A Comparative Study of Supervised Learning Algorithms for Symmetric Positive Definite Features

Ammar Mian*, Elias Raninen* and Esa Ollila*
* Aalto University, Dept. of Signal Processing and Acoustics, P.O. Box 15400, FI-00076 Aalto, Finland
E-mails: ammar.mian@aalto.fi, elias.raninen@aalto.fi, esa.ollila@aalto.fi

***Abstract*—In recent years, the use of Riemannian geometry has reportedly shown an increased performance for machine learning problems whose features lie in the symmetric positive definite (SPD) manifold. The present paper aims at reviewing several approaches based on this paradigm and provide a reproducible comparison of their output on a classic learning task of pedestrian detection. Notably, the robustness of these approaches to corrupted data will be assessed.**

***Index Terms*—Supervised learning; Riemannian geometry; Covariance matrix; Pedestrian detection;**

I. Introduction

Symmetric positive definite (SPD) matrices are central in many machine learning problems. For example, in image processing, the features of an image can be encoded using a *covariance descriptor*, which is a SPD matrix. Covariance descriptors have proven to be powerful in classification problems such as in pedestrian detection [1], [2], [3], or in EEG classification tasks [4], and [5].

SPD matrices are defined as the set $\mathbb{S}_{+}^{p} := \{\mathbf{A} \in \mathbb{R}^{p\times p} : \mathbf{A} = \mathbf{A}^{\top}, \mathbf{A} \succ \mathbf{0}\}$, where $\mathbf{A} \succ \mathbf{0}$ means $\mathbf{x}^{\top}\mathbf{A}\mathbf{x} > 0$ for all $\mathbf{x} \neq \mathbf{0}$. The set $\mathbb{S}_{+}^{p}$ forms an open convex half-cone in the Euclidean space $\mathbb{R}^{p\times p}$. For example, all covariance and correlation matrices are SPD matrices.

Since most prominent Machine Learning (ML) algorithms are designed for vector-valued features, the most straightforward approach of handling SPD features would be to vectorize them. Although this approach is simple and straightforward, it omits the underlying structure of this set which is a non-Euclidean one. Indeed, the set of SPD matrices is not a vector space. Rather, it admits the structure of a smooth manifold, and hence, the tools of Riemannian geometry can be applied. Viewing the set $\mathbb{S}_{+}^{p}$ as a manifold rather than as a subset of the Euclidean space has been fruitful and has led to improvements in the performance in many machine learning tasks.

Specifically, in supervised machine vision tasks such as pedestrian detection, algorithms that leverage on non-Euclidean distances on $\mathbb{S}_{+}^{p}$ have been developed. These studies have shown an increased classification accuracy compared to other simpler Euclidean algorithms. Unfortunately, so far there has not been a thorough study comparing the accuracy and computational efficiency of these methods. This is due to several reasons. In the papers (e.g., [1] and [3]), different preprocessing steps are applied to the data. This together with the fact that there is no publicly available code for the methods, makes reproducibility of the results difficult.

This paper aims to provide a framework for such comparisons on the pedestrian detection task with Riemannian based algorithms on covariance matrices. Another aim is to review some classical supervised ML algorithms that have been adapted to take into account the geometry of the SPD feature space. We will describe the algorithms based on Euclidean feature spaces and show how they have been modified to work with SPD matrices. A numerical comparison will shed some light on how much the performance is improved when taking into account the manifold structure of the data. Furthermore, we will examine how well the different methods perform when increasing the amount of data that is corrupted.

The rest of the paper is organized as follows. We briefly discuss the geometry of $\mathbb{S}_{+}^{p}$ in Section II and provide the main definitions used in the remainder of the paper. In section III, we formalize the classification problem and review the used learning algorithms. These algorithms are then compared on a pedestrian classification task in section IV. A robustness study with a discussion is performed in this section.

II. Geometry of the SPD manifold

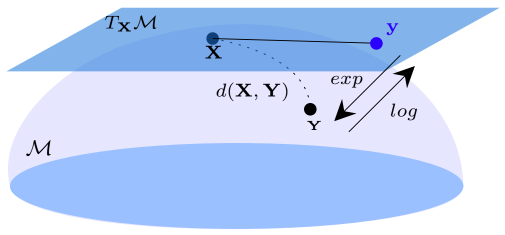

Fig. 1. Illustration of Riemannian geometry concepts.

