Robust Geometric Metric Learning

Antoine Collas*, Arnaud Breloy†, Guillaume Ginolhac‡, Chengfang Ren*, Jean-Philippe Ovarlez*§
*SONDRA, CentraleSupélec, University Paris Saclay, †LEME, University Paris Nanterre,
‡LISTIC, University Savoie Mont-Blanc, §DEMR, ONERA, University Paris Saclay

***Abstract*—This paper proposes new algorithms for the *metric learning* problem. We start by noticing that several classical *metric learning* formulations from the literature can be viewed as modified covariance matrix estimation problems. Leveraging this point of view, a general approach, called *Robust Geometric Metric Learning* (RGML), is then studied. This method aims at simultaneously estimating the covariance matrix of each class while shrinking them towards their (unknown) barycenter. We focus on two specific costs functions: one associated with the Gaussian likelihood (*RGML Gaussian*), and one with Tyler's M-estimator (*RGML Tyler*). In both, the barycenter is defined with the Riemannian distance, which enjoys nice properties of geodesic convexity and affine invariance. The optimization is performed using the Riemannian geometry of symmetric positive definite matrices and its submanifold of unit determinant. Finally, the performance of RGML is asserted on real datasets. Strong performance is exhibited while being robust to mislabeled data.**

***Index Terms*—covariance, robust estimation, Riemannian geometry, Riemannian distance, geodesic convexity, metric learning**

I. Introduction

Many classification algorithms rely on the distance between data points. These algorithms include the classical *K-means*, *Nearest centroïd classifier*, *k-nearest neighbors* and their variants. The definition of the distance is thus of crucial importance since it determines which points will be considered similar or not, thus implies the classification rule. In practice, classification algorithms most generally rely on the the Euclidean distance, which is $d_{\boldsymbol{I}_p}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2$ for $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathbb{R}^p$. However, this distance is prone to several issues. A pathological example is when two classes have a high variance along one common axis: within this configuration, two data points from the same class can be far away from each other, while two data points from two different classes can be very close.

To find a more relevant distance for classification, the problem of *metric learning* has been proposed. *Metric learning* aims at finding a Mahalanobis distance

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

that brings data points from same class closer, and furthers data points from different classes away. Mathematically, *metric learning* is an optimization problem of a loss function that relies on $d_{\boldsymbol{A}}$. This minimization is achieved over $\boldsymbol{A}$, a matrix that belongs to $\mathcal{S}_p^+$ the set of $p \times p$ symmetric positive definite matrices. The constraints of symmetricity and positivity are enforced so that $d_{\boldsymbol{A}}$ is a distance.

In the following, we consider being in a supervised regime with K classes, *i.e.* m data points $\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_m\}$ in $\mathbb{R}^p$ with their labels in $[\![1, K]\!]$ are available. Data points can be grouped by classes and the elements of the k^{th} class are denoted $\{\boldsymbol{x}_{kl}\}$. Then, n_k pairs, $(\boldsymbol{x}_{kl}, \boldsymbol{x}_{kq})$ with $kl \neq kq$, of elements of the class k are formed. The set S_k contains all these pairs and S contains the $n_{\mathcal{S}} = \sum_{k=1}^K n_k$ pairs of all the classes. When S is used, the class of a pair is not relevant, thus it is denoted by $(\boldsymbol{x}_l, \boldsymbol{x}_q)$ instead of $(\boldsymbol{x}_{kl}, \boldsymbol{x}_{kq})$. The ratio $\frac{n_k}{n_{\mathcal{S}}}$ is denoted π_k. Then, each vector $\boldsymbol{s}_{ki}$ is defined as the subtraction of the elements of each pair in $\mathcal{S}_k$, *i.e.* $\boldsymbol{s}_{ki} = \boldsymbol{x}_{kl} - \boldsymbol{x}_{kq}$ for $(\boldsymbol{x}_{kl}, \boldsymbol{x}_{kq}) \in \mathcal{S}_k$. Thus, the set $\{\boldsymbol{s}_{ki}\}$ contains n_k elements. Then, the set $\mathcal{D}$ contains $n_{\mathcal{D}}$ pairs of vectors that do not belong to the same class. Each vector $\boldsymbol{d}_i$ is defined as the subtraction of the elements of each pair in $\mathcal{D}$, *i.e.* $\boldsymbol{d}_i = \boldsymbol{x}_l - \boldsymbol{x}_q$ for $(\boldsymbol{x}_l, \boldsymbol{x}_q) \in \mathcal{D}$. Finally, $\mathcal{S}_p$ is the set of $p \times p$ symmetric matrices, $\mathcal{S}_p^+$ is the set of $p \times p$ symmetric positive definite matrices, and $\mathcal{SS}_p^+$ is the set of $p \times p$ symmetric positive definite matrices with unit determinant.

A. State of the art

Many *metric learning* problems have been formulated over the years (see e.g. [1] for a complete survey). In the following, we present notable ones that are related to our proposal.

