Riemannian and information geometry in signal processing and machine learning

Part III: Riemannian geometry applied to machine learning

Florent Bouchard, Arnaud Breloy and **Ammar Mian**

Outline

1. **Introduction**
2. **Parameter on a manifold**
   - General context
   - Gaussian mixture models
   - Metric Learning
   - Deep learning optimization
3. **Data on a manifold**
   - General principles of using Riemannian Geometry
   - Tangent-space based approaches
   - Distance based approaches
   - More complex algorithms
4. **Numerical aspects and Toolboxes**

Outline

Riemannian geometry popularity in machine learning

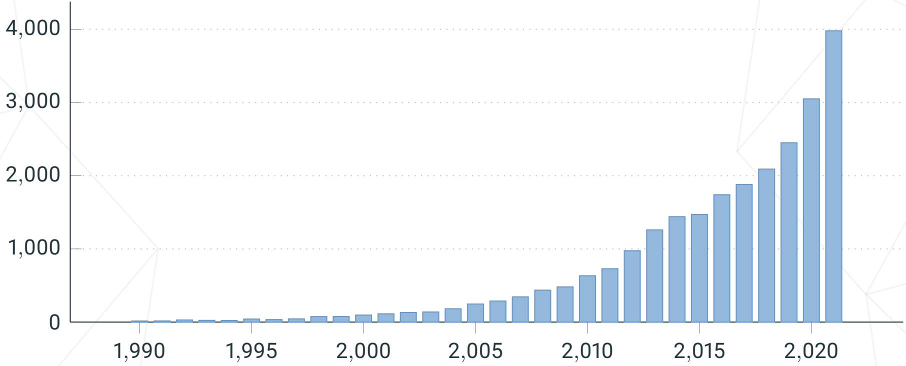


Figure 1: Number of articles with keyword **Riemannian machine learning** per year on Google scholar. Data obtained thanks to [Str18].

A machine learning taxonomy

	Supervised	Unsupervised
Data on a manifold	**Other** MDM [Bar+12b; Hip+21] KNN [MRO20] Boosting [TPM08b] Kernels [Jay+15] Geometric deep learning [Bro+17] [HG17] [HWV18] **Neural Network**	**Other** K-means [Col+21] Dictionary learning [HS15a] **Neural Network**
Parameter on a manifold	**Other** Metric Learning [Col+22] LDA Orthogonal weights [RMH18; Hua+18] Covariance pooling [Li+17] **Neural Network**	**Other** GMM [SH16] PCA [Bre+21] **Neural Network**

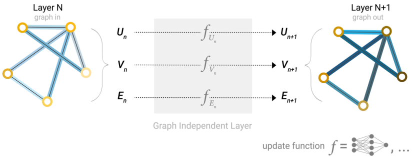
A machine learning taxonomy

	Supervised	Unsupervised
Data on a manifold	**Other** **MDM [Bar+12b; Hip+21]** **KNN [MRO20]** **Boosting [TPM08b]** **Kernels [Jay+15]** **Geometric deep learning [Bro+17] [HG17] [HWV18]** **Neural Network**	**Other** **K-means [Col+21]** Dictionary learning [HS15a] **Neural Network**
Parameter on a manifold	**Other** **Metric Learning [Col+22]** LDA **Orthogonal weights [RMH18; Hua+18]** **Covariance pooling [Li+17]** **Neural Network**	**Other** **GMM [SH16]** PCA [Bre+21] **Neural Network**

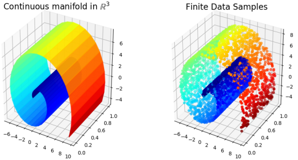
Other problems

What we don't talk about:

- Graph machine learning (geometric deep learning, etc) [Bro+17; Wu+20], `https://distill.pub/2021/gnn-intro/`



- Manifold learning [Ize12] `https://drewwilimitis.github.io/Manifold-Learning/`



Outline

Setting: supervised case

Suppose we have **euclidean** data points for which we consider a **supervised** problem:

- **Data:** $\Omega = \{(\mathbf{x}_k, y_k) \in \mathbb{R}^d \times \mathcal{C} : 1 \leq k \leq N\}$, where $\mathcal{C}$ is either a continuous (regression) or discrete space (classification).
- **Cost function:** $f(\{\hat{y}_k\}, \{y_k\}) : \mathcal{C}^N \times \mathcal{C}^N \mapsto \mathbb{R}$

Given a model $h_\theta : \mathbb{R}^d \mapsto \mathcal{C}$ parametrised by $\theta \in \mathcal{M}$, the training phase consists in solving:

$$\hat{\theta} = \underset{\theta \in \mathcal{M}}{\operatorname{argmin}} \quad f(\{h(y_k)\}, \{y_k\}). \tag{1}$$

Setting: supervised case

Suppose we have **euclidean** data points for which we consider a **supervised** problem:

- **Data:** $\Omega = \{(\mathbf{x}_k, y_k) \in \mathbb{R}^d \times \mathcal{C} : 1 \leq k \leq N\}$, where $\mathcal{C}$ is either a continuous (regression) or discrete space (classification).
- **Cost function:** $f(\{\hat{y}_k\}, \{y_k\}) : \mathcal{C}^N \times \mathcal{C}^N \mapsto \mathbb{R}$

Given a model $h_\theta : \mathbb{R}^d \mapsto \mathcal{C}$ parametrised by $\theta \in \mathcal{M}$, the training phase consists in solving:

$$\hat{\theta} = \underset{\theta \in \mathcal{M}}{\operatorname{argmin}} \quad f(\{h(y_k)\}, \{y_k\}). \tag{1}$$

Idea

When $\mathcal{M}$ is a Riemannian manifold, we can leverage the optimization framework developped in previous parts!

Setting: unsupervised case

Suppose we have **euclidean** data points for which we consider a **unsupervised** problem:

- **Data:** $\Omega = \{(\mathbf{x}_k) \in \mathbb{R}^d : 1 \leq k \leq N\}$
- **A target:** $\theta \in \mathcal{M}$. For example:
  K-means: $\theta = \{M_k : 1 \leq k \leq K\}$ Voronoï partitions,
  GMM: $\theta = \{(\alpha_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) : 1 \leq k \leq K\}$.
- **Cost function:** $f(\{\mathbf{x}_k\}, \theta) : \Omega \times \mathcal{M} \mapsto \mathbb{R}$

The learning task can be written as solving:

$$\hat{\theta} = \underset{\theta \in \mathcal{M}}{\operatorname{argmin}} \quad f(\{\mathbf{x}_k\}, \theta). \tag{2}$$

Setting: unsupervised case

Suppose we have **euclidean** data points for which we consider a **unsupervised** problem:

- **Data:** $\Omega = \{(\mathbf{x}_k) \in \mathbb{R}^d : 1 \leq k \leq N\}$
- **A target:** $\theta \in \mathcal{M}$. For example:
  K-means: $\theta = \{M_k : 1 \leq k \leq K\}$ Voronoï partitions,
  GMM: $\theta = \{(\alpha_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) : 1 \leq k \leq K\}$.
- **Cost function:** $f(\{\mathbf{x}_k\}, \theta) : \Omega \times \mathcal{M} \mapsto \mathbb{R}$

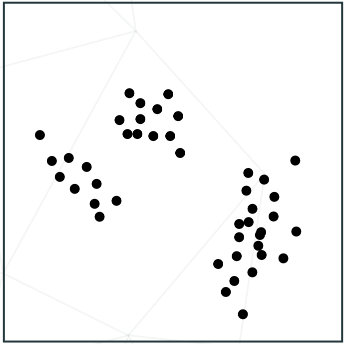
The learning task can be written as solving:

$$\hat{\theta} = \underset{\theta \in \mathcal{M}}{\operatorname{argmin}} \quad f(\{\mathbf{x}_k\}, \theta). \tag{2}$$

Idea

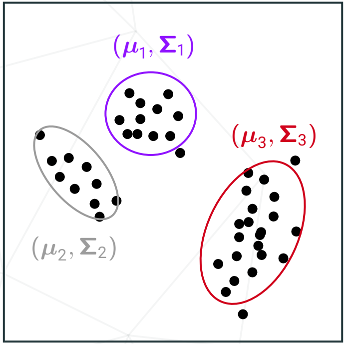
When $\mathcal{M}$ is a Riemannian manifold, we can leverage the optimization framework developped in previous parts!