The open convex cone of SPD matrices $\mathbb{S}_{+}^{p}$ admits a differentiable (smooth) Riemannian manifold structure $(\mathcal{M}, g)$, where $\mathcal{M} = \mathbb{S}_{+}^{p}$ and g is a *Riemannian metric*, i.e., an inner product defined on the tangent space $T_{\mathbf{\Sigma}}\mathcal{M}$ of every point $\mathbf{\Sigma}$ of $\mathcal{M}$ such that for the smooth vector fields $\mathbf{A}, \mathbf{B} : \mathcal{M} \to T\mathcal{M}$, the inner product is smoothly varying in the sense that $\mathbf{\Sigma} \mapsto g(\mathbf{A}_{\mathbf{\Sigma}}, \mathbf{B}_{\mathbf{\Sigma}})_{\mathbf{\Sigma}} \equiv \langle \mathbf{A}_{\mathbf{\Sigma}}, \mathbf{B}_{\mathbf{\Sigma}}\rangle_{\mathbf{\Sigma}}$ is a smooth function on $\mathcal{M}$, where $\mathbf{A}_{\mathbf{\Sigma}}, \mathbf{B}_{\mathbf{\Sigma}} \in T_{\mathbf{\Sigma}}\mathcal{M}$ denote the smooth vector fields evaluated at the point $\mathbf{\Sigma}$ [6].

Let $\mathbf{X} \in \mathbb{S}_{+}^{p}$ and $\mathbf{y} \in T_{\mathbf{X}}\mathbb{S}_{+}^{p}$. The *exponential map* $\mathbf{y} \mapsto \exp_{\mathbf{X}}(\mathbf{y}) : T_{\mathbf{X}}\mathbb{S}_{+}^{p} \to \mathbb{S}_{+}^{p}$ maps the point $\mathbf{y}$ from the tangent space to a point $\mathbf{Y}$ on the manifold. The length $\|\mathbf{y}\|_{\mathbf{X}} = \sqrt{\langle \mathbf{y}, \mathbf{y}\rangle_{\mathbf{X}}}$ is called the geodesic distance and denoted $d(\mathbf{X}, \mathbf{Y})$. Conversely, given two points on the manifold

$\mathbf{X}, \mathbf{Y} \in \mathbb{S}_+^p$, one can map $\mathbf{Y}$ to the tangent space $T_\mathbf{X}\mathbb{S}_+^p$ through the *logarithmic map* $\log_\mathbf{X} : \mathbb{S}_+^p \to T_\mathbf{X}\mathbb{S}_+^p$. These concepts are illustrated in Figure 1.

Let $\mathbf{X}, \mathbf{Y} \in \mathbb{S}_+^p$. Depending on the choice of the metric, different mappings and distances are obtained[1]:

- the *Euclidean metric*, which omits the Riemannian structure of the SPD manifold is defined as:

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

- the *log-Euclidean* metric leads to the following geodesic distance:

$$d_L(\mathbf{X}, \mathbf{Y}) = \|\log(\mathbf{X}) - \log(\mathbf{Y})\|_\mathrm{F}, \tag{2}$$

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

$$\log(\mathbf{X}) = \mathbf{U}\log_\odot(\mathbf{\Lambda})\mathbf{U}^\top, \tag{3}$$

  with $\log_\odot$ being the point-wise logarithm function.
- the *affine-invariant metric* (*natural metric*) leading to the geodesic distance:

$$d_A(\mathbf{X}, \mathbf{Y}) = \|\log(\mathbf{X}^{-1/2}\mathbf{Y}\mathbf{X}^{-1/2})\|_\mathrm{F}. \tag{4}$$

Many ML algorithms necessitate the computation of the mean of the feature vectors which is traditionally done by using a simple arithmetic mean. In a geometric context of SPD matrices, it is possible to consider a more general definition of the mean that takes into account the geometric properties of the feature space. Given an arbitrary distance d, it is possible to compute a weighted mean of a set of SPD matrices $\{\mathbf{M}_i : 1 \leq i \leq N\}$ with weights $\{\alpha_i : 1 \leq i \leq N\}$ through the optimization problem[2]:

$$\mathbf{\Pi}\left(\{\alpha_i, \mathbf{M}_i\}_{1 \leq i \leq N}\right) = \underset{\mathbf{M} \in \mathbb{S}_+^p}{\operatorname{argmin}} \sum_{i=1}^N \alpha_i d^2(\mathbf{M}, \mathbf{M}_i). \tag{5}$$