MMC [2] (Mahalanobis Metric for Clustering) was one of the earliest paper in this field. This method minimizes the sum of squared distances over similar data while constraining dissimilar data to be far away from each other. *MMC* writes

$$\begin{aligned} &\underset{\boldsymbol{A} \in \mathcal{S}_p^+}{\text{minimize}} \sum_{(\boldsymbol{x}_l, \boldsymbol{x}_q) \in \mathcal{S}} d_{\boldsymbol{A}}^2(\boldsymbol{x}_l, \boldsymbol{x}_q) \\ &\text{subject to} \sum_{(\boldsymbol{x}_l, \boldsymbol{x}_q) \in \mathcal{D}} d_{\boldsymbol{A}}(\boldsymbol{x}_l, \boldsymbol{x}_q) \geq 1. \end{aligned} \tag{2}$$

Notice that $d_{\boldsymbol{A}}$ (rather than $d_{\boldsymbol{A}}^2$) is involved in the constraint in order to avoid a trivial rank-one solution.

Then, *ITML* [3] (Information-Theoretic Metric Learning) proposed to find a matrix $\boldsymbol{A}$ that stays close to a predefined matrix $\boldsymbol{A}_0$ while respecting constraints of similarities and dissimilarities. The proximity between $\boldsymbol{A}$ and $\boldsymbol{A}_0$ is measured with the *Gaussian Kullback-Leibler divergence* $D_{KL}(\boldsymbol{A}_0, \boldsymbol{A}) = \operatorname{Tr}(\boldsymbol{A}^{-1}\boldsymbol{A}_0) + \log|\boldsymbol{A}\boldsymbol{A}_0^{-1}|$. *ITML* writes

$$\begin{aligned} \underset{\boldsymbol{A} \in \mathcal{S}_p^+}{\text{minimize}} \quad & \operatorname{Tr}(\boldsymbol{A}^{-1}\boldsymbol{A}_0) + \log|\boldsymbol{A}| \\ \text{subject to} \quad & d_{\boldsymbol{A}}^2(\boldsymbol{x}_l, \boldsymbol{x}_q) \leq u, \quad (\boldsymbol{x}_l, \boldsymbol{x}_q) \in \mathcal{S}, \\ & d_{\boldsymbol{A}}^2(\boldsymbol{x}_l, \boldsymbol{x}_q) \geq l, \quad (\boldsymbol{x}_l, \boldsymbol{x}_q) \in \mathcal{D}, \end{aligned} \tag{3}$$

where $u, v \in \mathbb{R}$ are threshold parameters, chosen to enforce closeness of similar points and farness of dissimilar points.

Usually $\boldsymbol{A}_0$ is chosen as the identity matrix or as the sample covariance matrix (SCM) of the set $\{\boldsymbol{s}_{ki}\}$.

Next, *GMML* (Geometric Mean Metric Learning) [4] is an algorithm of great interest. Indeed, it achieves impressive performance on several datasets while being very fast thanks to a closed form formula. The *GMML* problem writes

$$\underset{\boldsymbol{A}\in\mathcal{S}_p^+}{\text{minimize}} \frac{1}{n_{\mathcal{S}}} \sum_{(\boldsymbol{x}_l,\boldsymbol{x}_q)\in\mathcal{S}} d^2_{\boldsymbol{A}}(\boldsymbol{x}_l,\boldsymbol{x}_q) + \frac{1}{n_{\mathcal{D}}} \sum_{(\boldsymbol{x}_l,\boldsymbol{x}_q)\in\mathcal{D}} d^2_{\boldsymbol{A}^{-1}}(\boldsymbol{x}_l,\boldsymbol{x}_q). \quad (4)$$

The intuition behind this problem is that $d_{\boldsymbol{A}^{-1}}$ should be able to further away dissimilar points while $d_{\boldsymbol{A}}$ close together similar points. Then, *GMML* formulation (4) can be rewritten

$$\underset{\boldsymbol{A}\in\mathcal{S}_p^+}{\text{minimize}} \operatorname{Tr}(\boldsymbol{A}^{-1}\boldsymbol{S}) + \operatorname{Tr}(\boldsymbol{A}\boldsymbol{D}), \quad (5)$$

where $\boldsymbol{S} = \frac{1}{n_{\mathcal{S}}} \sum_{k=1}^{K} \sum_{i=1}^{n_k} \boldsymbol{s}_{ki}\boldsymbol{s}_{ki}^T$ and $\boldsymbol{D} = \frac{1}{n_{\mathcal{D}}} \sum_{i=1}^{n_{\mathcal{D}}} \boldsymbol{d}_i\boldsymbol{d}_i^T$. In [4], the solution of (5) is derived. It is the geodesic mid-point between $\boldsymbol{S}^{-1}$ and $\boldsymbol{D}$, *i.e.* $\boldsymbol{A}^{-1} = \boldsymbol{S}^{-1}\#_{\frac{1}{2}}\boldsymbol{D}$ where

$$\boldsymbol{S}^{-1}\#_t\boldsymbol{D} = \boldsymbol{S}^{-\frac{1}{2}} \left(\boldsymbol{S}^{\frac{1}{2}}\boldsymbol{D}\boldsymbol{S}^{\frac{1}{2}}\right)^t \boldsymbol{S}^{-\frac{1}{2}}. \quad (6)$$

Then, [4] proposes to generalize this solution by $\boldsymbol{A}^{-1} = \boldsymbol{S}^{-1}\#_t\boldsymbol{D}$ with $t \in [0,1]$.

B. Metric learning as covariance matrix estimation

In this sub-section, some *metric learning* problems are expressed as covariance matrix estimation problems.

