SPDNet-AE: a Compact SPD Representation through Riemannian Autoencoding

Charlotte Boucherie[1], Thibault de Surrel[2], Florian Yger[1] *

1- LITIS, INSA Rouen-Normandy, France

2- LAMSADE, CNRS, PSL Univ. Paris-Dauphine, France

Abstract. When building dimension reduction methods tailored for Symmetric Positive Definite (SPD) matrices, it is crucial to account for their Riemannian geometry. In this work, we propose an SPDNet-based autoencoder, that we call *SPDNet-AE*, that learns low-dimensional SPD representations of high-dimensional SPD matrices while preserving the geometry throughout the network. The SPDNet-AE is built using the BiMap layer of the SPDNet, but we allow it to have multiple channels. We show that our SPDNet-AE is able to learn a useful low-dimensional representation of the data for classification (without any class information). Moreover, we show that with a comparable number of parameters, a classical Euclidean autoencoder is not able to learn and maintain the SPD constraint on the input matrices.

1 Introduction

Respecting the intrinsic geometry of the data is crucial when dealing with structured data [1]. This is especially true when dealing with covariance matrices, which are Symmetric Positive Definite (SPD) matrices and form a Riemannian manifold of negative curvature. This space being non-flat, using classical Euclidean methods can lead to suboptimal results or to distortion issues (such as the swelling effect [2]). Covariance matrices, and more generally SPD matrices, are widely used in machine learning [3] as well as in many applications, such as in fault detection [4], biomedical image analysis [5] or Brain Computer Interfaces (BCI) [6] and being able to reduce the dimensionality of these data in a principled way would benefit these fields.

The goal of this work is to propose a method that is able to reduce the dimension of SPD matrices, while retaining the relevant information and the geometry of the data. To do so, we propose to use a geometry-aware AutoEncoder, that we call *SPDNet-AE*. For this, we will use the SPDNet [7], a Riemannian network that preserves the SPD nature of the input. Let us introduce the specifications for our SPDNet-AE:

1. We want to reduce the dimension of the input SPD matrices while preserving the structure of the data. More specifically, we want our low-dimensional (LD) representation of the data to remain SPD, and we want to ensure that the reconstructed matrices are also SPD.

*This work was funded by the French National Research Agency for project PROTEUS (grant ANR-22-CE33-0015-01).

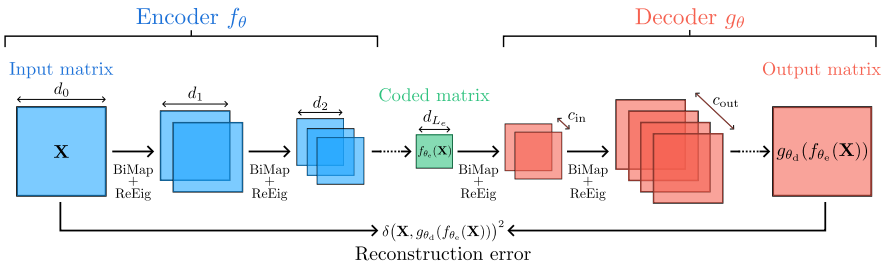

Fig. 1: Architecture of the SPDNet Autoencoder. The encoder f and the decoder g are composed of a succession of BiMaps, each followed by a ReEig. The encoder reduces the dimension of the input SPD matrices while the decoder increases it. Each layer can have a different number of input or output channels. The architectures of the encoder and of the decoder are not necessarily symmetric.

2. We aim to learn a useful LD representation that still allows accurate classification on the resulting SPD matrices.

We will introduce the Riemannian geometry used for SPD matrices and the Riemannian network SPDNet in Section 2. In Section 3, we introduce our SPDNet-AE and its architecture. We finally test our SPDNet-AE and compare it to other dimension reduction algorithms on real datasets in Section 4.

2 SPD matrices and the SPDNet

2.1 The manifold of SPD matrices

We consider the set $\mathcal{P}_d$ of $d \times d$ *Symmetric Positive Definite* (SPD) matrices defined as follows:

$$\mathcal{P}_d = \{\boldsymbol{X} \in \mathbb{R}^{d\times d} \mid \boldsymbol{X}^\top = \boldsymbol{X} \text{ and } \forall v \in \mathbb{R}^d \setminus \{0\}, v^\top \boldsymbol{X} v > 0\}.$$

This set can be endowed with a Riemannian dot product, leading to a Riemannian manifold of dimension $d(d+1)/2$ [5]. This Riemannian structure allows us to define a distance δ between two SPD matrices:

$$\forall \boldsymbol{X}, \boldsymbol{Y} \in \mathcal{P}_d,\ \delta(\boldsymbol{X}, \boldsymbol{Y}) = \|\log(\boldsymbol{X}^{-1/2}\boldsymbol{Y}\boldsymbol{X}^{-1/2})\|_F$$

with log the matrix logarithm and $\|\cdot\|_F$ the Frobenius norm.