Thus we can define three different means based upon the three distances (d_E, d_L, and d_A) presented earlier:

- the Euclidean mean:

$$\mathbf{\Pi}_E\left(\{\alpha_i, \mathbf{M}_i\}_{1 \leq i \leq N}\right) = \sum_{i=1}^N \alpha_i \mathbf{M}_i \tag{6}$$

- the log-Euclidean mean:

$$\mathbf{\Pi}_L\left(\{\alpha_i, \mathbf{M}_i\}_{1 \leq i \leq N}\right) = \exp\left(\sum_{i=1}^N \alpha_i \log(\mathbf{M}_i)\right), \tag{7}$$

  where the $\exp$ is defined analogously to the logarithm of a matrix (see (3)) with a point-wise exponential function in place of the logarithm.
- the affine-invariant Riemannian mean which does not have a closed form expression but can be obtained through the iterative procedure [8] presented in Algorithm 1 which is similar to a gradient descent.

[1]For the sake of clarity, only the metrics used in this paper are defined. For a comprehensive survey, see [7].

[2]When the weights α_i are all equal, $\mathbf{\Pi}$ is called the Karcher mean.

```
$\mathbf{\Pi}_0 \leftarrow \mathbf{\Pi}_E(\{\alpha_i, \mathbf{M}_i\}_{1 \leq i \leq N})$, $k \leftarrow 0$
**while** *not converged* **do**
    $\mathbf{\Pi}_{k+1} \leftarrow \mathbf{\Pi}_k^{\frac{1}{2}} \exp\left(\gamma \sum_{i=1}^N \alpha_i \log\left(\mathbf{\Pi}_k^{-\frac{1}{2}} \mathbf{M}_i \mathbf{\Pi}_k^{-\frac{1}{2}}\right)\right) \mathbf{\Pi}_k^{\frac{1}{2}}$
    $k \leftarrow k + 1$
**end**
$\mathbf{\Pi}_A\left(\{\alpha_i, \mathbf{M}_i\}_{1 \leq i \leq N}\right) \leftarrow \mathbf{\Pi}_{k+1}$
```

Algorithm 1: Weighted affine-invariant Riemannian mean. Parameter $\gamma > 0$ is the learning rate.

The different tools based on Riemannian geometry presented in this section will enable developing algorithms for ML classification problems that obtain better performance over conventional Euclidean methods due to leveraging on the geometry of the SPD matrix manifold. This is explored in the next section.

III. Classificaton on the SPD manifold

We consider a supervised learning problem where we have a set of training data $\{(\mathbf{X}_i, y_i) \in \mathbb{S}_+^p \times \mathcal{C} : 1 \leq i \leq N\}$, where $\mathcal{C}$ is a discrete set of class labels. For a binary classification task $\mathcal{C} = \{-1, 1\}$ while for a multi-class problem with N_C classes, $\mathcal{C} = \{0, \ldots, N_c - 1\}$. The training data is used to find a decision function (classifier) $f : \mathbb{S}_+^p \to \mathcal{C}$ which outputs a class label $y \in \mathcal{C}$ to the given input data $\mathbf{X} \in \mathbb{S}_+^p$ with the best accuracy possible.

In traditional ML problems, where the input data is from a vector space, called *feature space*, the decision function splits the feature space into disjoint decision regions. For example, the Support Vector Machine (SVM) algorithm finds a hyperplane separating the feature space for a binary classification problem. In the case of SPD matrices, which do not form a vector space, some adaptions to the conventional ML algorithms are needed. Splitting the SPD manifold into decision regions is much more complicated. In order to tackle this problem several approaches that leverage the tools provided by the Riemannian geometry of SPD matrices have been considered in the literature.