The first remark concerns the *ITML* formulation (3). Indeed, when the latter is written with the SCM as a prior matrix, it amounts to maximizing the likelihood of a multivariate Gaussian distribution under constraints. Therefore, *ITML* can be viewed as a *covariance* matrix estimation problem.

The second remark concerns the *GMML* solution of (5) which is generalized to $\boldsymbol{A}^{-1} = \boldsymbol{S}^{-1}\#_t\boldsymbol{D}$ with $t \in [0,1]$. In their experiments on real datasets, the authors often get their best performance with t small (or even null) (see Figure 3 of [4]). In this case, the *GMML* algorithm gives $\boldsymbol{A} = \boldsymbol{S}$. This simple, yet effective, solution can be reinterpreted with an additional assumption on the data. Let us assume that data points of each class are realizations of independent random vectors with class-dependent first and second order moments,

$$\boldsymbol{x}_{kl} \overset{d}{=} \boldsymbol{\mu}_k + \boldsymbol{\Sigma}_k^{\frac{1}{2}}\boldsymbol{u}_{kl}, \quad (7)$$

with $\boldsymbol{\mu}_k \in \mathbb{R}^p$, $\boldsymbol{\Sigma}_k \in \mathcal{S}_p^+$, $\mathbb{E}[\boldsymbol{u}_{kl}] = \boldsymbol{0}$ and $\mathbb{E}[\boldsymbol{u}_{kl}\boldsymbol{u}_{kq}^T] = \boldsymbol{I}_p$ if $kl = kq$, $\boldsymbol{0}_p$ otherwise. Thus, it follows that $\boldsymbol{s}_{ki} \overset{d}{=} \boldsymbol{\Sigma}_k^{\frac{1}{2}}(\boldsymbol{u}_{kl} - \boldsymbol{u}_{kq})$. Hence, the covariance matrix of $\boldsymbol{s}_{ki}$ is twice the covariance matrix of the k^{th} class, $\mathbb{E}[\boldsymbol{s}_{ki}\boldsymbol{s}_{ki}^T] \overset{d}{=} 2\boldsymbol{\Sigma}_k$. It results that, in expectation, $\boldsymbol{S}$ is twice the arithmetic mean of the covariance matrices of the different classes,

$$\mathbb{E}[\boldsymbol{S}] = \frac{1}{n_{\mathcal{S}}} \sum_{k=1}^{K} \sum_{i=1}^{n_k} \mathbb{E}[\boldsymbol{s}_{ki}\boldsymbol{s}_{ki}^T] = 2\sum_{k=1}^{K} \pi_k\boldsymbol{\Sigma}_k. \quad (8)$$

The only additional assumption added to *GMML* to get (8) is (7). This hypothesis is broad since it encompasses classical assumptions such as the Gaussian one. Also notice that using $\boldsymbol{S}$ in the Mahalanobis distance (1) is reminiscent of the linear discriminant analysis (LDA) pre-whitening step of the data.

C. Motivations and contributions

From Section I-B, *GMML* can be interpreted as a 2-steps method that computes, first, the SCM of each class and, two, their arithmetic mean. Thus, this simple approach is not robust to outliers (*e.g.* mislabeled data) since it uses the SCM as an estimator. Moreover, other mean computation can be used, such as the Riemannian mean which benefits from many properties compared to its Euclidean counterpart [5]. We propose a *metric learning* framework that jointly estimates regularized covariance matrices, in a robust manner, while computing their Riemannian mean. We name this framework *Riemannian Geometric Metric Learning* (*RGML*).

This idea of estimating covariance matrices while averaging them was firstly proposed in [6]. The novelty here is fourfold: **1)** this formulation is applied to the problem of *metric learning* (see Section II), **2)** it makes use of the Riemannian distance on $\mathcal{S}_p^+$ which was not covered by [6] (see Section II), **3)** we leverage the Riemannian geometries of $\mathcal{S}_p^+$ and $\mathcal{SS}_p^+$ [7], [8] along with the framework of Riemannian optimization [9] and hence the proposed algorithms are flexible and could be applied to other cost functions than the Gaussian and Tyler [10] ones (see Section III), **4)** the framework is applied on real datasets and shows strong performance while being robust to mislabeled data (see Section IV).

II. Problem formulation

A. General formulation of RGML

The formulation of the *RGML* optimization problem is

$$\underset{\theta\in\mathcal{M}_{p,K}}{\text{minimize}} \left\{ h(\theta) = \sum_{k=1}^{K} \pi_k \left[\mathcal{L}_k(\boldsymbol{A}_k) + \lambda d^2(\boldsymbol{A},\boldsymbol{A}_k)\right] \right\}, \quad (9)$$

where $\theta = (\boldsymbol{A}, \{\boldsymbol{A}_k\})$, $\mathcal{M}_{p,K}$ is the $K+1$ product set of $\mathcal{S}_p^+$, *i.e.* $\mathcal{M}_{p,K} = \left(\mathcal{S}_p^+\right)^{K+1}$, $\mathcal{L}_k$ is a covariance matrix estimation loss on $\{\boldsymbol{s}_{ki}\}$, $\lambda > 0$ and d is a distance between matrices. In the next subsections two costs will be considered: the Gaussian negative log-likelihood and the Tyler cost function. Once (9) is achieved, the center matrix $\boldsymbol{A}$ is used in the Mahalanobis distance (1) and the $\boldsymbol{A}_k$ are discarded. The cost function h is explained more in details in the following.

