
- Large extended discussion of dictionary learning applications for medical imaging.
- Matlab and python toolboxes.

Outline

Conclusion

References i

- P-A Absil, Robert Mahony, and Rodolphe Sepulchre, *Optimization algorithms on matrix manifolds*, Princeton University Press, 2009.
- T. Blumensath and M. E. Davies, *Normalized iterative hard thresholding: Guaranteed stability and performance*, IEEE Journal of Selected Topics in Signal Processing **4** (2010), no. 2, 298–309.
- Peter J. Huber, *Robust estimation of a location parameter*, Ann. Math. Statist. **35** (1964), no. 1, 73–101.
- E. Ollila and A. Mian, *Block-wise minimization-majorization algorithm for huber's criterion: Sparse learning and applications*.