An often-used solution to deal with SPD features is to map the data via the logarithmic mapping to a common tangent space which is a vector space. This method, however, requires that the mapped SPD matrices and the reference point of the tangent space are close enough to each other such that the relative distances on the tangent space are not too different from the actual geodesic distances. In [5] it was proposed to use this approach with the affine-invariant metric, d_A, which leads to the following logarithmic mapping:

$$\log_\mathbf{X}(\mathbf{Y}) = \mathbf{X}^{\frac{1}{2}} \log(\mathbf{X}^{-\frac{1}{2}}\mathbf{Y}\mathbf{X}^{-\frac{1}{2}})\mathbf{X}^{\frac{1}{2}}. \tag{8}$$

The reference point $\mathbf{X}$ is important since it defines the tangent space we are working on. A good solution in practice is to consider the Riemannian mean $\mathbf{\Pi}_A\left(\{\alpha_i, \mathbf{X}_i\}_{1 \leq i \leq N}\right)$ with weights $\alpha_i = 1/N$.

Then, given a classification procedure on a vector feature space $f : \mathbf{x} \in \mathbb{R}^d \to \mathcal{C}$, its Riemannian variant is obtained by

$$f \circ \log_{\mathbf{\Pi}_A(\{1/N, \mathbf{X}_i\}_{1 \leq i \leq N})} : \mathbb{S}_+^p \to \mathcal{C}. \tag{9}$$

In this paper, we will consider the use of a logistic regression on the tangent space for the pedestrian detection task. Next, we will explain the classification methods compared in this paper.

A. Boosting

Boosting [9] is a popular classification technique that has shown impressive results. Boosting. combines several simple classifiers $\{f_l : 1 \leq i \leq L\}$, called *weak learners*,in order to obtain a strong learner with good accuracy. A weak learner is a learning algorithm that is efficient to compute but has low or mediocre accuracy. Depending on the strategy, several algorithms have been proposed [9]. Among them, the logit-boost algorithm has been considered both for vector-valued and SPD-valued features in [1].

Let us consider a binary classification case[3], where $f_l(\mathbf{x}) \in \{-1, 1\}$. Boosting the weak learners by minimizing a negative binomial log-likelihood of the data using a technique called forward stagewise additive modeling. The probability for feature vector $\mathbf{x}$ of being in class 1 is represented by:

$$p(\mathbf{x}) = \frac{\exp\left(F(\mathbf{x})\right)}{\exp\left(F(\mathbf{x})\right) + \exp\left(-F(\mathbf{x})\right)}, \tag{10}$$

where $F(\mathbf{x}) = \frac{1}{2}\sum_{l=1}^{L} f_l(\mathbf{x})$. The final combined classifier for a sample $\mathbf{x}$ is then $\text{sign}[F(\mathbf{x})]$.

Given a training set $\{(\mathbf{x}_i, y_i) : 1 \leq i \leq N\}$, fitting the weak learners $\{f_l : 1 \leq l \leq L\}$ can be done by the optimization procedure presented in Algorithm 2.

```
Set $w_i \leftarrow 1/N$, $1 \leq i \leq N$ and $F(\mathbf{x}) \leftarrow 0$
for $l = 0$ to $L$ do
    Compute for $1 \leq i \leq N$:
        $z_i \leftarrow \frac{y_i - p(\mathbf{x}_i)}{p(\mathbf{x}_i)(1 - p(\mathbf{x}_i))}$,
        $w_i \leftarrow p(\mathbf{x}_i)(1 - p(\mathbf{x}_i))$
    Fit the function $f_l(\mathbf{x})$ by a weighted least squares
      regression of $z_i$ to $\mathbf{x}_i$ using weights $w_i$
    Set $F(\mathbf{x}) \leftarrow F(\mathbf{x}) + \frac{1}{2} f_l(\mathbf{x})$ and $p(\mathbf{x})$ of (10)
end
```

Algorithm 2: Logitboost training algorithm.

In order to use SPD matrices $\mathbf{X}_i$ as inputs (rather than vectors $\mathbf{x}_i$), [1] proposed to map the data to the tangent space of its weighted mean at each iteration. This can be done by replacing all the $\mathbf{x}_i$ in Algorithm 2 by

$$\log_{\mathbf{\Pi}_A(\{w_i, \mathbf{X}_i\}_{1 \leq i \leq N})}(\mathbf{X}_i),$$

[3]The multiclass case is obtained by adapting Algorithm 6 of [9] to this setup.