First of all, for a fixed center matrix $\boldsymbol{A}$, (9) reduces to k separable problems

$$\underset{\boldsymbol{A}_k}{\text{minimize}} \mathcal{L}_k(\boldsymbol{A}_k) + \lambda d^2(\boldsymbol{A},\boldsymbol{A}_k), \quad (10)$$

whose solutions are estimates of $\{\boldsymbol{\Sigma}_k\}$ that are regularized towards $\boldsymbol{A}$.

Second, for $\{\boldsymbol{A}_k\}$ fixed, solving (9) averages the matrices $\{\boldsymbol{A}_k\}$. Indeed, in this case, (9) reduces to

$$\underset{\boldsymbol{A}\in\mathcal{S}_p^+}{\text{minimize}} \sum_{k=1}^{K} \pi_k d^2(\boldsymbol{A},\boldsymbol{A}_k). \quad (11)$$

For example, if d is the Euclidean distance $d_E(\boldsymbol{A}, \boldsymbol{A}_k) = \|\boldsymbol{A} - \boldsymbol{A}_k\|_2$, then the minimum of (11) is the arithmetic mean $\sum_{k=1}^K \pi_k \boldsymbol{A}_k$. In the rest of the paper, we consider the Riemannian distance on $\mathcal{S}_p^+$ [7], that is

$$d_R(\boldsymbol{A}, \boldsymbol{A}_k) = \left\| \operatorname{logm}\left(\boldsymbol{A}^{-\frac{1}{2}} \boldsymbol{A}_k \boldsymbol{A}^{-\frac{1}{2}}\right) \right\|_2 . \tag{12}$$

A nice property of d_R (12) is its affine invariance. Indeed, for any $\boldsymbol{C}$ invertible, we have $d_R(\boldsymbol{C}\boldsymbol{A}\boldsymbol{C}^T, \boldsymbol{C}\boldsymbol{A}_k\boldsymbol{C}^T) = d_R(\boldsymbol{A}, \boldsymbol{A}_k)$. Thus, if $\{\boldsymbol{s}_{ki}\}$ is transformed to $\{\boldsymbol{C}\boldsymbol{s}_{ki}\}$ then the minimum $(\boldsymbol{A}, \{\boldsymbol{A}_k\})$ of (13) becomes $\left(\boldsymbol{C}\boldsymbol{A}\boldsymbol{C}^T, \left\{\boldsymbol{C}\boldsymbol{A}_k\boldsymbol{C}^T\right\}\right)$. Another nice property of this distance is its geodesic convexity, as it will be discussed in Section III.

With this Riemannian distance, the general formulation of the *RGML* optimization problem (9) becomes

$$\underset{\theta \in \mathcal{M}_{p,K}}{\operatorname{minimize}} \left\{ h(\theta) = \sum_{k=1}^K \pi_k \left[\mathcal{L}_k(\boldsymbol{A}_k) + \lambda d_R^2(\boldsymbol{A}, \boldsymbol{A}_k) \right] \right\}. \tag{13}$$

We emphasis that the optimization of (13) is performed with respect to all the matrices $\boldsymbol{A}$ and $\{\boldsymbol{A}_k\}$ at the same time. Thus it both estimates regularized covariance matrices $\{\boldsymbol{A}_k\}$ while averaging them to estimate their unknown barycenter $\boldsymbol{A}$.

B. RGML Gaussian

To get a practical cost function h (13), it only remains to specify the functions $\mathcal{L}_k$. The most classical assumption on the data distribution is the Gaussian one (*e.g.* considered in *ITML* with the SCM as prior). Thus, the first functions $\mathcal{L}_k$ considered are the centered multivariate Gaussian negative log-likelihoods

$$\mathcal{L}_{G,k}(\boldsymbol{A}) = \frac{1}{n_k} \sum_{i=1}^{n_k} \boldsymbol{s}_{ki}^T \boldsymbol{A}^{-1} \boldsymbol{s}_{ki} + \log|\boldsymbol{A}|. \tag{14}$$

With this negative log-likelihood, the *RGML* optimization problem (13) becomes

$$\underset{\theta \in \mathcal{M}_{p,K}}{\operatorname{minimize}} \left\{ h_G(\theta) = \sum_{k=1}^K \pi_k \left[\mathcal{L}_{G,k}(\boldsymbol{A}_k) + \lambda d_R^2(\boldsymbol{A}, \boldsymbol{A}_k) \right] \right\} \tag{15}$$

C. RGML Tyler

When data is non-Gaussian, robust covariance matrix estimation methods are a preferred choice. This occurs whenever the probability distribution of the data is heavy-tailed or a small proportion of the samples represents outlier behavior. In a classification setting, the latter happens when data are mislabeled. A classical robust estimator is the Tyler's estimator [10] which is the minimizer of the following cost function

$$\mathcal{L}_{T,k}(\boldsymbol{A}) = \frac{p}{n_k} \sum_{i=1}^{n_k} \log\left(\boldsymbol{s}_{ki}^T \boldsymbol{A}^{-1} \boldsymbol{s}_{ki}\right) + \log|\boldsymbol{A}|. \tag{16}$$