2.2 The SPDNet

One of the pioneering works focusing on a Riemannian neural network for SPD matrices is SPDNet [7]. The specificity of this network is that it preserves the SPD nature of the input at each layer of the network. The main layers of the SPDNet are the linear-like *BiMap* layer and the ReLU-like eigenvalue rectification *ReEig* layer. Let us briefly introduce these two layers. We used the implementation of SPDNet proposed by the authors of [8].

The BiMap layer This linear layer projects the data into a new space using learnable parameters. It is defined as: $\boldsymbol{X}_k = f_b^{(k)}(\boldsymbol{X}_{k-1}; \boldsymbol{W}_k) = \boldsymbol{W}_k \boldsymbol{X}_{k-1} \boldsymbol{W}_k^\top$ where $\boldsymbol{X}_{k-1} \in \mathcal{P}_{d_{k-1}}$ is the SPD input matrix of the k-th layer, $\boldsymbol{W}_k \in \mathbb{R}^{d_k \times d_{k-1}}$ is the (learnable) projection matrix and $\boldsymbol{X}_k$ is the output of the layer. In order to preserve the SPDness of the output matrix $\boldsymbol{X}_k$, the authors propose to restrict the weight matrix $\boldsymbol{W}_k$ to the compact Stiefel manifold ($\boldsymbol{W}_k^\top \boldsymbol{W}_k = I_{d_{k-1}}$).

The ReEig layer The ReEig layer mimics the traditional ReLU function by adjusting the eigenvalues of the input SPD matrix $\boldsymbol{X}_k$ based on a threshold ϵ: $\boldsymbol{X}_k = f_r^{(k)}(\boldsymbol{X}_{k-1}) = \boldsymbol{U}_{k-1} \max(\epsilon \boldsymbol{I}_{d_{k-1}}, \boldsymbol{\Sigma}_{k-1}) \boldsymbol{U}_{k-1}^\top$ where $\boldsymbol{U}_{k-1}$ and $\boldsymbol{\Sigma}_{k-1}$ are obtained using the eigenvalue decomposition of $\boldsymbol{X}_{k-1}$: $\boldsymbol{X}_{k-1} = \boldsymbol{U}_{k-1} \boldsymbol{\Sigma}_{k-1} \boldsymbol{U}_{k-1}^\top$. The max operator is applied element-wise between the diagonal elements of the input matrix and the threshold ϵ. In all our experiments, we set $\epsilon = 10^{-4}$.

3 SPDNet Autoencoders

3.1 The architecture

The goal of our autoencoder is to learn an LD SPD representation, of the input SPD matrices. For this, our *SPDNet-AE* will be composed of an encoder and a decoder each composed of a succession of BiMap layers each followed by a ReEig layer. The encoder will reduce the dimension of the SPD matrices so the BiMaps weights will have more columns than rows. The decoder will increase the dimension of the SPD matrices so the BiMaps weights will have more rows than columns. The global architecture is illustrated in Figure 1. To allow more flexibility in the network, we add the possibility to have multiple channels in the BiMap layers. The encoder and decoder architectures are not necessarily symmetric, so depending on the data or application, they may have different numbers of layers. Allowing the BiMap layers to have multiple channels is especially useful for the decoder. Indeed, the trainable projection matrices $\boldsymbol{W}_k$ are constrained to belong to the Stiefel manifold, therefore we have that $\boldsymbol{W}_k^\top \boldsymbol{W}_k = I_{d_{k-1}}$. Thus, the coefficients of $\boldsymbol{W}_k$ will always be smaller than 1. This can be problematic when reconstructing SPD matrices with big coefficients, and adding the possibility of summing several outputs of BiMap layers (through multiple channels), will help such reconstruction and extend the expressivity of our network. The loss that will be used is the Riemannian reconstruction error, defined as:

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \delta\big(\boldsymbol{X}^{(i)}, g_{\theta_\mathrm{d}}(f_{\theta_\mathrm{e}}(\boldsymbol{X}^{(i)}))\big)^2 \tag{1}$$

where $\boldsymbol{X}^{(i)}$ is the i-th input sample, $g_{\theta_\mathrm{d}}(f_{\theta_\mathrm{e}}(\boldsymbol{X}^{(i)}))$ is the output of the autoencoder (f_{θ_e} is the encoder and g_{θ_d} the decoder), N is the number of samples and $\theta_\mathrm{e}, \theta_\mathrm{d}$ are the trainable parameters, respectively of the encoder and the decoder.