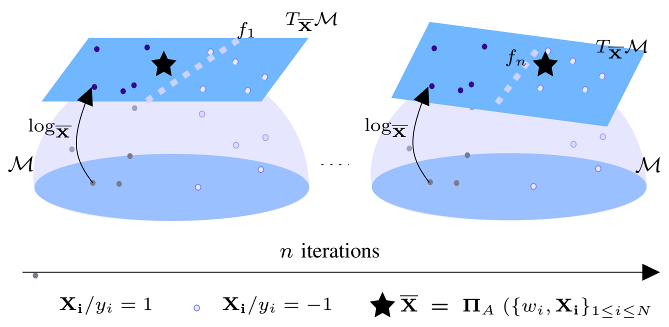
f_1
$T_{\overline{\mathbf{X}}}\mathcal{M}$
$\log_{\overline{\mathbf{X}}}$
$\mathcal{M}$
f_n
$T_{\overline{\mathbf{X}}}\mathcal{M}$
$\log_{\overline{\mathbf{X}}}$
$\mathcal{M}$
n iterations
$\mathbf{X_i}/y_i = 1$
$\mathbf{X_i}/y_i = -1$
$\overline{\mathbf{X}} = \mathbf{\Pi}_A\left(\{w_i, \mathbf{X_i}\}_{1 \leq i \leq N}\right)$

Fig. 2. Illustration of logitboost algorithm. Since one sample was misclassified, the Riemannian mean shifts towards it.

where the logarithm mapping is defined in (8). Using this approach, the Riemannian mean is updated at each iteration and it shifts toward the misclassified samples. An illustration of this is shown in Figure 2. The weak learners used in the simulations of this paper are decision stumps (one-level decision trees).

B. Riemannian Gaussian Kernel

While the Riemannian logitboost allows leveraging the structure of $\mathbb{S}_+^p$, it is computationally expensive as it requires the computation of both the Riemannian mean of the samples as well as the mapping the data to the tangent space which are computationally demanding operations. Moreover, the distances on the manifold are approximated in the tangent space which is not ideal when the data for a problem spans a large area of the manifold.

A computationally lighter approach was proposed in [3], which used kernel methods in order to embed the original data to a higher dimensional Reproducing Kernel Hilbert Space (RKHS) while keeping the true distances on the manifold. Kernels methods have been successful in many ML tasks that are not based on vector feature space [10]. In [3], the authors adapt the popular Gaussian (or Radial Basis Function (RBF)) kernel to features in $\mathbb{S}_+^p$ by exploiting the log-Euclidean distance[4], and define the kernel function as:

$$k_{\text{G}}(\mathbf{X}, \mathbf{Y}) = \exp\left(-\gamma d_L(\mathbf{X}, \mathbf{Y})\right), \tag{11}$$

where $\gamma > 0$ is a tuning parameter. The authors showed that this kernel is positive definite, so it satisfies Mercer's theorem. This means that in practice the knowledge of the Gram matrix $K_{ij} = k_{\text{G}}(\mathbf{X}_i, \mathbf{X}_j)$ is sufficient for most algorithms. In the simulations, we consider the use of the SVM algorithm with the Riemannian Gaussian kernel for pedestrian detection.

The parameter γ largely depends on the training data distribution, and hence, it can be obtained by cross-validation. Otherwise, we propose to consider the variance of the data which in the Riemannian context leads to:

$$\gamma = N\left[\sum_{i=0}^{N} d_L^2\left(\mathbf{X}_i, \mathbf{\Pi}_L\left(\{1/N, \mathbf{X}_i\}_{1 \leq i \leq N}\right)\right)\right]^{-1}. \tag{12}$$

[4]Up to our best knowledge, this the only work allowing to obtain a kernel based on a true geodesic distance and that is positive definite. For other kernels, see for example [11].

C. *Classification based on distance*

Finally, one simple way to obtain a classifier that takes advantage of the Riemannian framework is to consider a learning algorithm based on distances such as the Minimum Distance to Mean (MDM) of the k-Nearest Neighbors (k-NN).

In the case of MDM, the Riemannian mean of each class of $\mathcal{C}$ is computed during the training phase and the classification is done by assigning each sample to the class of the closest mean. For the case of k-NN, the training samples are kept in memory and the classification is performed by computing the distance of the test sample to each training sample and assigning the majority class of the k closest training samples. This approach is computationally very heavy since it requires a huge number of comparisons depending on the size of the training set. However, these algorithms have reported good results for BCI classification tasks [5].