An important remark is that (16) is invariant to the scale of $\boldsymbol{A}$. Indeed $\forall \alpha > 0$, it is easily checked that $\mathcal{L}_{T,k}(\alpha \boldsymbol{A}) = \mathcal{L}_{T,k}(\boldsymbol{A})$. Thus, a constraint of unit determinant is added to (13) to fix the scales of $\{\boldsymbol{A}_k\}$. Furthermore, the Riemannian distance (12) is also the one on $\mathcal{S}\mathcal{S}_p^+$. Thus, we choose to also constrain $\boldsymbol{A}$ so that it is the Riemannian mean of $\{\boldsymbol{A}_k\}$ on $\mathcal{S}\mathcal{S}_p^+$. We denote by $\mathcal{S}\mathcal{M}_{p,K}$ this new parameter space

$$\mathcal{S}\mathcal{M}_{p,K} = \{\theta \in \mathcal{M}_{p,K}, |\boldsymbol{A}| = |\boldsymbol{A}_k| = 1, \forall k \in [\![1, K]\!]\}. \tag{17}$$

Thus, the *RGML* optimization problem (13) with the Tyler cost function (16) becomes

$$\underset{\theta \in \mathcal{S}\mathcal{M}_{p,K}}{\operatorname{minimize}} \left\{ h_T(\theta) = \sum_{k=1}^K \pi_k \left[\mathcal{L}_{T,k}(\boldsymbol{A}_k) + \lambda d_R^2(\boldsymbol{A}, \boldsymbol{A}_k) \right] \right\} \tag{18}$$

III. Riemannian optimization

The objective of this section is to present the Algorithms 1 and 2 which minimize (15) and (18) respectively. They leverage the Riemannian optimization framework [9], [11]. The products manifolds $\mathcal{M}_{p,K}$ and $\mathcal{S}\mathcal{M}_{p,K}$ (directly inherited from $\mathcal{S}_p^+$ and $\mathcal{S}\mathcal{S}_p^+$ [7], [8]) are presented.

A. Riemannian optimization and g-convexity on $\mathcal{M}_{p,K}$

Since, $\mathcal{M}_{p,K}$ is an open set in a vector space, the tangent space $T_\theta \mathcal{M}_{p,K}$ (linearization of the Riemannian manifold at a given point) is identified to $(\mathcal{S}_p)^{K+1}$. Then, the affine invariant metric is chosen as the Riemannian metric [7], $\forall \xi = (\boldsymbol{\xi}, \{\boldsymbol{\xi}_k\}), \forall \eta = (\boldsymbol{\eta}, \{\boldsymbol{\eta}_k\}) \in T_\theta \mathcal{M}_{p,K}$

$$\langle \xi, \eta \rangle_\theta^{\mathcal{M}_{p,K}} = \operatorname{Tr}\left(\boldsymbol{A}^{-1}\boldsymbol{\xi}\boldsymbol{A}^{-1}\boldsymbol{\eta}\right) + \sum_{k=1}^K \operatorname{Tr}\left(\boldsymbol{A}_k^{-1}\boldsymbol{\xi}_k\boldsymbol{A}_k^{-1}\boldsymbol{\eta}_k\right). \tag{19}$$

Thus the orthogonal projection from the ambient space onto the tangent space at θ is

$$P_\theta^{\mathcal{M}_{p,K}}(\xi) = (\operatorname{sym}(\boldsymbol{\xi}), \{\operatorname{sym}(\boldsymbol{\xi}_k)\}) , \tag{20}$$

where $\operatorname{sym}(\boldsymbol{\xi}) = \frac{1}{2}(\boldsymbol{\xi} + \boldsymbol{\xi}^T)$. Then, the exponential map (function that maps tangent vectors, such as gradients of loss functions, to points on the manifold) is

$$\exp_\theta^{\mathcal{M}_{p,K}}(\xi) = \left(\exp_{\boldsymbol{A}}^{\mathcal{S}_p^+}(\boldsymbol{\xi}), \left\{ \exp_{\boldsymbol{A}_k}^{\mathcal{S}_p^+}(\boldsymbol{\xi}_k) \right\} \right), \tag{21}$$

where $\exp_{\boldsymbol{A}}^{\mathcal{S}_p^+}(\boldsymbol{\xi}) = \boldsymbol{A} \operatorname{expm}(\boldsymbol{A}^{-1}\boldsymbol{\xi})$. Then, for a loss function $\ell : \mathcal{M}_{p,K} \to \mathbb{R}$, the Riemannian gradient at θ denoted $\nabla_{\mathcal{M}_{p,K}} \ell(\theta)$ is defined as the unique element such that $\forall \xi \in T_\theta \mathcal{M}_{p,K}$, $\mathrm{D}\,\ell(\theta)[\xi] = \langle \nabla_{\mathcal{M}_{p,K}} \ell(\theta), \xi \rangle_\theta^{\mathcal{M}_{p,K}}$ where D is the directional derivative. It results that

$$\nabla_{\mathcal{M}_{p,K}} \ell(\theta) = P_\theta^{\mathcal{M}_{p,K}} (\boldsymbol{A}\boldsymbol{G}\boldsymbol{A}, \{\boldsymbol{A}_k\boldsymbol{G}_k\boldsymbol{A}_k\}) , \tag{22}$$