Gaussian mixture models: the problem



Given data points $\Omega = \{(\mathbf{x}_k) \in \mathbb{R}^d : 1 \leq k \leq N\}$ and a number K of Gaussian mixtures, we want to estimate:

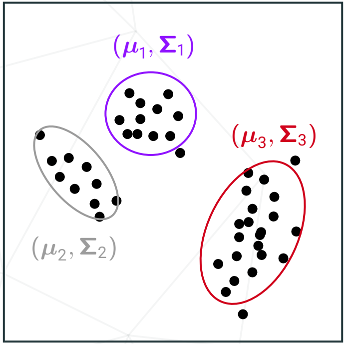
Gaussian mixture models: the problem



Given data points $\Omega = \{(\mathbf{x}_k) \in \mathbb{R}^d : 1 \leq k \leq N\}$ and a number K of Gaussian mixtures, we want to estimate:

$$\theta = \{(\alpha_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) : 1 \leq k \leq K\}$$

Gaussian mixture models: the problem



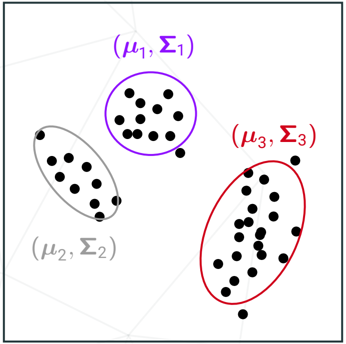
Given data points $\Omega = \{(\mathbf{x}_k) \in \mathbb{R}^d : 1 \leq k \leq N\}$ and a number K of Gaussian mixtures, we want to estimate:

$$\theta = \{(\alpha_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) : 1 \leq k \leq K\}$$

through minimisation of the negative log-likelihood:

$$\min_{\boldsymbol{\alpha} \in \Delta_K, \{\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j \succ 0\}_{j=1}^K} - \sum_{i=1}^n \log \left(\sum_{j=1}^K \alpha_j p_{\mathcal{N}} \left(\mathbf{x}_i; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j \right) \right) \quad (3)$$

Gaussian mixture models: the problem



Given data points $\Omega = \{(\mathbf{x}_k) \in \mathbb{R}^d : 1 \leq k \leq N\}$ and a number K of Gaussian mixtures, we want to estimate:

$$\theta = \{(\alpha_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) : 1 \leq k \leq K\}$$

through minimisation of the negative log-likelihood:

$$\min_{\boldsymbol{\alpha} \in \Delta_K, \{\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j \succ 0\}_{j=1}^K} -\sum_{i=1}^n \log \left(\sum_{j=1}^K \alpha_j p_{\mathcal{N}} \left(\boldsymbol{x}_i; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j \right) \right) \tag{3}$$

Problem

The cost function is not **g-convex** in the parameters. How to solve this problem through Riemannian optimization ?

Riemannian framework [HS15b]

It is possible to consider an alternate problem which is g-convex by using the following reparametrization:

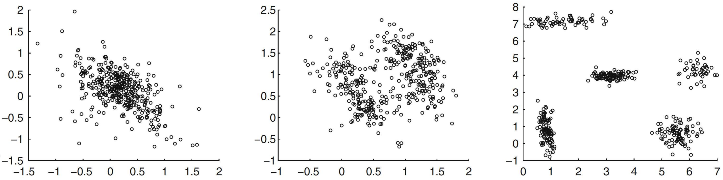
- $\mathbf{y}_i^{\mathrm{T}} = [\mathbf{x}_i^{\mathrm{T}} \quad 1]$
- $\boldsymbol{S}_k = \begin{pmatrix} \boldsymbol{\Sigma}_k + \boldsymbol{\mu}_k \boldsymbol{\mu}_k^{\mathrm{T}} & \boldsymbol{\mu}_k \\ \boldsymbol{\mu}_k^{\mathrm{T}} & 1 \end{pmatrix}$
- $q_{\mathcal{N}}(\mathbf{y}_i, \boldsymbol{S}) = \sqrt{2\pi} \exp(\frac{1}{2}) p_{\mathcal{N}}(\mathbf{y}_i; \mathbf{0}, \boldsymbol{S})$
- $\eta_k = \frac{\alpha_k}{\alpha_K}$ and having $\eta_0 = 0$

We then solve:

$$\max_{\{\boldsymbol{S}_j \succ 0\}_{j=1}^{K}, \{\eta_j\}_{j=1}^{K-1}} \widehat{\mathcal{L}}\left(\{\boldsymbol{S}_j\}_{j=1}^{K}, \{\eta_j\}_{j=1}^{K-1} \right) := \sum_{i=1}^{n} \log \left(\sum_{j=1}^{K} \frac{\exp(\eta_j)}{\sum_{k=1}^{K} \exp(\eta_k)} q_{\mathcal{N}}(\boldsymbol{y}_i; \boldsymbol{S}_j) \right) \tag{4}$$

on the product manifold $(\prod_{j=1}^{K} \mathcal{S}_d^+) \times \mathbb{R}^{K-1}$.

Some results [HS15b]



		EM ($e = 10$)		LBFGS ($e = 10$)		EM ($e = 1$)		LBFGS ($e = 1$)	
		Time (s)	ALL	Time (s)	ALL	Time (s)	ALL	Time (s)	ALL
$c = 0.2$	$K = 2$	1.1 ± 0.4	-10.7	5.6 ± 2.7	-10.7	65.7 ± 33.1	17.6	39.4 ± 19.3	17.6
	$K = 5$	30.0 ± 45.5	-12.7	49.2 ± 35.0	-12.7	365.6 ± 138.8	17.5	160.9 ± 65.9	17.5
$c = 1$	$K = 2$	0.5 ± 0.2	-10.4	3.1 ± 0.8	-10.4	6.0 ± 7.1	17.0	12.9 ± 13.0	17.0
	$K = 5$	104.1 ± 113.8	-13.4	79.9 ± 62.8	-13.3	40.5 ± 61.1	16.2	51.6 ± 39.5	16.2
$c = 5$	$K = 2$	0.2 ± 0.2	-11.0	3.4 ± 1.4	-11.0	0.2 ± 0.1	17.1	3.0 ± 0.5	17.1
	$K = 5$	38.8 ± 65.8	-12.8	41.0 ± 45.7	-12.8	17.5 ± 45.6	16.1	20.6 ± 22.5	16.1

Figure 2: Speed and average log-likelihood (ALL) comparisons for $d = 20$, exentricity $e = 10$, and $e = 1$. The numbers are averaged values for 20 runs over different sampled datasets

Metric Learning: problem

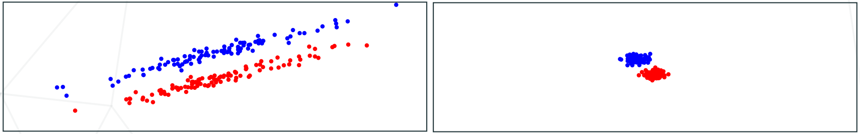
We consider here a supervised problem with K classes.
$\Omega = \{(\mathbf{x}_i, y_i) \in \mathbb{R}^d \times \{1, \ldots, K\} : 1 \leq i \leq N\}$.

Metric Learning Approach

Find a *Mahalanobis* distance

$$d_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^{\mathrm{T}} \mathbf{A}^{-1} (\mathbf{x}_i - \mathbf{x}_j)}, \tag{5}$$

that is relevant for the problem at hand.



$\{\mathbf{x}_i\}$ $\left\{\mathbf{A}^{-\frac{1}{2}} \mathbf{x}_i\right\}$

Geometric Mean Metric Learning [ZHS16]

Formulation

We consider the following objective function:

$$\hat{A} = \underset{\mathbf{A} \succ \mathbf{0}}{argmin} \sum_{(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{S}} d_{\boldsymbol{A}} \left(\boldsymbol{x}_i, \boldsymbol{x}_j \right) + \sum_{(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{D}} d_{\boldsymbol{A}^{-1}} \left(\boldsymbol{x}_i, \boldsymbol{x}_j \right), \tag{6}$$

where $\mathcal{S}$ is the set of all two samples with same class and $\mathcal{D}$ is the set of all two samples with different class.

Geometric Mean Metric Learning [ZHS16]

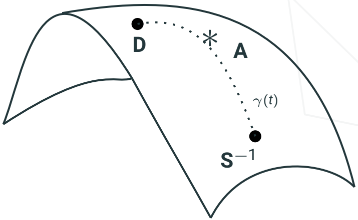
Formulation

We consider the following objective function:

$$\hat{A} = \underset{\mathbf{A} \succ \mathbf{0}}{argmin} \sum_{(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{S}} d_{\boldsymbol{A}}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) + \sum_{(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{D}} d_{\boldsymbol{A}^{-1}}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right), \tag{6}$$

where $\mathcal{S}$ is the set of all two samples with same class and $\mathcal{D}$ is the set of all two samples with different class.

In practice, we consider: $\min_{\mathbf{A} \succ \mathbf{0}} \quad \mathrm{tr}(\mathbf{AS}) + \mathrm{tr}(\mathbf{A}^{-1}\mathbf{D})$,
where $\boldsymbol{S} := \sum_{(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{S}} \left(\boldsymbol{x}_i - \boldsymbol{x}_j\right)\left(\boldsymbol{x}_i - \boldsymbol{x}_j\right)^T$ and
$\boldsymbol{D} := \sum_{(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{D}} \left(\boldsymbol{x}_i - \boldsymbol{x}_j\right)\left(\boldsymbol{x}_i - \boldsymbol{x}_j\right)^T$, which is solved thanks to a geometric mean:
$\boldsymbol{A} = \boldsymbol{S}^{-1} \sharp_{1/2} \boldsymbol{D} = \boldsymbol{S}^{-1/2}\left(\boldsymbol{S}^{1/2}\boldsymbol{D}\boldsymbol{S}^{1/2}\right)^{1/2}\boldsymbol{S}^{-1/2}$.



Robust Geometric Mean Learning: cf Antoine Collas

Metric learning

Find a *Mahalanobis* distance

$$d_{\boldsymbol{A}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{(\boldsymbol{x}_i - \boldsymbol{x}_j)^T \boldsymbol{A}^{-1} (\boldsymbol{x}_i - \boldsymbol{x}_j)} \tag{7}$$

relevant for classification problems.

Metric learning as covariance estimation

Proposed minimization problem:

$$\underset{(\boldsymbol{A}, \{\boldsymbol{A}_k\}) \in \left(\mathcal{S}_p^{++}\right)^{K+1}}{\text{minimize}} \quad \underbrace{\sum_{k=1}^{K} \pi_k \mathcal{L}_k(\boldsymbol{A}_k)}_{\text{negative log-likelihood}} + \lambda \underbrace{\sum_{k=1}^{K} \pi_k d^2_{\mathcal{S}_p^{++}}(\boldsymbol{A}, \boldsymbol{A}_k)}_{\substack{\text{cost function to compute} \\ \text{the center of mass of } \{\boldsymbol{A}_k\}}} \tag{8}$$

$\{\pi_k\}$ are the proportions of the classes and $\{\mathcal{L}_k\}$ are to be defined.