IV. Numerical experiments

In this section, we consider numerical examples to illustrate the classification algorithms presented in this paper.

A. *Reproducibility matters*

In order to provide a framework for reproducible results on Riemannian-based methods for the pedestrian task, Python (3.7) codes for this section are made publicly available on https://github.com/AmmarMian/Comparative_study_pedestrian_Eusipco. The coding procedure mainly relied on the python package *scikit-learn* [12] as well as useful functions from *pyriemann* package developed by [5]. Two different datasets have been considered for the study: the INRIA person dataset [13] as well as the DaimerChrysler benchmark dataset [14].

B. *Pedestrian detection using covariance descriptors*

The pedestrian detection task is a binary classification problem where the aim is to decide if a person is present or not in a given window of an image. [1] proposed to use covariance descriptors as features to represent each window and then do classification based on them. Since typically images are monochromatic, the image is first pre-processed to obtain a feature vector for each pixel. For an image with pixel locations, (x, y), and intensity derivatives, I_x, I_y, one defines the feature vectors as [1]:

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

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

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

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

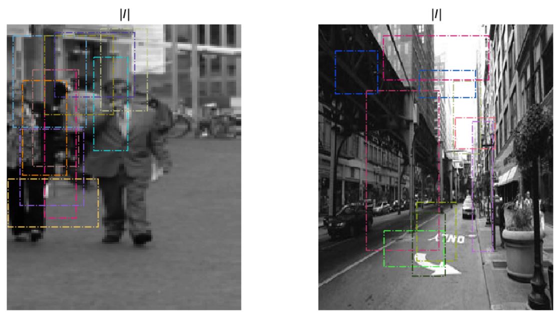

Fig. 3. Examples of positive (left) and negative (right) training images of the INRIA dataset. The regions to compute covariance descriptors are shown with colored boxes.

C. *Methodology*

1) Pre-processing: In order to obtain various training samples, windows are sampled for positive and negative images using the methodology described in [3]. An example of sampling is shown for positive and negatives images in Figure 3. To enhance robustness to illumination changes between the images, the covariance matrices are normalized by the covariance of the whole image and the final covariance of the window is obtained through $\hat{\mathbf{C}}_W = \mathrm{diag}(\mathbf{C})^{-1/2}\mathbf{C}_W\ \mathrm{diag}(\mathbf{C})^{-1/2}$, where $\mathbf{C}$ is the covariance matrix of the whole image.

We used 10 sampling windows for each positive and negative image of the INRIA dataset leading to 35480 positive samples and 12120 negative samples. For the DaimerChrysler dataset, 2 sampling regions were used for each image leading to 49000 positive and 48000 negative samples.

2) Simulation setup: The methodology recommended by [14] has been used here. K-fold cross-validation was used on the training set to do cross-validation and obtain the best hyperparameters. Then the classifiers have been trained again on the same K-fold training set leading to K different classifiers that have been applied to the testing set to measure accuracy. K was set to 4 for the INRIA dataset and to 3 for the DaimerChrysler one.

D. *Results*

The accuracy of classification are reported in Tables I and II. Several observations can be made:

- Riemannian variant of the algorithms tested appear to have in general better results compared to their Euclidean counterparts for both datasets. The case of the kernel methods is peculiarly striking in that regards. This illustrates how the Riemannian framework allows to improve classification on covariance matrices.
- The MDM algorithms do not perform well since they only rely only on the mean of each class which doesn't seem to be a good decision criterion for classification.
- The Riemannian logitboost and kernel methods appear to be the best methodologies depending on the dataset considered. Since the kernel approach is much more computationally efficient, it is the most attractive one.

TABLE I
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

TABLE II
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

E. Robustness study

We also consider the robustness of the algorithms with regards to outliers in the dataset. To that end, we consider a subset of a 1000 matrices from the DaimerChrysler dataset (800 for training and 200 for testing) and replace a certain amount of the training samples α of those samples with random outliers generated through a Wishart distribution with an arbitrary Toeplitz covariance parameter. We increase the number of outliers from 0% to 10% and perform 100 different experiments for each value of α. We obtain the plots in Figure 4 by averaging the accuracy over the experiments.