where $(\boldsymbol{G}, \{\boldsymbol{G}_k\})$ is the classical Euclidean gradient of ℓ at θ. In practice this Euclidean gradient can be computed using automatic differentiation libraries such as JAX [12]. With the exponential map (21), and the Riemannian gradient (22), we have the main tools to minimize (15). However, to improve the numerical stability, a retraction (approximation of the exponential map (21)) is preferred,

$$R_\theta^{\mathcal{M}_{p,K}}(\xi) = \left(R_{\boldsymbol{A}}^{\mathcal{S}_p^+}(\boldsymbol{\xi}), \left\{ R_{\boldsymbol{A}_k}^{\mathcal{S}_p^+}(\boldsymbol{\xi}_k) \right\} \right), \tag{23}$$

where $R_{\boldsymbol{A}}^{\mathcal{S}_p^+}(\boldsymbol{\xi}) = \boldsymbol{A} + \boldsymbol{\xi} + \frac{1}{2}\boldsymbol{\xi}\boldsymbol{A}^{-1}\boldsymbol{\xi}$. A Riemannian gradient descent minimizing (15) is presented in Algorithm 1.

Algorithm 1: Riemannian gradient descent to minimize h_G (15)

Input: Data $\{\boldsymbol{s}_{ki}\}$, initialization $\theta^{(0)} \in \mathcal{M}_{p,K}$
Output: $\theta^{(t)} \in \mathcal{M}_{p,K}$
for $t = 0$ ***to convergence*** **do**
  Compute a step size α (see [9, Ch. 4]) and set
  $\theta^{(t+1)} = R_{\theta^{(t)}}^{\mathcal{M}_{p,K}}\left(-\alpha \nabla_{\mathcal{M}_{p,K}} h_G(\theta^{(t)})\right)$

Algorithm 2: Riemannian gradient descent to minimize h_T (18)

Input: Data $\{\boldsymbol{s}_{ki}\}$, initialization $\theta^{(0)} \in \mathcal{SM}_{p,K}$
Output: $\theta^{(t)} \in \mathcal{SM}_{p,K}$
for $t = 0$ ***to convergence*** **do**
  Compute a step size α (see [9, Ch. 4]) and set
  $\theta^{(t+1)} = R_{\theta^{(t)}}^{\mathcal{SM}_{p,K}}\left(-\alpha \nabla_{\mathcal{SM}_{p,K}} h_T(\theta^{(t)})\right)$

We finish this subsection by presenting the geodesic convexity of h_G (15) on $\mathcal{M}_{p,K}$ (see [11, Chapter 11] for a presentation of the geodesic convexity). First of all, the geodesic on $\mathcal{M}_{p,K}$ between $a = (\boldsymbol{A}, \{\boldsymbol{A}_k\})$ and $b = (\boldsymbol{B}, \{\boldsymbol{B}_k\})$ is

$$a \#_t b = (\boldsymbol{A} \#_t \boldsymbol{B}, \{\boldsymbol{A}_k \#_t \boldsymbol{B}_k\}) \ , \tag{24}$$

where $\#$ is the geodesic (6) on $\mathcal{S}_p^+$ and $t \in [0, 1]$. Then, a loss function ℓ is said to be geodesically convex (or g-convex) if

$$\ell\left(a \#_t b\right) \leq t\, \ell(a) + (1 - t)\, \ell(b), \quad \forall t \in [0, 1]. \tag{25}$$

If ℓ is g-convex, then any local minimizer is a global minimizer. [6] proves that h_G (15) is g-convex. Hence, any local minimizer of (15) is a global minimizer.

B. $\mathcal{SM}_{p,K}$: a geodesic submanifold of $\mathcal{M}_{p,K}$

In (17), $\mathcal{SM}_{p,K}$ is defined as a subset of $\mathcal{M}_{p,K}$. In fact, $\mathcal{SM}_{p,K}$ can even be turned into a Riemannian submanifold of $\mathcal{M}_{p,K}$. First of all, the tangent space of $\mathcal{SM}_{p,K}$ at θ is

$$T_\theta \mathcal{SM}_{p,K} = \Big\{\xi \in T_\theta \mathcal{M}_{p,K} : \operatorname{Tr}(\boldsymbol{A}^{-1}\boldsymbol{\xi}) = 0, \\ \operatorname{Tr}(\boldsymbol{A}_k^{-1}\boldsymbol{\xi}_k) = 0 \quad \forall k \in [\![1, K]\!]\Big\}. \tag{26}$$