Robust Geometric Metric Learning (RGML)

Let $\boldsymbol{s}_{ki} = \boldsymbol{x}_l - \boldsymbol{x}_m$ where $\boldsymbol{x}_l, \boldsymbol{x}_m$ belong to the class k.

Gaussian negative log-likelihood

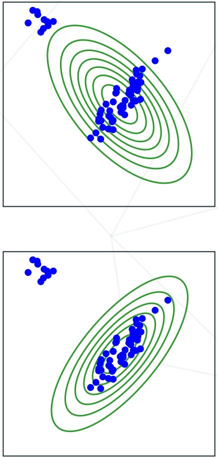
$$\mathcal{L}_{G,k}(\boldsymbol{A}_k) = \frac{1}{n_k} \sum_{i=1}^{n_k} \boldsymbol{s}_{ki}^T \boldsymbol{A}_k^{-1} \boldsymbol{s}_{ki} + \log |\boldsymbol{A}_k| \tag{9}$$

$$\text{minimized for } \boldsymbol{A}_k = \frac{1}{n_k} \sum_{i=1}^{n_k} \boldsymbol{s}_{ki} \boldsymbol{s}_{ki}^T \tag{10}$$

Tyler cost function

$$\mathcal{L}_{T,k}(\boldsymbol{A}_k) = \frac{p}{n_k} \sum_{i=1}^{n_k} \log \left(\boldsymbol{s}_{ki}^T \boldsymbol{A}_k^{-1} \boldsymbol{s}_{ki} \right) + \log |\boldsymbol{A}_k| \tag{11}$$

$$\text{minimized for } \boldsymbol{A}_k = \frac{1}{n_k} \sum_{i=1}^{n_k} \underbrace{\frac{p}{\boldsymbol{s}_{ki}^T \boldsymbol{A}_k^{-1} \boldsymbol{s}_{ki}}}_{\text{weight of sample } \boldsymbol{s}_{ki}} \boldsymbol{s}_{ki} \boldsymbol{s}_{ki}^T \tag{12}$$



Robust Geometric Metric Learning (RGML)

Riemannian metric

$\forall \xi = (\boldsymbol{\xi}, \{\boldsymbol{\xi}_k\}), \eta = (\boldsymbol{\eta}, \{\boldsymbol{\eta}_k\})$ in the tangent space

$$\langle \xi, \eta \rangle_{(\boldsymbol{A}, \{\boldsymbol{A}_k\})} = \mathrm{Tr}\left(\boldsymbol{A}^{-1}\boldsymbol{\xi}\boldsymbol{A}^{-1}\boldsymbol{\eta}\right) + \sum_{k=1}^{K} \mathrm{Tr}\left(\boldsymbol{A}_k^{-1}\boldsymbol{\xi}_k\boldsymbol{A}_k^{-1}\boldsymbol{\eta}_k\right) \quad (13)$$

$\Longrightarrow$ strongly geodesically convexity of the minimization problem

$\Longrightarrow$ the Riemannian gradient descent is fast

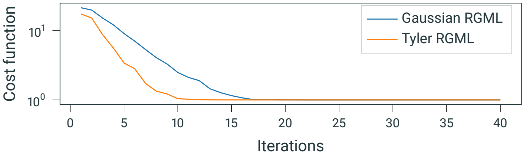


Figure 3: Cost function versus the iterations.

Robust Geometric Metric Learning (RGML)

RGML + *k-NN* on datasets from the UCI Machine Learning Repository

	Wine $p = 13$, $n = 178, K = 3$				Vehicle $p = 18, n = 846, K = 4$				Iris $p = 4, n = 150, K = 3$			
Method	Mislabeling rate				Mislabeling rate				Mislabeling rate			
	0%	5%	10%	15%	0%	5%	10%	15%	0%	5%	10%	15%
Euclidean	30.12	30.40	31.40	32.40	38.27	38.58	39.46	40.35	3.93	4.47	5.31	**6.70**
SCM	10.03	11.62	13.70	17.57	23.59	24.27	25.24	26.51	12.57	13.38	14.93	16.68
ITML - Identity	3.12	4.15	5.40	**7.74**	24.21	23.91	24.77	26.03	3.04	4.47	5.31	**6.70**
ITML - SCM	2.45	4.76	6.71	10.25	23.86	23.82	24.89	26.30	3.05	13.38	14.92	16.67
GMML	2.16	3.58	5.71	9.86	21.43	22.49	23.58	25.11	2.60	5.61	9.30	12.62
LMNN	4.27	6.47	7.83	9.86	20.96	24.23	26.28	28.89	3.53	9.59	11.19	12.22
Proposed - Gaussian	**2.07**	**2.93**	5.15	9.20	**19.76**	21.19	22.52	24.21	**2.47**	5.10	8.90	12.73
Proposed - Tyler	**2.12**	**2.90**	**4.51**	8.31	19.90	**20.96**	**22.11**	**23.58**	**2.48**	**2.96**	**4.65**	7.83

Table 1: Misclassification errors on 3 datasets: Wine, Vehicle and Iris. Mislabeling rate: percentage of labels randomly changed in the training set.

Github: `https://github.com/antoinecollas/robust_metric_learning`

Neural networks : a few definitions

Reconsidering the supervised case:

- **Data:** $\Omega = \{(\mathbf{x}_k, y_k) \in \mathbb{R}^d \times \mathcal{C} : 1 \leq k \leq N\}$, where $\mathcal{C}$ is either a continuous (regression) or discrete space (classification).
- **Cost function:** $f(\{\hat{y}_k\}, \{y_k\}) : \mathcal{C}^N \times \mathcal{C}^N \mapsto \mathbb{R}$

We consider a convolutional neural network model, where $h(\mathbf{x}; \theta)$ is a composition of multiple simple non-linear functions $h_l : \mathbb{R}^{d_l} \mapsto \mathbb{R}^{n_l}$ such that:

$$\mathbf{x}_l(x_l; \mathbf{W}_l, \mathbf{b}_l) = \varphi(\mathbf{W}_l \mathbf{x}_{l-1} + \mathbf{b}_l),$$

where $\mathbf{x}_{l-1} \in \mathbb{R}^{d_l}$ is the output of the previous hidden layer, $\mathbf{W}_l \in \mathbb{R}^{n_l \times d_l}$, $\mathbf{b} \in \mathbb{R}^{n_l}$ and φ is a non-linearity.

The learned parameters are then:

$$\theta = \{(\mathbf{W}_l, \mathbf{b}_l) : 1 \leq l \leq L\} \tag{14}$$

Orthogonal weights

Recent studies have considered regularizations on the weights parameters in order to improve convergence and stability of the training phase.

Among them, one approach is to restrain the weights to be orthogonal:

$$\mathbf{W}_l \in \mathcal{O}^{n_l, d_l} = \{\mathbf{W} \in \mathbb{R}^{n_l \times d_l} : \mathbf{W}\mathbf{W}^{\mathrm{T}} = \mathbf{I}_{n_l}\},$$

where $\mathcal{O}$ is the Stiefel manifold. This approach has shown to improve the problem of vashining or exploding gradient [Hua+18].

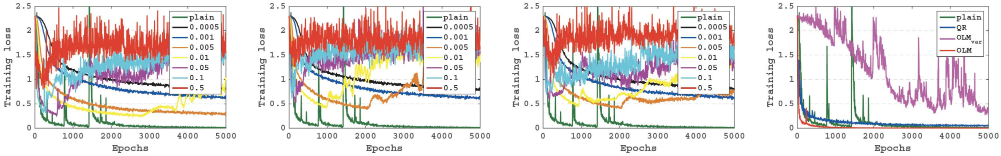
The training phase consists then in solving:

$$\hat{\theta} = \underset{\theta \in \mathcal{M}}{\operatorname{argmin}} \quad f(\{\mathbf{x}_k\}, \theta), \tag{15}$$

where $\mathcal{M} = \prod_{l=1}^{L} \left(\mathcal{O}_{n_l, d_l} \times \mathbb{R}^{n_l}\right)$.

Some numerical results [Hua+18]

In [Hua+18], the authors propose an alternative approach to Riemannian optimization by reparemtrising the problem into an equivalent euclidean problem which appears to be more stable:



(a) EI+QR (b) CI+QR (c) CayT (d) Our OLM

Figure 4: Results of training loss on MNIST dataset, with a Multilayer perceptron with 4 hidden layers. (a), (b) and (c) are Riemannian based approaches, (d) is the one proposed in the article.

$\rightarrow$ Optimization tuning can be difficult and Riemannian is not always the easiest. Another approach has also been proposed in [AP22].

Classification results [Hua+18]

Table 1: Test error (%) on VGG-style over CIFAR datasets. We report the 'mean $\pm std$' computed over 5 independent runs.

	CIFAR-10	CIFAR-100
plain	10.39 $\pm$ 0.14	36.02 $\pm$ 0.40
WN	10.29$\pm$ 0.39	34.66 $\pm$ 0.75
OLM-L2	10.06 $\pm$ 0.23	35.42 $\pm$ 0.32
OLM-L4	9.61 $\pm$ 0.23	33.66 $\pm$ 0.11
OLM	**8.61** $\pm$ 0.18	**32.58** $\pm$ 0.10

Table 2: Test error (%) on BN-Inception over CIFAR datasets. We report the 'mean $\pm std$' computed over 5 independent runs.