From this, we can observe that additionally to having better performance, Riemannian methodologies tend to have a better better behaviour in presence of outliers. Indeed, the accuracy of the kernel, MDM and logistic regression decrease slower than their Euclidean counterparts. For MDM and logistic regression, this is explained by the fact that they perform using the Riemannian mean over the training samples which is known to be more robust to outliers. For the kernel approach, by taking into account the geometry of the manifold, the distances between the outliers and the accurate training samples are usually high which means that for example in the case of the kernel approach, they can't correspond to support vectors. Concerning KNN approaches, the robustness is explained by the same fact: outliers are never the nearest neighbors.

ACKNOWLEDGMENT

We acknowledge the computational resources provided by the Aalto Science-IT project. This work was supported by Academy of Finland grant No. 298118.

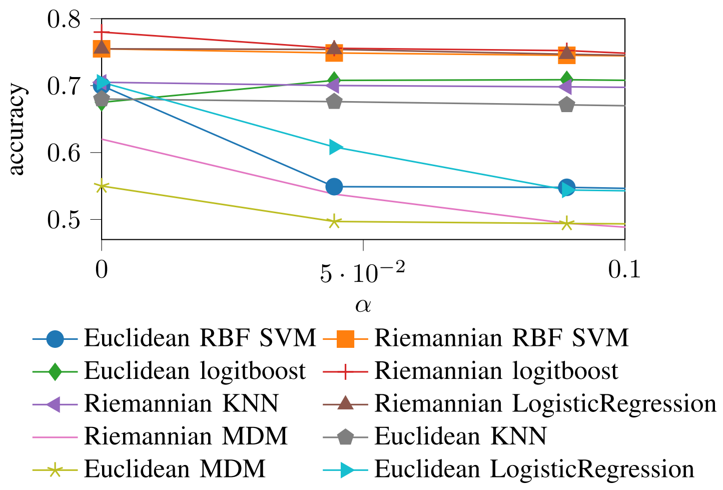

Fig. 4. Accuracies with regards to percent of outliers $100 \times \alpha$. $N = 1000$

REFERENCES

[1] O. Tuzel, F. Porikli, and P. Meer, "Pedestrian detection via classification on riemannian manifolds," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 10, pp. 1713–1727, 10 2008.

[2] M. San-Biagio, M. Crocco, M. Cristani, S. Martelli, and V. Murino, "Low-level multimodal integration on riemannian manifolds for automatic pedestrian detection," in *2012 15th International Conference on Information Fusion*, 7 2012, pp. 2223–2229.

[3] S. Jayasumana, R. Hartley, M. Salzmann, H. Li, and M. Harandi, "Kernel Methods on the Riemannian Manifold of Symmetric Positive Definite Matrices," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 6 2013, pp. 73–80.

[4] M. Congedo, A. Barachant, and R. Bhatia, "Riemannian geometry for EEG-based brain-computer interfaces; a primer and a review," *Brain-Computer Interfaces*, vol. 4, no. 3, pp. 155–174, 7 2017.

[5] A. Barachant, S. Bonnet, M. Congedo, and C. Jutten, "Multiclass Brain–Computer Interface Classification by Riemannian Geometry," *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 4, pp. 920–928, 4 2012.

[6] L. W. Tu, *Differential geometry: connections, curvature, and characteristic classes*. Springer, 2017, vol. 275.

[7] X. Pennec, "Manifold-valued image processing with spd matrices," in *Riemannian Geometric Statistics in Medical Image Analysis*, X. Pennec, S. Sommer, and T. Fletcher, Eds. Academic Press, 2020, pp. 75 – 134.

[8] B. Jeuris, R. Vandebril, and B. Vandereycken, "A survey and comparison of contemporary algorithms for computing the matrix geometric mean," *Electronic Transactions on Numerical Analysis*, vol. 39, no. ARTICLE, pp. 379–402, 2012.

[9] J. Friedman, T. Hastie, R. Tibshirani *et al.*, "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)," *The annals of statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[10] J. Shawe-Taylor, N. Cristianini *et al.*, *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[11] F. Yger, "A review of kernels on covariance matrices for BCI applications," in *2013 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 9 2013, pp. 1–6.

[12] 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.

[13] N. Dalal and B. 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.

[14] S. Munder and D. M. Gavrila, "An experimental study on pedestrian classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 11, pp. 1863–1868, 11 2006.