Dataset	Application	Dimension	# matrices	# classes	Reference	# params SPDNet-AE	# params Eucl. AE
Salinas	Hyperspectral imaging	16×16	15,000	6	-	22,352	23,008
Indiana Pines	Hyperspectral imaging	16×16	14,641	12	[9]	22,352	23,008
Textile	Image Analysis	10×10	16,000	2	[10]	3,464	2,837

Table 1: Summary of the datasets used for the experiments.

4 Testing the SPDNet-AE on real data

Setup We train and test our SPDNet-AE on 3 different datasets summarized in Table 1. We will compare our SPDNet-AE to the *Riemannian PCA* [11] and the *Principal Geodesic Analysis* (PGA) [12], two dimensionality reduction methods designed for SPD matrices. We will also compare our SPDNet-AE to a SPDNet-AE trained with a Euclidean loss and to a classical Euclidean autoencoder. For the SPDNet-AEs, the encoder consists of 3 or 4 multi-channels BiMap layers, each followed by a ReEig layer. The low-dimensional (LD) representation has half the size of the input SPD matrices. The decoder mirrors the encoder to reconstruct the input. The Euclidean autoencoders were designed to have roughly the same number of trainable parameters of the SPDNet-AE so that the comparison was fair. The number of trainable parameters for each autoencoder is reported in Table 1.

To assess the quality of the different pipelines, we will use several metrics. First, we will give the percent of output matrices that are SPD as we want the output of our SPDNet-AE to be SPD. Then, we will perform a classification task on the learned LD representations to see if they are useful for downstream tasks. The classifiers will be trained and tested on unseen data by the AE during its training. We will compare the accuracy of a given classifier on the orgininal HD data and on the LD data computed using the different pipelines. The classifiers used are a Minimum Distance to Mean (MDM) classifier [13] and a fully connected neural network. We used a 5-fold cross-validation procedure for each classifier. As the MDM is designed for SPD matrices, and the Euclidean-AE does not guarantee that the LD matrices are SPD, we will only use this classifier on the SPDNet-AEs and the Riemannian PCA.

Results As expected, the SPDNet-AE outputs 100% of SPD matrices, as it was build to do so. On the other hand, the Euclidean AE performs poorly producing 0% of SPD matrices for the datasets *Indiana* and *Textile* and only 18.58% for the dataset *Salinas*. Therefore, using a usual Euclidean AE on SPD matrices can lead to real problems as it fails to reconstruct the important structure of the data. This result confirms our choice of building a network that respects the geometry of the data using the SPDNet.

The results of the classification experiment on the different datasets are given at Figure 2. We recall that our goal is that the accuracy in the LD code space matches, as much as possible, the accuracy on the HD data (here in dark green). On can see that the SPDNet-AE, when trained using a Riemannian loss (here in orange), is able to learn an LD representation of the data that contains enough

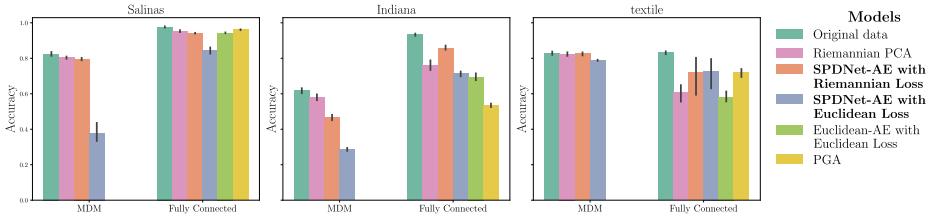

Fig. 2: Results of the classification experiment on the different datasets.

information to be correctly classified. Indeed, the classification results in the LD space nearly match the results in the HD space, despite a signification drop in the dimension of the data. When we change the training loss to a Euclidean one, the classification results (here in blue), significantly drops. This is coherent, as we are taking the geometry of the data into account in our network, but not anymore in the loss. Therefore, to correctly train the SPDNet-AE, we need to use the Riemannian loss. The Euclidean AE (here in light green) performs as good as the SPDNet-AE. However, we recall that the Euclidean AE does not guarantee that the output matrices are SPD. Compared to the Riemannian PCA (in pink) and the PGA (in yellow), the SPDNet-AE is competitive, as all those LD representations lead to similar classification results. However, the SPDNet-AE has the advantage of being inductive, meaning that once the network is trained, it can be used to encode new points that were not used during the training. This is not the case for the Riemannian PCA or the PGA.