	CIFAR-10	CIFAR-100
plain	5.38$\pm$ 0.18	24.87 $\pm$ 0.15
WN	5.87$\pm$ 0.35	23.85 $\pm$ 0.28
OLM	**4.74$\pm$ 0.16**	**22.02** $\pm$ 0.13

Table 3: Test errors (%) of different methods on CIFAR-10 and CIFAR-100. For OLM, we report the 'mean $\pm std$' computed over 5 independent runs. 'WRN-28-10*' indicates the new results given by authors on their Github.

	CIFAR-10	CIFAR-100
pre-Resnet-1001	4.62	22.71
WRN-28-10	4.17	20.04
WRN-28-10*	3.89	18.85
WRN-28-10-OLM (ours)	**3.73** $\pm$ 0.12	18.76 $\pm$ 0.40
WRN-28-10-OLM-L1 (ours)	3.82 $\pm$ 0.19	**18.61** $\pm$ 0.14

Table 4: Top-5 test error (%, single model and single-crop) on ImageNet dataset.

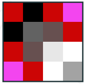
	AlexNet	BN-Inception	ResNet	Pre-ResNet
plain	20.91	12.5	9.84	9.79
OLM	**20.43**	**9.83**	**9.68**	**9.45**

Outline

The setup

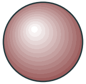
Suppose we have data points living in a Riemannian manifold:

SPD matrices $\mathbb{S}^p_+$



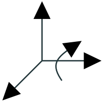
fMRI
Computer vision
SAR images

Hypersphere S_p



Geography
PolSAR

Rotations $SO(3)$



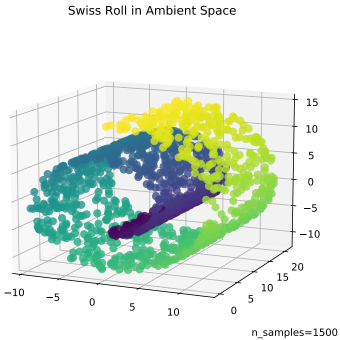
Camera position
3D objects

- **Data:** $\Omega = \{(\mathbf{x}_k, y_k) \in \mathcal{M} \times \mathcal{C} : 1 \leq k \leq N\}$, where $\mathcal{C}$ is either a continuous (regression) or discrete space (classification).
- **Cost function:** $f(\{\hat{y}_k\}, \{y_k\}) : \mathcal{C}^N \times \mathcal{C}^N \mapsto \mathbb{R}$

Problem

How to design a model h_θ taking into account the non-euclidean nature of the dataset?

Swiss-roll example



Solution 1: Map data to an euclidan space

Since many learning algorithms have already very fast and robust implementations, it would be interesting to map the data to an euclidean space in a way that preserves some notion about the distance on the manifold.

Idea

The tangent space around a point X is defined such that the geodesic between X and another point Y on the manifold ils the norm of the vector $\log_X(Y)$ according to the metric in the tangent space.

Solution 1: Map data to an euclidan space

Since many learning algorithms have already very fast and robust implementations, it would be interesting to map the data to an euclidean space in a way that preserves some notion about the distance on the manifold.

Idea

The tangent space around a point X is defined such that the geodesic between X and another point Y on the manifold ils the norm of the vector $\log_X(Y)$ according to the metric in the tangent space.

$\rightarrow$ We may be able to approximate geodesics distance in a Euclidean space thanks to the tangent space tool!

Solution 1: Map data to an euclidan space

Since many learning algorithms have already very fast and robust implementations, it would be interesting to map the data to an euclidean space in a way that preserves some notion about the distance on the manifold.

Idea

The tangent space around a point X is defined such that the geodesic between X and another point Y on the manifold ils the norm of the vector $\log_X(Y)$ according to the metric in the tangent space.

→ We may be able to approximate geodesics distance in a Euclidean space thanks to the tangent space tool!
→ We need the data points to belong to the same tangent-space to make it work, otherwise we would just compute euclidean distances in the embedded space.

Solution 1: Map data to an euclidan space

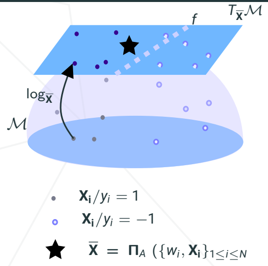
Since many learning algorithms have already very fast and robust implementations, it would be interesting to map the data to an euclidean space in a way that preserves some notion about the distance on the manifold.

Idea

The tangent space around a point X is defined such that the geodesic between X and another point Y on the manifold ils the norm of the vector $\log_X(Y)$ according to the metric in the tangent space.

→ We may be able to approximate geodesics distance in a Euclidean space thanks to the tangent space tool!
→ We need the data points to belong to the same tangent-space to make it work, otherwhise we would just compute euclidean distances in the embedded space.
→ Which point would be most suitable ?

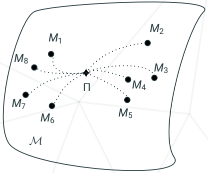
The geodesic mean



Consider a model $f : \mathbb{R}^d \mapsto \mathcal{C}$, we can adapt it through:

$$f \circ \log_{\Pi_A(\{1/N, \mathbf{X}_i\}_{1\leq i\leq N})} : \mathcal{M} \to \mathcal{C}. \qquad (16)$$

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



$$\Pi\ (\{\alpha_i, \mathbf{M}_i\}_{1\leq i\leq N}) = \underset{\mathbf{M}\in\mathcal{M}}{\operatorname{argmin}} \sum_{i=1}^{N} \alpha_i d^2(\mathbf{M}, \mathbf{M}_i). \qquad (17)$$

Solution 2: Adapt algorithms to use geodesic distances as a similarity measure

- **Case 1**: Distance and mean based algorithms (KNN, MDM, K-means)
  $\rightarrow$ Replace the euclidean metric by a Riemannian one
  $\rightarrow$ Replace the euclidean mean by a Riemannian mean

- **Case 2**: More complex algorithms (EM, Kernels, Neural networks, etc)
  $\rightarrow$ Case by case adaptation necessary

Approaches when data is on a manifold

Solution 1: Tangent space mapping

Anything Goes

- SVM: p.30
- Logistic Regression: p.32
- MLP: p.30
- Logitboost: p.35

Solution 2: Account for the true distances

distance-based algorithm: p. 38

- KNN
- MDM
- K-means

more complex algorithms

- GMM: p.58
- Kernels: p.54
- PCA: p.66
- Neural Networks p.59

Approaches when data is on a manifold

Solution 1: Tangent space mapping

Anything Goes

SVM: p.30
Logistic Regression: p.32
MLP: p.30
Logitboost: p.35

Solution 2: Account for the true distances

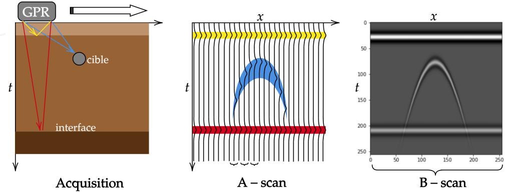
distance-based algorithm: p. 38

- KNN
- MDM
- K-means

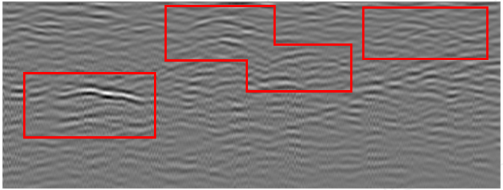
more complex algorithms

- GMM: p.58
- Kernels: p.54
- PCA: p.66
- Neural Networks p.59

Example: GPR classification [Gal+22]

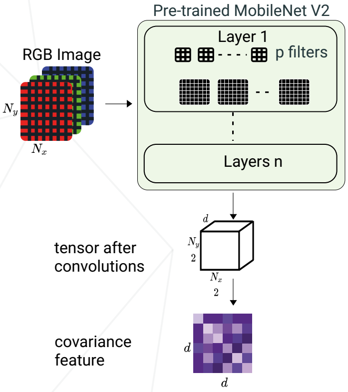


(a) Diagram of the acquisition of a radargram

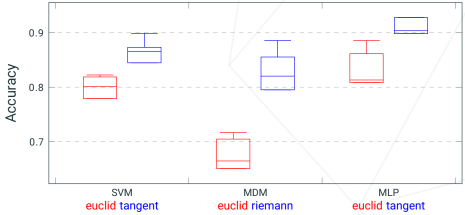


(b) Illustration of a GPR image

Methodology



(a) Covariance feature extraction



(b) Classification results

Example: Pedestrian detection [MRO20]

Datasets

- INRIA person [DT05]: 3548 positive and 1212 negative images
- DaimerChrysler pedestrian dataset [MG06]: 24500 positive and 24000 negative images

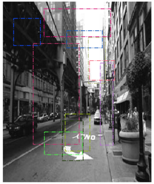
Code

- Available at:
  `https://github.com/AmmarMian/Comparative_study_pedestrian_Eusipco`
- Built on top of *scikit-learn* [**scikit-learn**] and pyRiemann package [Bar+12a]

Methodology

|I|

|I|



Features:

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

Approach

- Random sampling of windows with low overlap for each positive and negative images
- 10 windows for INRIA dataset, and 2 windows for DC
- 4-fold cross validation for INRIA and 3-fold for DC