By endowing $\mathcal{SM}_{p,K}$ with the Riemannian metric of $\mathcal{M}_{p,K}$, it becomes a Riemannian submanifold. $\forall \xi, \eta \in T_\theta \mathcal{SM}_{p,K}$ we have $\langle \xi, \eta \rangle_\theta^{\mathcal{SM}_{p,K}} = \langle \xi, \eta \rangle_\theta^{\mathcal{M}_{p,K}}$. The orthogonal projection from the ambient space onto the tangent space at θ is

$$P_\theta^{\mathcal{SM}_{p,K}}(\xi) = \left(P_{\boldsymbol{A}}^{\mathcal{SS}_p^+}(\boldsymbol{\xi}), \left\{P_{\boldsymbol{A}_k}^{\mathcal{SS}_p^+}(\boldsymbol{\xi}_k)\right\}\right), \tag{27}$$

where $P_{\boldsymbol{A}}^{\mathcal{SS}_p^+}(\boldsymbol{\xi}) = \operatorname{sym}(\boldsymbol{\xi}) - \frac{1}{p}\operatorname{Tr}\left(\boldsymbol{A}^{-1}\operatorname{sym}(\boldsymbol{\xi})\right)\boldsymbol{A}$. A remarkable result is that $\mathcal{SM}_{p,K}$ is a geodesic submanifold of $\mathcal{M}_{p,K}$, *i.e.*, the geodesics of $\mathcal{SM}_{p,K}$ are those of $\mathcal{M}_{p,K}$. It results that the exponential mapping on $\mathcal{SM}_{p,K}$ is $\exp_\theta^{\mathcal{SM}_{p,K}}(\xi) = \exp_\theta^{\mathcal{M}_{p,K}}(\xi)$. Then, for a loss function $\ell : \mathcal{SM}_{p,K} \to \mathbb{R}$, the Riemannian gradient at θ is

$$\nabla_{\mathcal{SM}_{p,K}} \ell(\theta) = P_\theta^{\mathcal{SM}_{p,K}}\left(\boldsymbol{AGA}, \{\boldsymbol{A}_k \boldsymbol{G}_k \boldsymbol{A}_k\}\right) , \tag{28}$$

where $(\boldsymbol{G}, \{\boldsymbol{G}_k\})$ is the classical Euclidean gradient of ℓ at θ. Once again, a retraction that approximates the exponential mapping is leveraged to improve the numerical stability,

$$R_\theta^{\mathcal{SM}_{p,K}}(\xi) = \left(R_{\boldsymbol{A}}^{\mathcal{SS}_p^+}(\boldsymbol{\xi}), \left\{R_{\boldsymbol{A}_k}^{\mathcal{SS}_p^+}(\boldsymbol{\xi}_k)\right\}\right), \tag{29}$$

where $R_{\boldsymbol{A}}^{\mathcal{SS}_p^+}(\boldsymbol{\xi}) = \dfrac{\boldsymbol{A} + \boldsymbol{\xi} + \frac{1}{2}\boldsymbol{\xi}\boldsymbol{A}^{-1}\boldsymbol{\xi}}{\left|\boldsymbol{A} + \boldsymbol{\xi} + \frac{1}{2}\boldsymbol{\xi}\boldsymbol{A}^{-1}\boldsymbol{\xi}\right|^{\frac{1}{p}}}$.

Finally, h_T (18) is g-convex on $\mathcal{SM}_{p,K}$. Indeed, [6] proved that h_T is g-convex on $\mathcal{M}_{p,K}$ and $\mathcal{SM}_{p,K}$ is a geodesic submanifold of $\mathcal{M}_{p,K}$.

IV. Experiments

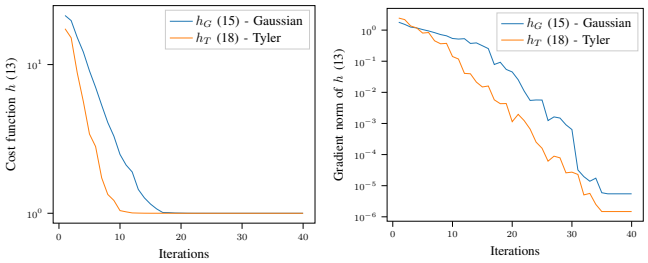

Fig. 1: Left: Gaussian (15) and Tyler (18) costs functions with respect to the number of iterations of Algorithms 1 and 2 respectively. Right: Riemannian gradient norms of Gaussian (15) and Tyler (18) costs functions. The optimization is performed on the *Wine* dataset.

In this section, we exhibit a practical interest of the *RGML* method developed in Sections II and III. All implementations of the following experiments are available at https://github.com/antoinecollas/robust_metric_learning. We apply it on real datasets from the *UCI machine learning repository* [13]. The three considered datasets are: *Wine*, *Vehicle*, and *Iris*. They are classification datasets, and their data dimensions along with their number of classes are presented in Table I. These datasets are well balanced, *i.e.* they roughly have the same number of data for all the classes. The numbers of generated pairs in $\mathcal{S}$ and $\mathcal{D}$ are $n_{\mathcal{S}} = n_{\mathcal{D}} = 75\, K(K-1)$ (as in [3] and [4]).

The classification is done following a very classical protocol in *metric learning*. **1)** A matrix $\boldsymbol{A}$ is estimated via a *metric learning* method. **2)** The data $\{\boldsymbol{x}_l\}$ are multiplied by $\boldsymbol{A}^{-\frac{1}{2}}$ to get $\{\boldsymbol{A}^{-\frac{1}{2}}\boldsymbol{x}_l\}$. **3)** The data $\{\boldsymbol{A}^{-\frac{1}{2}}\boldsymbol{x}_l\}$ are classified using a *k-nearest neighbors* with 5 neighbors. Thus, the classification is performed using the Mahalanobis distance $d_{\boldsymbol{A}}$ defined by (1) in the Introduction. This classification is repeated 200 times via cross-validation. The proportion of the training/test sets is $50/50$. The error of classification is computed for each fold and the mean error is reported in Table I. In order to show the robustness of the proposed method, mislabeled data are introduced. To do so, we randomly select data in the training set whose labels are then randomly changed for new labels.