5 Conclusion

In this paper, we introduced a geometry-aware dimension reduction method for SPD matrices using an SPDNet-based autoencoder: the SPDNet-AE. This SPDNet-AE is built using the classical layers of the SPDNet, that is BiMap and ReEig layers. We tested our SPDNet-AE and compared it to a Euclidean AE and to other dimension reduction algorithms on real datasets. We first showed that, with a comparable number of trainable parameters, the Euclidean AE is not able to correctly reconstruct the SPD matrices, while the SPDNet-AE is able to do so (by definition of our network). Then, we showed that the SPDNet-AE is able to learn an LD representation of the data that retains the most relevant part of the information, as the classification results in the LD space are comparable to the ones in the HD space.

One could argue that classical Euclidean neural networks, when built with enough learnable parameters, could be able to learn the geometry of SPD matrices. While we agree with this perspective, we have shown that it is possible to learn a low-dimensional SPD representation with far fewer parameters than Euclidean networks, requiring less data and computational effort. This is particularly important in applications where data and computing resources are limited. Geometry-aware networks thus offer an effective way to reduce both model size

and data requirements. Moreover, there are no guarantees that a deep model will be able to generate SPD matrices, while our model is guaranteed to do so. Finally, the use of SPD-AE in a denoising or a contrastive setup is also a promising lead towards an expressive model for metric learning on $\mathcal{P}_d$ and its application in challenging setups such as BCI [14].

References

[1] Sophia Sanborn, Johan Mathe, Mathilde Papillon, Domas Buracas, Hansen J Lillemark, Christian Shewmake, Abby Bertics, Xavier Pennec, and Nina Miolane. Beyond euclid: An illustrated guide to modern machine learning with geometric, topological, and algebraic structures. *arXiv preprint arXiv:2407.09468*, 2024.

[2] Vincent Arsigny, Pierre Fillard, Xavier Pennec, and Nicholas Ayache. Geometric Means in a Novel Vector Space Structure on Sysmetric Positive-Definite Matrices. *SIAM J. Matrix Analysis Applications*, 29:328–347, January 2006.

[3] Rui Wang, Xiao-Jun Wu, and Josef Kittler. Symnet: A simple symmetric positive definite manifold deep learning method for image set classification. *IEEE Transactions on Neural Networks and Learning Systems*, 33(5):2208–2222, 2021.

[4] Evan L. Russell, Leo H. Chiang, and Richard D. Braatz. Fault detection in industrial processes using canonical variate analysis and dynamic principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 51(1):81–93, 2000.

[5] Xavier Pennec. 3 - manifold-valued image processing with spd matrices. In Xavier Pennec, Stefan Sommer, and Tom Fletcher, editors, *Riemannian Geometric Statistics in Medical Image Analysis*, pages 75–134. Academic Press, 2020.

[6] F. Lotte, L. Bougrain, A. Cichocki, M. Clerc, M. Congedo, A. Rakotomamonjy, and F. Yger. A review of classification algorithms for EEG-based brain–computer interfaces: A 10 year update. *Journal of Neural Engineering*, 15(3):031005, April 2018.

[7] Zhiwu Huang and Luc Van Gool. A Riemannian Network for SPD Matrix Learning, December 2016.

[8] Daniel Brooks, Olivier Schwander, Frédéric Barbaresco, Jean-Yves Schneider, and Matthieu Cord. Second-order networks in PyTorch. In *Lecture Notes in Computer Science*, volume 11712 of *Lecture Notes in Computer Science*, pages 751–758, Toulouse, France, August 2019. Springer.

[9] Marion F. Baumgardner, Larry L. Biehl, and David A. Landgrebe. 220 band aviris hyperspectral image data set: June 12, 1992 indian pine test site 3, Sep 2015.

[10] Paul Bergmann, Kilian Batzner, Michael Fauser, David Sattlegger, and Carsten Steger. The mvtec anomaly detection dataset: A comprehensive real-world dataset for unsupervised anomaly detection. *Int. J. Comput. Vision*, April 2021.

[11] Inbal Horev, Florian Yger, and Masashi Sugiyama. Geometry-aware principal component analysis for symmetric positive definite matrices. *Machine Learning*, 106(4):493–522, April 2017.

[12] P.T. Fletcher, Conglin Lu, S.M. Pizer, and Sarang Joshi. Principal geodesic analysis for the study of nonlinear statistics of shape. *IEEE Transactions on Medical Imaging*, 23(8):995–1005, August 2004.

[13] Alexandre Barachant, Stephane Bonnet, Marco Congedo, and Christian Jutten. Riemannian geometry applied to BCI classification. In *LVA/ICA 2010 - 9th International Conference on Latent Variable Analysis and Signal Separation*, volume 6365, page 629. Springer, September 2010.

[14] Florian Yger, Maxime Berar, and Fabien Lotte. Riemannian Approaches in Brain-Computer Interfaces: A Review. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 25(10):1753–1762, October 2017.