Results

	Fold 1	Fold 2	Fold 3	Fold 4	mean
Euclidean LogisticRegression	0.831	0.831	0.832	0.831	0.831
Riemannian LogisticRegression	0.741	0.709	0.719	0.685	0.714

Table 2: Results on INRIA dataset

	Fold 1	Fold 2	Fold 3	mean
Euclidean LogisticRegression	0.700	0.702	0.700	0.701
Riemannian LogisticRegression	0.733	0.736	0.735	0.735

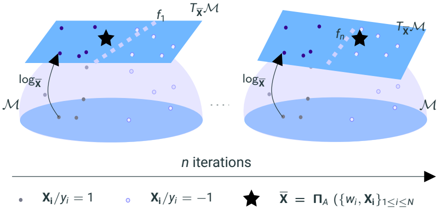
Table 3: Results on DaimerChrysler dataset

Another approach: boosting [TPM08a]

Classification based on combining weak-learners (decision trees) $\{f_l : 1 \leq l \leq L\}$ into a classifier with the form $\mathrm{sign}[F(\mathbf{x})] = \mathrm{sign}[\frac{1}{2}\sum_{l=1}^{L} f_l(\mathbf{x})]$. The probability for feature vector $\mathbf{x}$ of being in class 1 is represented by:

$$p(\mathbf{x}) = \frac{\exp\left(F(\mathbf{x})\right)}{\exp\left(F(\mathbf{x})\right) + \exp\left(-F(\mathbf{x})\right)}, \tag{18}$$

Riemannian equivalent:



Results

	Fold 1	Fold 2	Fold 3	Fold 4	mean
Euclidean LogisticRegression	0.831	0.831	0.832	0.831	0.831
Riemannian LogisticRegression	0.741	0.709	0.719	0.685	0.714
Euclidean Logitboost	0.934	0.931	0.933	0.935	0.933
Riemannian Logitboost	0.948	0.947	0.946	0.950	0.948

Table 4: Results on INRIA dataset

	Fold 1	Fold 2	Fold 3	mean
Euclidean LogisticRegression	0.700	0.702	0.700	0.701
Riemannian LogisticRegression	0.733	0.736	0.735	0.735
Euclidean logitboost	0.730	0.734	0.729	0.731
Riemannian logitboost	0.741	0.745	0.738	0.741

Table 5: Results on DaimerChrysler dataset

Approaches when data is on a manifold

Solution 1: Tangent space mapping

Anything Goes

SVM: p.30
Logistic Regression: p.32
MLP: p.30
Logitboost: p.35

Solution 2: Account for the true distances

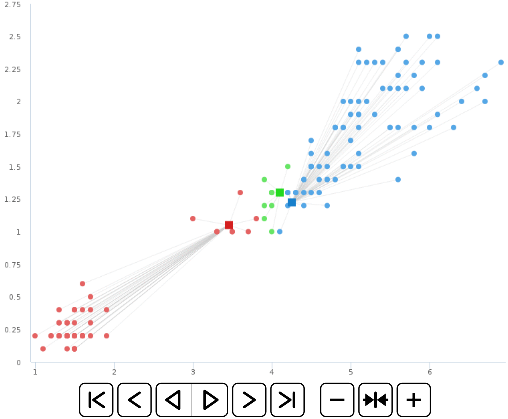
distance-based algorithm: p. 38

KNN
MDM
K-means

more complex algorithms

GMM: p.58
Kernels: p.54
PCA: p.66
Neural Networks p.59

KNN, MDM and K-means



→ We need a descriptive feature and a distance on the feature space !

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

Results on DaimerChrysler dataset

		Fold 1	Fold 2	Fold 3	mean
Euclidean	RBF SVM	0.726	0.727	0.727	0.727
	logitboost	0.730	0.734	0.729	0.731
	KNN	0.710	0.708	0.711	0.710
	MDM	0.592	0.590	0.591	0.591
	LogisticRegression	0.700	0.702	0.700	0.701
Riemannian	RBF SVM	**0.814**	**0.814**	**0.814**	**0.814**
	logitboost	0.741	0.745	0.738	0.741
	KNN	0.727	0.723	0.727	0.726
	MDM	0.638	0.636	0.638	0.638
	LogisticRegression	0.733	0.736	0.735	0.735

Example: EEG signals classification

See paper presentation: **”Riemannian classification of EEG signals with missing values”** on Wednesday !

Example: Classification of multispectral satellite images

In recent years, many image time series have been taken from the **earth** with different technologies: **SAR, multi/hyper spectral imaging**, ...

Objective

Segment semantically these data using **spatial** information, **temporal** information and **sensor diversity** (spectral bands, polarization...).

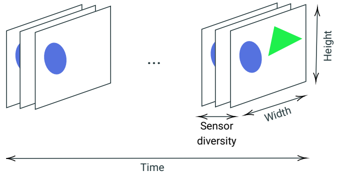


Figure 7: Multivariate image time series.

Applications

Disaster assessment, activity monitoring, land cover mapping, crop type mapping, ...

Example of a hyperspectral image

Indian pines dataset:

145 $\times$ 145 pixels, 200 spectral bands,
16 classes (corn, grass, wood, ...).

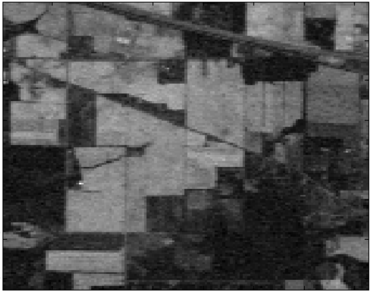


Figure 8: Raw image.

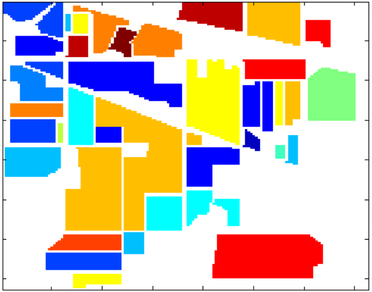


Figure 9: Segmented image, one color = one class.

Example of multi-spectral time series

Breizhcrops dataset[1]:

more than 600 000 crop time series across the whole Brittany,
13 spectral bands, 9 classes.

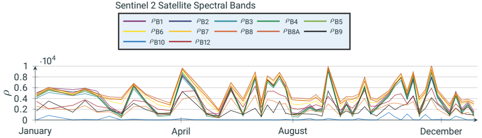


Figure 10: Reflectances ρ of a time series of **meadows**.

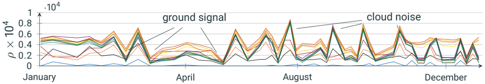


Figure 11: Reflectances ρ of a time series of **corn**.

[1] https://breizhcrops.org/

Clustering/classification pipeline and Riemannian geometry

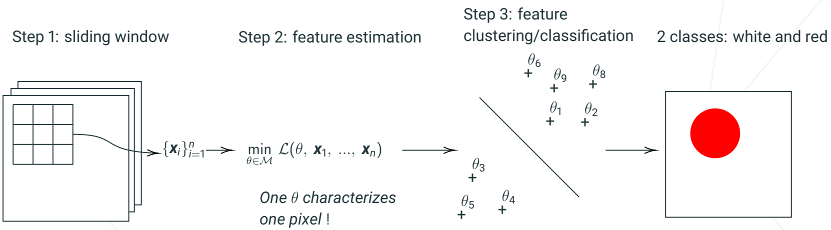


Figure 12: Clustering/classification pipeline.

Examples of θ:

$\theta = \boldsymbol{\Sigma}$ a covariance matrix, $\theta = (\boldsymbol{\mu}, \boldsymbol{\Sigma})$ a vector and a covariance matrix, $\theta = (\{\tau_i\}, \boldsymbol{U})$ a scalar and an orthogonal matrix...

Clustering/classification pipeline and Riemannian geometry

Clustering/classification and Riemannian geometry

$\theta \in \mathcal{M}$, a *Riemannian manifold* (contraints and non-constant metric):

step 2: minimization of $\mathcal{L}$ over $\mathcal{M}$,

step 3: computing distances and centers of mass on $\mathcal{M}$.

Existing work (*e.g.* in BCI classification)

$\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n \in \mathbb{R}^p$ realizations of $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma})$, $\boldsymbol{\Sigma} \in \mathcal{S}_p^{++}$.
Step 2: maximum likelihood estimator:

$$\theta = \hat{\boldsymbol{\Sigma}} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x}_i \boldsymbol{x}_i^T. \tag{19}$$

Step 3: Riemannian distance on $\mathcal{S}_p^{++}$ (geodesic distance):

$$d_{\mathcal{S}_p^{++}}(\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2) = \left\| \log \left(\boldsymbol{\Sigma}_1^{-\frac{1}{2}} \boldsymbol{\Sigma}_2 \boldsymbol{\Sigma}_1^{-\frac{1}{2}} \right) \right\|_2. \tag{20}$$

Study of a ”low rank” statistical model

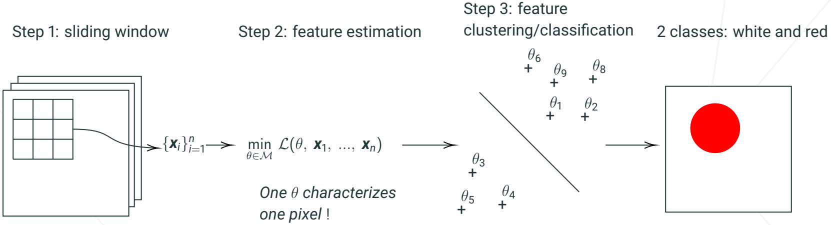


Figure 13: Clustering/classification pipeline.