The implementations of the cross-validation as well as the *k-nearest neighbors* are from the scikit-learn library [14].

	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
RGML - 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
RGML - 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 I: Misclassification errors on 3 datasets: Wine, Vehicle and Iris. Best results and those within 0.05% are in **bold**. The mislabeling rates indicate the percentage of labels that are randomly changed in the training set.

The proposed methods *RGML Gaussian* and *RGML Tyler* have been implemented using JAX [12]. The chosen value of parameter λ is 0.05. Its value has little impact on performance as long as it is neither too small nor too large. The proposed algorithms are compared to the classical *metric learning* algorithms: the identity matrix (called Euclidean in Table I), the SCM computed on all the data, *ITML* [3], *GMML* [4], and *LMNN* [15]. The implementations of the metric-learn library [16] are used for the last three algorithms.

From Table I, several observations are made. First of all, on the raw data (*i.e.* when the mislabeling rate is 0%) the *RGML Gaussian* is always the best performing algorithm among those tested. Also, the *RGML Tyler* always comes close with a maximum discrepancy of 0.26% versus the *RGML Gaussian*. Then, the *RGML Tyler* is the best performing algorithm when the mislabeling rate is 5% or 10%. When the mislabeling rate is 15%, *RGML Tyler* is the best performing algorithm for the *Vehicle* dataset and it is only beaten by *ITML - Identity* on the two other datasets. This shows the interest of considering robust cost functions such the Tyler's cost function (16) in the presence of poor labeling.

Finally, the *RGML* algorithms are fast. Indeed, Figure 1 shows that both *RGML Gaussian* and *RGML Tyler* converge in less than 20 iterations on the *Wine* dataset.

V. Conclusions

This paper has proposed to view some classical *metric learning* problems as covariance matrix estimation problems. From this point of view, the *RGML* optimization problem has been formalized. It aims at estimating regularized covariance matrices, in a robust manner, while computing their Riemannian mean. The formulation is broad and several more specific costs functions have been studied. The first one leverages the classical Gaussian likelihood and the second one the Tyler's cost function. In both cases, the *RGML* problem is g-convex and thus any local minimizer is a global one. Two Riemannian-based optimization algorithms are proposed to minimize these cost functions. Finally, the performance of the proposed approach is studied on several datasets. They improve the classification accuracy and are robust to mislabeled data.

References

[1] J. L. Suárez, S. García, and F. Herrera, "A tutorial on distance metric learning: Mathematical foundations, algorithms, experimental analysis, prospects and challenges," *Neurocomputing*, vol. 425, pp. 300–322, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231220312777

[2] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell, "Distance metric learning, with application to clustering with side-information," in *Advances in Neural Information Processing Systems 15*. MIT Press, 2003, pp. 505–512.

[3] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon, "Information-theoretic metric learning," in *Proceedings of the 24th International Conference on Machine Learning*, ser. ICML '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 209–216. [Online]. Available: https://doi.org/10.1145/1273496.1273523

[4] P. H. Zadeh, R. Hosseini, and S. Sra, "Geometric mean metric learning," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ser. ICML'16. JMLR.org, 2016, p. 2464–2471.

[5] X. Yuan, W. Huang, P.-A. Absil, and K. A. Gallivan, *Averaging Symmetric Positive-Definite Matrices*. Cham: Springer International Publishing, 2020, pp. 555–575. [Online]. Available: https://doi.org/10.1007/978-3-030-31351-7_20

[6] E. Ollila, I. Soloveychik, D. E. Tyler, and A. Wiesel, "Simultaneous penalized m-estimation of covariance matrices using geodesically convex optimization," 2016.

[7] L. T. Skovgaard, "A riemannian geometry of the multivariate normal model," *Scandinavian Journal of Statistics*, vol. 11, no. 4, pp. 211–223, 1984. [Online]. Available: http://www.jstor.org/stable/4615960

[8] X. Pennec, P. Fillard, and N. Ayache, "A riemannian framework for tensor computing," *International Journal of Computer Vision*, vol. 66, pp. 41–66, 2005.

[9] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization algorithms on matrix manifolds*. Princeton University Press, 2008.

[10] D. E. Tyler, "A Distribution-Free M-Estimator of Multivariate Scatter," *The Annals of Statistics*, vol. 15, no. 1, pp. 234 – 251, 1987. [Online]. Available: https://doi.org/10.1214/aos/1176350263

[11] N. Boumal, "An introduction to optimization on smooth manifolds," To appear with Cambridge University Press, Jan 2022. [Online]. Available: http://www.nicolasboumal.net/book

[12] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: composable transformations of Python+NumPy programs," 2018. [Online]. Available: http://github.com/google/jax

[13] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[15] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *The Journal of Machine Learning Research*, vol. 10, pp. 207–244, 2009.

[16] W. de Vazelhes, C. Carey, Y. Tang, N. Vauquier, and A. Bellet, "metric-learn: Metric Learning Algorithms in Python," *Journal of Machine Learning Research*, vol. 21, no. 138, pp. 1–6, 2020.