Statistical model

$\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n \in \mathbb{R}^p, \forall k < p$:

$$\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \tau_i \boldsymbol{U}\boldsymbol{U}^T + \boldsymbol{I}_p) \tag{21}$$

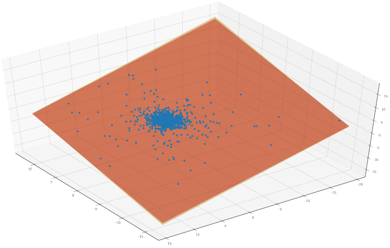
with $\tau_i > 0$ and $\boldsymbol{U} \in \mathbb{R}^{p \times k}$ is an orthogonal basis ($\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I}_k$).
Goal: estimate and classify $\theta = (\boldsymbol{U}, \boldsymbol{\tau})$.

Study of a ”low rank” statistical model

Statistical model

$$\underbrace{\mathbf{x}_i}_{\in\mathbb{R}^p} \overset{d}{=} \underbrace{\sqrt{\tau_i}\boldsymbol{U}\boldsymbol{g}_i}_{\text{signal}\in\text{span}(\boldsymbol{U})} + \underbrace{\boldsymbol{n}_i}_{\text{noise}\in\mathbb{R}^p} \quad (22)$$

where $\boldsymbol{g}_i \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_k)$ and $\boldsymbol{n}_i \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_p)$ are independent, $\boldsymbol{\tau} \in (\mathbb{R}_*^+)^n$, and $\boldsymbol{U} \in \mathbb{R}^{p\times k}$ is an orthogonal basis ($\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I}_k$).



Study of a "low rank" statistical model: estimation

Maximum likelihood estimation (MLE)

Minimization of the negative log-likelihood with constraints:

- $\boldsymbol{U} \in \mathrm{Gr}_{p,k}$: orthogonal basis of the subspace (and thus invariant by rotation !)
- $\boldsymbol{\tau} \in (\mathbb{R}_*^+)^n$: positivity constraints

$$\underset{(\boldsymbol{U},\boldsymbol{\tau}) \in \mathrm{Gr}_{p,k} \times (\mathbb{R}_*^+)^n}{\text{minimize}} \ \mathcal{L}(\boldsymbol{U}, \boldsymbol{\tau}) \tag{23}$$

Study of a "low rank" statistical model: estimation

Fisher information metric

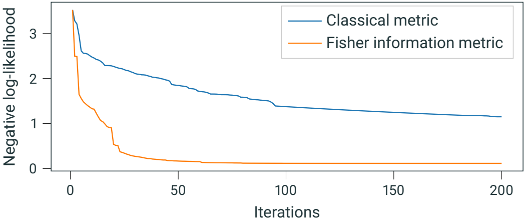
$\forall \xi = (\boldsymbol{\xi}_{\boldsymbol{U}}, \boldsymbol{\xi}_{\boldsymbol{\tau}}), \eta = (\boldsymbol{\eta}_{\boldsymbol{U}}, \boldsymbol{\eta}_{\boldsymbol{\tau}})$ in the tangent space

$$\langle \xi, \eta \rangle^{\text{FIM}}_{(\boldsymbol{U},\boldsymbol{\tau})} = \mathbb{E}\left[\mathrm{D}\,\mathcal{L}(\theta)[\xi]\,\mathrm{D}\,\mathcal{L}(\theta)[\eta]\right] \tag{24}$$

$$= 2nc_{\boldsymbol{\tau}}\,\mathrm{Tr}\left(\boldsymbol{\xi}_{\boldsymbol{U}}^T \boldsymbol{\eta}_{\boldsymbol{U}}\right) + k\left(\boldsymbol{\xi}_{\boldsymbol{\tau}} \odot (\mathbf{1} + \boldsymbol{\tau})^{\odot -1}\right)^T \left(\boldsymbol{\eta}_{\boldsymbol{\tau}} \odot (\mathbf{1} + \boldsymbol{\tau})^{\odot -1}\right), \tag{25}$$

where $c_{\boldsymbol{\tau}} = \frac{1}{n}\sum_{i=1}^{n} \frac{\tau_i^2}{1+\tau_i}$.

To solve (23) : Riemannian gradient descent on $\left(\mathrm{Gr}_{p,k} \times (\mathbb{R}_*^+)^n, \langle ., . \rangle^{\text{FIM}}_{.}\right)$.



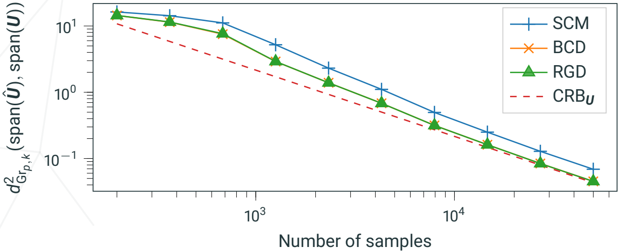
Study of a ”low rank” statistical model: bounds

Intrinsic Cramér-Rao bounds

Study of the performance through intrinsic Cramér-Rao bounds:

$$\overbrace{\mathbb{E}[d^2_{\mathrm{Gr}_{p,k}}(\mathrm{span}(\hat{\boldsymbol{U}}), \mathrm{span}(\boldsymbol{U}))]}^{\text{subspace estimation error}} \geq \frac{(p-k)k}{nc_{\boldsymbol{\tau}}} \approx \frac{(p-k)k}{n \times \mathrm{SNR}} \tag{26}$$

$$\underbrace{\mathbb{E}[d^2_{(\mathbb{R}^+_*)^n}(\hat{\boldsymbol{\tau}}, \boldsymbol{\tau})]}_{\text{texture estimation error}} \geq \frac{1}{k} \sum_{i=1}^{n} \frac{(1+\tau_i)^2}{\tau_i^2} \tag{27}$$



Study of a "low rank" statistical model: *K-means++*

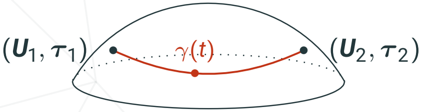


Figure 16: Distance.

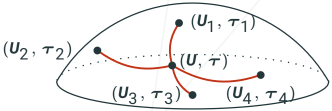


Figure 17: Center of mass (U, τ).

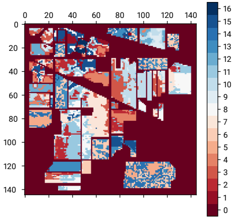


Figure 18: Euclidean *K-means++*: OA $= 31.2\%$.

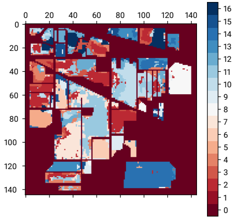


Figure 19: Proposed *K-means++*: OA $= 47.2\%$.

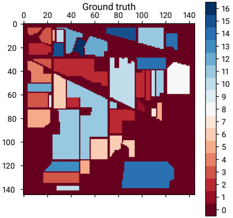


Figure 20: Ground truth.

Approaches when data is on a manifold

Solution 1: Tangent space mapping

Anything Goes

- SVM: p.30
- Logistic Regression: p.32
- MLP: p.30
- Logitboost: p.35

Solution 2: Account for the true distances

distance-based algorithm: p. 38

- KNN
- MDM
- K-means

more complex algorithms

- **GMM: p.58**
- **Kernels: p.54**
- **PCA: p.66**
- **Neural Networks p.59**

Kernels on Riemannian manifold [JHS16]

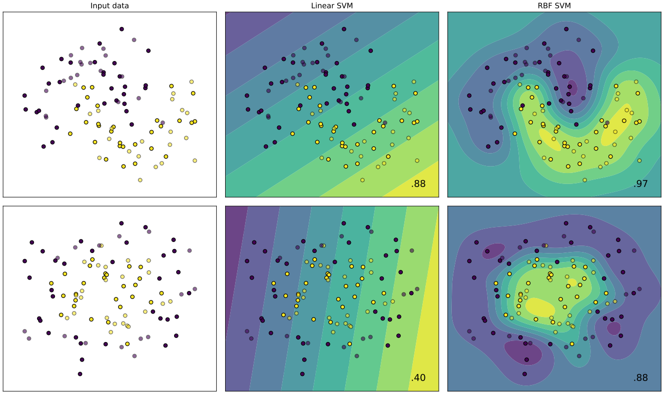


Figure 21: From scikit-learn documentation

Traditional RBF kernel to Riemannian RBF

Definition

Let $\mathcal{X}$ be a nonempty set and $f : (\mathcal{X}, \mathcal{X}) \to \mathbb{R}$ be a kernel. The kernel $\exp(-\gamma f(\mathbf{x}, \mathbf{y}))$ is positive definite for all $\gamma > 0$ if and only f is negative definite.

Usually on euclidean spaces, $f(x, y) = \|\mathbf{x} - \mathbf{y}\|_2$, but one idea is to replace it by a geodesic distance and have a Kernel:

$$\exp(-\gamma d^2(\mathbf{x}, \mathbf{y})).$$

Which conditions to have positive definiteness of the kernel?

Riemannian RBF kernel [JHS16]

Theorem

Let (M, d) be a metric space and define $k : (M \times M) \mapsto \mathbb{R}$ by $k(x, y) = \exp(-\gamma d^2(x, y))$. Then, k is a positive definite kernel for all $\gamma > 0$ if and only there exists an inner product space $\mathcal{V}$ and a function $\phi : M \mapsto \mathcal{V}$ such that $d(x, y) = \|\phi(x) - \phi(y)\|_{\mathcal{V}}$.

$\rightarrow$ Thus depening on the geodesic distance, it isn't guaranteed to define a positive definite kernel !

Known kernels on $\mathcal{S}_d^+$ and $G_{n,r}$

Metric Name	Formula	Geodesic Distance	Positive Definite Gaussian Kernel for all $\gamma > 0$
Log-Euclidean	$\lVert \log(\mathbf{S}_1) - \log(\mathbf{S}_2) \rVert_F$	Yes	Yes
Affine-Invariant	$\left\lVert \log\left(\mathbf{S}_1^{-1/2}\mathbf{S}_2\mathbf{S}_1^{-1/2}\right)\right\rVert_F$	Yes	No
Cholesky	$\lVert \mathrm{chol}(\mathbf{S}_1) - \mathrm{chol}(\mathbf{S}_2) \rVert_F$	No	Nes
Power-Euclidean	$\frac{1}{\alpha}\lVert \mathbf{S}_1^\alpha - \mathbf{S}_2^\alpha \rVert_F$	No	Yes
Root Stein Divergence	$\left[\log\det\left(\frac{1}{2}\mathbf{S}_1 + \frac{1}{2}\mathbf{S}_2\right) - \frac{1}{2}\log\det(\mathbf{S}_1\mathbf{S}_2)\right]^{1/2}$	No	No

Table 6: RBF Kernels for different metric on $\mathcal{S}_d^+$.

Metric Name	Formula	Geodesic Distance	Positive Definite Gaussian Kernel for all $\gamma > 0$
Projection	$2^{-1/2}\lVert Y_1Y_1^T - Y_2Y_2^T \rVert_F = \left(\sum_i \sin^2\theta_i\right)^{1/2}$	No	Yes
Arc length	$\left(\sum_i \theta_i^2\right)^{1/2}$	Yes	No
Fubini-Study	$\arccos\left\lvert\det\left(Y_1^TY_2\right)\right\rvert = \arccos\left(\prod_i \cos\theta_i\right)$	No	No
Chordal 2-norm	$\lVert Y_1U - Y_2V \rVert_2 = 2\max_i \sin\frac{1}{2}\theta_i$	No	No
Chordal F-norm	$\lVert Y_1U - Y_2V \rVert_F = 2\left(\sum_i \sin^2\frac{1}{2}\theta_i\right)^{1/2}$	No	No

Table 7: RBF Kernels on $G_{n,r}$. Here, USV^{T} is the singular value decomposition of $Y1^{\mathrm{T}}Y_2$, whereas θ_is are the the principal angles between the two subspaces $[Y_1]$ and $[Y_2]$.

GMM on a Riemannian manifold

Problems

- How to define the concept of a Gaussian distribution on a Riemannian manifold?
- Extend it to the mixture model?
- Develop an EM like algorithm on the manifold?

For SPD matrices $\mathcal{S}_d^+$, see [Sai+17].

Neural networks on Riemannian manifold

Let us now reconsider Neural Networks. How do we adapt classic networks architecture to consider input data on a Riemannian manifold $\mathcal{M}$?

Neural networks on Riemannian manifold

Let us now reconsider Neural Networks. How do we adapt classic networks architecture to consider input data on a Riemannian manifold $\mathcal{M}$?

- Definition of a convolution?

Neural networks on Riemannian manifold

Let us now reconsider Neural Networks. How do we adapt classic networks architecture to consider input data on a Riemannian manifold $\mathcal{M}$?

- Definition of a convolution?
- Non-linearity?

Neural networks on Riemannian manifold

Let us now reconsider Neural Networks. How do we adapt classic networks architecture to consider input data on a Riemannian manifold $\mathcal{M}$?

- Definition of a convolution?
- Non-linearity?
- Dense layers?

SPDnet Architecture [huang2017riemannian]

One layer of the SPDnet architecture:

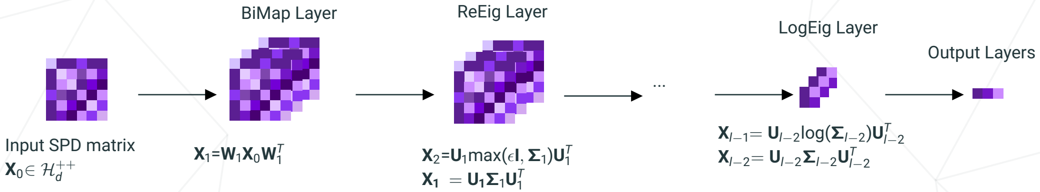


Figure 22: SPD layers

Forward 1/2

- BiMap Layer (to generate more compact and discriminative SPD matrices):

$$\mathbf{X}_k = f_b^{(k)}(\mathbf{X}_{k-1}; \mathbf{W}_k) = \mathbf{W}_k \mathbf{X}_{k-1} \mathbf{W}_k^T$$

  $\mathbf{W}_k$ is the Stiefeld manifold $St(d_k, d_{k-1})$ with $d_k < d_{k-1}$

- ReEig Layer. First we compute the EVD of $\mathbf{X}_{k-1}$:

$$\mathbf{X}_{k-1} = \mathbf{U}_{k-1} \mathbf{\Sigma}_{k-1} \mathbf{U}_{k-1}^T$$

  and then

$$\mathbf{X}_k = f^r(\mathbf{X}_{k-1}) = \mathbf{U}_{k-1} \max(\epsilon \mathbf{I}, \mathbf{\Sigma}_{k-1}) \mathbf{U}_{k-1}^T$$

Forward 2/2

- LogEig Layer. We recall that:

$$\mathbf{X}_{k-1} = \mathbf{U}_{k-1}\boldsymbol{\Sigma}_{k-1}\mathbf{U}_{k-1}^T$$

  and then

$$\mathbf{X}_k = f^l(\mathbf{X}_{k-1}) = \log(\mathbf{X}_{k-1}) = \mathbf{U}_{k-1}\log(\boldsymbol{\Sigma}_{k-1})\mathbf{U}_{k-1}^T$$

  where $\log(\boldsymbol{\Sigma}_{k-1})$ is the diagonal matrix of eigenvalue logarithms.
- Other layers: at the end, FC layer could be inserted. Moreover, the two first layers could be repeated several times.

Back-Propagation 1/2

- Principle of chain rule:

$$
\begin{aligned}
\frac{\partial L^{(k)}(\mathbf{X}_{k-1}, y)}{\partial \mathbf{W}_k} &= \frac{\partial L^{(k+1)}(\mathbf{X}_k, y)}{\partial \mathbf{X}_k} \frac{\partial f^{(k)}(\mathbf{X}_{k-1})}{\partial \mathbf{W}_k} \\
\frac{\partial L^{(k)}(\mathbf{X}_{k-1}, y)}{\partial \mathbf{X}_{k-1}} &= \frac{\partial L^{(k+1)}(\mathbf{X}_k, y)}{\partial \mathbf{X}_k} \frac{\partial f^{(k)}(\mathbf{X}_{k-1})}{\partial \mathbf{X}_{k-1}}
\end{aligned}
$$

- Update of the weights $\mathbf{W}_k$ by a Riemannian gradient descent on Stifield:

$$
\begin{aligned}
\tilde{\Delta} L^{(k)}_{\mathbf{W}_k^t} &= \Delta L^{(k)}_{\mathbf{W}_k^t} - \Delta L^{(k)}_{\mathbf{W}_k^t} (\mathbf{W}_k^t)^T \mathbf{W}_k^t \\
\mathbf{W}_k^{t+1} &= \Gamma(\mathbf{W}_k^t - \lambda \tilde{\Delta} L^{(k)}_{\mathbf{W}_k^t})
\end{aligned}
$$

where Γ is the retraction operator on Stiefield and λ is the learning rate and $\Delta L^{(k)}_{\mathbf{W}_k^t}$ is the euclidean gradient:

$$
\Delta L^{(k)}_{\mathbf{W}_k^t} = 2 \frac{\partial L^{(k+1)}(\mathbf{X}_k, y)}{\partial \mathbf{X}_k} \mathbf{W}_k^t \mathbf{X}_{k-1}
$$

Back-Propagation 2/2

- For ReEig and LogEig, results come from [IVS15]:

$$\frac{\partial L^{(k)}(\mathbf{X}_{k-1}, y)}{\partial \mathbf{X}_{k-1}} = 2\mathbf{U}\left(\mathbf{P}^T \circ \left(\mathbf{U}^T \frac{\partial L^{(k')}}{\partial \mathbf{U}}\right)_{sym}\right)\mathbf{U}^T + \mathbf{U}\left(\frac{\partial L^{(k')}}{\partial \mathbf{\Sigma}}\right)_{diag}\mathbf{U}^T$$

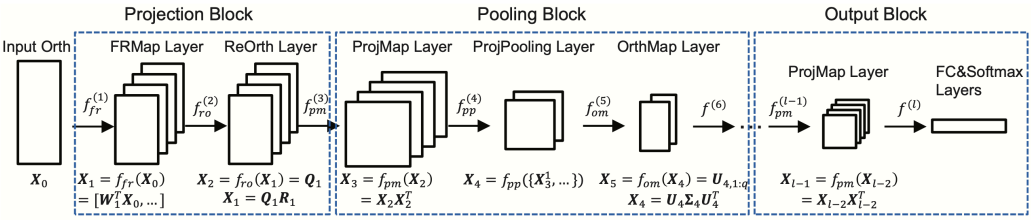
- For ReEig:

$$\begin{aligned} \frac{\partial L^{(k')}}{\partial \mathbf{U}} &= 2\left(\frac{\partial L^{(k+1)}}{\partial \mathbf{X}_k}\right)_{sym} \mathbf{U}\max(\epsilon\mathbf{I}, \mathbf{\Sigma}) \\ \frac{\partial L^{(k')}}{\partial \mathbf{\Sigma}} &= \mathbf{Q}\mathbf{U}^T\left(\frac{\partial L^{(k+1)}}{\partial \mathbf{X}_k}\right)_{sym} \mathbf{U} \end{aligned}$$

- For LogEig:

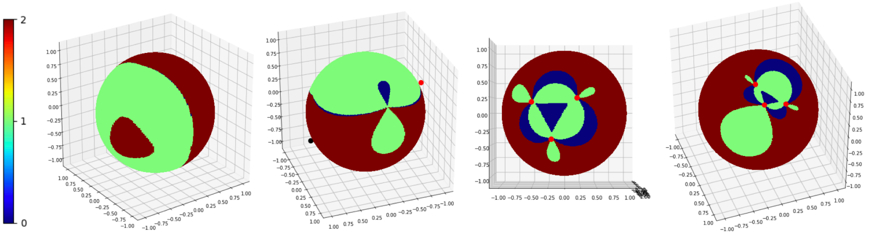
$$\begin{aligned} \frac{\partial L^{(k')}}{\partial \mathbf{U}} &= 2\left(\frac{\partial L^{(k+1)}}{\partial \mathbf{X}_k}\right)_{sym} \mathbf{U}\log(\mathbf{\Sigma}) \\ \frac{\partial L^{(k')}}{\partial \mathbf{\Sigma}} &= \mathbf{\Sigma}^{-1}\mathbf{U}^T\left(\frac{\partial L^{(k+1)}}{\partial \mathbf{X}_k}\right)_{sym} \mathbf{U} \end{aligned}$$

Grassman manifold approach [HWV18]



PCA on a Riemannian manifold? Geodesic PCA

Intricate problem. Some advances towards it in [Pen16] thanks to the notion of barycenters.



Outline

Numerical ressources

- Matlab: https://www.manopt.org
- Python:

  Riemannian geometry:

    https://geomstats.github.io/

    pyRiemann https://pyriemann.readthedocs.io/en/latest/

  Optimization:

    https://pymanopt.org

    https://geoopt.readthedocs.io/en/latest/

    https://github.com/mctorch/mctorch

  Autodifferentiation:

    pytorch, tensorflow

    https://github.com/HIPS/autograd

    https://jax.readthedocs.io/en/latest/

- Julia: https://manoptjl.org/

References i

[AP22] Pierre Ablin and Gabriel Peyré. “Fast and accurate optimization on the orthogonal manifold without retraction”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2022, pp. 5636–5657.

[Bar+12a] A. Barachant et al. “Multiclass Brain–Computer Interface Classification by Riemannian Geometry”. In: *IEEE Transactions on Biomedical Engineering* 59.4 (Apr. 2012), pp. 920–928.

[Bar+12b] Alexandre Barachant et al. “Multiclass Brain–Computer Interface Classification by Riemannian Geometry”. In: *IEEE Transactions on Biomedical Engineering* 59.4 (2012), pp. 920–928. DOI: `10.1109/TBME.2011.2172210`.

References ii

[Bre+21] Arnaud Breloy et al. “Majorization-Minimization on the Stiefel Manifold With Application to Robust Sparse PCA”. In: *IEEE Transactions on Signal Processing* 69 (2021), pp. 1507–1520. DOI: 10.1109/TSP.2021.3058442.

[Bro+17] Michael M Bronstein et al. “Geometric deep learning: going beyond euclidean data”. In: *IEEE Signal Processing Magazine* 34.4 (2017), pp. 18–42.

[Col+21] Antoine Collas et al. “Probabilistic PCA From Heteroscedastic Signals: Geometric Framework and Application to Clustering”. In: *IEEE Transactions on Signal Processing* 69 (2021), pp. 6546–6560. DOI: 10.1109/TSP.2021.3130997.

References iii

[Col+22] Antoine Collas et al. *Robust Geometric Metric Learning*. 2022. DOI: 10.48550/ARXIV.2202.11550. URL: https://arxiv.org/abs/2202.11550.

[DT05] Navneet Dalal and Bill Triggs. "Histograms of oriented gradients for human detection". In: *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. Vol. 1. IEEE. 2005, pp. 886–893.

[Gal+22] Matthieu Gallet et al. "Classification of GPR Signals via Covariance Pooling on CNN Features within a Riemannian Framework". In: *International Geoscience and Remote Sensing Symposium*. Kuala Lampur, Malaysia, July 2022. URL: https://hal.archives-ouvertes.fr/hal-03726277.

References iv

[HG17] Zhiwu Huang and Luc Van Gool. "A riemannian network for SPD matrix learning". In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. AAAI'17. San Francisco, California, USA: AAAI Press, 2017, pp. 2036–2042. URL: http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/download/14633/14371%20http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/download/14633/14371.

[Hip+21] Alexandre Hippert-Ferrer et al. *Riemannian classification of EEG signals with missing values*. 2021. DOI: 10.48550/ARXIV.2110.10011. URL: https://arxiv.org/abs/2110.10011.

References v

[HS15a] Mehrtash Harandi and Mathieu Salzmann. “Riemannian coding and dictionary learning: Kernels to the rescue”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3926–3935.

[HS15b] Reshad Hosseini and Suvrit Sra. “Matrix Manifold Optimization for Gaussian Mixtures”. In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes et al. Vol. 28. Curran Associates, Inc., 2015. URL: https://proceedings.neurips.cc/paper/2015/file/dbe272bab69f8e13f14b405e038deb64-Paper.pdf.

[Hua+18] Lei Huang et al. “Orthogonal weight normalization: Solution to optimization over multiple dependent stiefel manifolds in deep neural networks”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

References vi

[HWV18] Zhiwu Huang, Jiqing Wu, and Luc Van Gool. “Building deep networks on grassmann manifolds”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[IVS15] Catalin Ionescu, Orestis Vantzos, and Cristian Sminchisescu. *Training Deep Networks with Structured Layers by Matrix Backpropagation*. 2015. DOI: `10.48550/ARXIV.1509.07838`. URL: `https://arxiv.org/abs/1509.07838`.

[Ize12] Alan Julian Izenman. “Introduction to manifold learning”. In: *Wiley Interdisciplinary Reviews: Computational Statistics* 4.5 (2012), pp. 439–446.

References vii

[Jay+15] Sadeep Jayasumana et al. “Kernel Methods on Riemannian Manifolds with Gaussian RBF Kernels”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.12 (2015), pp. 2464–2477. DOI: 10.1109/TPAMI.2015.2414422.

[JHS16] Sadeep Jayasumana, Richard Hartley, and Mathieu Salzmann. “Kernels on Riemannian manifolds”. In: *Riemannian computing in computer vision*. Springer, 2016, pp. 45–67.

[Li+17] Peihua Li et al. “Is second-order information helpful for large-scale visual recognition?” In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2070–2078.

References viii

[MG06] S. Munder and D. M. Gavrila. “An Experimental Study on Pedestrian Classification”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.11 (Nov. 2006), pp. 1863–1868.

[MRO20] A. Mian, E. Raninen, and E. Ollila. “A Comparative Study of Supervised Learning Algorithms for Symmetric Positive Definite Features”. In: (2020).

[Pen16] Xavier Pennec. *Barycentric Subspace Analysis on Manifolds*. 2016. DOI: 10.48550/ARXIV.1607.02833. URL: https://arxiv.org/abs/1607.02833.

References ix

[RMH18] Soumava Kumar Roy, Zakaria Mhammedi, and Mehrtash Harandi. "Geometry aware constrained optimization techniques for deep learning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4460–4469.

[Sai+17] Salem Said et al. "Riemannian Gaussian Distributions on the Space of Symmetric Positive Definite Matrices". In: *IEEE Transactions on Information Theory* 63.4 (2017), pp. 2153–2170. DOI: 10.1109/TIT.2017.2653803.

References x

[SH16] Suvrit Sra and Reshad Hosseini. “Geometric Optimization in Machine Learning”. In: *Algorithmic Advances in Riemannian Geometry and Applications: For Machine Learning, Computer Vision, Statistics, and Optimization*. Ed. by Hà Quang Minh and Vittorio Murino. Cham: Springer International Publishing, 2016, pp. 73–91. ISBN: 978-3-319-45026-1. DOI: 10.1007/978-3-319-45026-1_3. URL: https://doi.org/10.1007/978-3-319-45026-1_3.

[Str18] Volker Strobel. *Pold87/academic-keyword-occurrence: First release*. Apr. 2018. DOI: 10.5281/zenodo.1218409. URL: https://zenodo.org/record/1218409 (visited on 08/18/2022).

References xi

[TPM08a] O. Tuzel, F. Porikli, and P. Meer. “Pedestrian Detection via Classification on Riemannian Manifolds”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.10 (Oct. 2008), pp. 1713–1727.

[TPM08b] Oncel Tuzel, Fatih Porikli, and Peter Meer. “Pedestrian Detection via Classification on Riemannian Manifolds”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.10 (2008), pp. 1713–1727. DOI: `10.1109/TPAMI.2008.75`.

[Wu+20] Zonghan Wu et al. “A comprehensive survey on graph neural networks”. In: *IEEE transactions on neural networks and learning systems* 32.1 (2020), pp. 4–24.

References xii

[ZHS16] Pourya Zadeh, Reshad Hosseini, and Suvrit Sra. "Geometric mean metric learning". In: *International conference on machine learning*. PMLR. 2016, pp. 2464–2471.