Generalized Learning Vector Quantization With Log-Euclidean Metric Learning on Symmetric Positive-Definite Manifold

Fengzhen Tang, Peter Tiňo, and Haibin Yu, *Senior Member, IEEE*

***Abstract*—In many classification scenarios, the data to be analyzed can be naturally represented as points living on the curved Riemannian manifold of symmetric positive-definite (SPD) matrices. Due to its non-Euclidean geometry, usual Euclidean learning algorithms may deliver poor performance on such data. We propose a principled reformulation of the successful Euclidean generalized learning vector quantization (GLVQ) methodology to deal with such data, accounting for the nonlinear Riemannian geometry of the manifold through log-Euclidean metric (LEM). We first generalize GLVQ to the manifold of SPD matrices by exploiting the LEM-induced geodesic distance (GLVQ-LEM). We then extend GLVQ-LEM with metric learning. In particular, we study both 1) a more straightforward implementation of the metric learning idea by adapting metric in the space of vectorized log-transformed SPD matrices and 2) the full formulation of metric learning without matrix vectorization, thus preserving the second-order tensor structure. To obtain the distance metric in the full LEM learning (LEML) approaches, two algorithms are proposed. One method is to restrict the distance metric to be full rank, treating the distance metric tensor as an SPD matrix, and readily use the LEM framework (GLVQ-LEML-LEM). The other method is to cast no such restriction, treating the distance metric tensor as a fixed rank positive semidefinite matrix living on a quotient manifold with total space equipped with flat geometry (GLVQ-LEML-FM). Experiments on multiple datasets of different natures demonstrate the good performance of the proposed methods.**

***Index Terms*—Generalized learning vector quantization (GLVQ), log-Euclidean metric (LEM), metric learning, Riemannian geodesic distance, Riemannian manifold.**

Manuscript received December 26, 2021; revised April 8, 2022; accepted May 24, 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 61803369; in part by the State Key Laboratory of Robotics under Grant 2022-Z02; in part by the Foundation for Innovative Research Groups of the National Natural Science Foundation of China under Grant 61821005; and in part by the European Commission Horizon 2020 Innovative Training Network SUNDIAL under Project 721463. This article was recommended by Associate Editor D. Wang. *(Corresponding authors: Fengzhen Tang; Haibin Yu.)*

Fengzhen Tang and Haibin Yu are with the State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China, and also with the Institutes for Robotics and Intelligent Manufacturing, Chinese Academy of Sciences, Shenyang 110169, China (e-mail: tangfengzhen@sia.cn; yhb@sia.cn).

Peter Tiňo is with the School of Computer Science, University of Birmingham, Birmingham B15 2TT, U.K. (e-mail: p.tino@cs.bham.ac.uk).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCYB.2022.3178412.

Digital Object Identifier 10.1109/TCYB.2022.3178412

I. Introduction

LEARNING vector quantization (LVQ) is a simple but powerful classification scheme [1], [2]. It is attractive due to several reasons. First, the LVQ system, different from deep networks or kernel methods, is straightforward to interpret. They construct a classifier parametrized by several labeled prototypes that live in the same space as the data space. The classification rule follows a winner-takes-all scheme, that is, assigning a new instance the class label of its closest prototype according to the related distance. Second, the LVQ method is implemented and realized in an intuitive and simple way, since it follows the intuitive Hebbian learning rule. Third, LVQ can naturally handle any number of classes with no modification on the learning rule, which is different from many alternatives like support vector machines (SVMs), whose basic form is confined to deal with only two classes.

Generalized LVQ (GLVQ) proposed in [3] constructs the LVQ system using an explicit cost function that aims at margin maximization, showing good generalization performance. The cost function has been extended to incorporate metric learning. A simple adaptive diagonal matrix of relevance is considered in generalized relevance LVQ (GRLVQ) [1], allowing different scaling of the features according to their relevance. A global adaptive full matrix of relevance [parameterized general Euclidean metric (EM)] is used in generalized matrix LVQ (GMLVQ) [4], accounting for pairwise correlations of features. Furthermore, local adaptive matrices are constructed to attach to individual prototypes, leading to localized GMLVQ (LGMLVQ) [4]. These LVQ methods are designed to analyze vector-formed data living in the Euclidean space.

However, in many learning scenarios, the data to be analyzed are naturally symmetric positive-definite (SPD) matrices. For instance, in medical image analysis, diffusion-weighted images obtained by diffusion tensor magnetic resonance imaging (DT-MRI or simply DTI) are entirely characterized by the "diffusion tensors," which are essentially SPD matrices [5]–[7]; in computer vision, covariance region descriptors (also SPD matrices) are often used to characterize images [8], [9]; and in brain–computer interface (BCI), spatial covariance matrix (SPD matrix) representation becomes very popular in the classification of motor imagery electroencephalogram (EEG) data [10], [11]. Unlike usual vector-valued data distributed in the Euclidean space, SPD matrices can form a nonlinear Riemannian manifold if equipped with an

appropriate metric (e.g., affine-invariant Riemannian metric (AIRM) [5] or log-Euclidean metric (LEM) [12]). Due to the non-Euclidean geometry of Riemannian manifolds, classical Euclidean LVQ methods result in inferior performance on such manifold-valued data as shown in [13].

In this article, we provide two key contributions to prototype-based learning on Riemannian manifolds.

1) *Section IV:* While in [13], the core GLVQ method was extended to the manifold of SPD matrices equipped with AIRM (GLVQ-AIRM), here we generalize the GLVQ method to the manifold of SPD matrices by employing the LEM (GLVQ-LEM). Arsigny *et al.* [12] provided a good discussion and comparison of LEMs and other choices for defining the Riemannian structure on SPD matrices. In particular, while LEMs are not affine invariant, they do possess useful invariance properties. For example, the distances are not changed by the matrix inversion operation, logarithmic multiplication (translation in the log domain), or orthogonal transformation and scaling (similarity). Crucially, compared with affine-invariant metrics, the evaluation of geodesic distances under the LEM is much more computationally efficient. Thus, the methods developed in this article are expected to be much faster than GLVQ-AIRM.
2) *Section V:* Taking advantage of the Euclidean vector space structure of the log-transformed SPD matrices,[1] we further extend GLVQ-LEM with metric learning. Indeed, it has been shown that distance-based approaches can be greatly improved with an adaptive distance measure (metric learning) [14]–[16]. In particular, we study both a) a more straightforward implementation of the metric learning idea by adapting metric in the space of vectorized (log-transformed) matrices and b) the full formulation of metric learning without matrix vectorization, thus preserving the second-order tensor structure.

The remainder of this article is organized as follows. Section II briefly presents LVQ. Section III describes the LEM. In Sections IV and V, we derive the proposed methods. Section VI gives experimental results. Section VII summarizes main findings and gives conclusions.

II. Learning Vector Quantization

In this section, we briefly introduce the LVQ methods involved in this article.

Given m training instances $\boldsymbol{x}_i \in \mathbb{R}^n$ labeled by $y_i \in \{1, \ldots, C\}$, $i = 1, \ldots, m$, where n denotes the input dimension and C represents the number of classes, a standard LVQ classifier consists of M ($M \geq C$) prototypes $\boldsymbol{w}_j \in \mathbb{R}^n$, labeled by $c_j \in \{1, \ldots, C\}$, $j = 1, 2, \ldots, M$. The classification rule takes the winner-takes-all scheme. The new instance $\boldsymbol{x} \in \mathbb{R}^n$ is assigned with the class of its closest prototype: that is, $\hat{y}(\boldsymbol{x}) := c_{j_*}$ such that $j_* = \arg\min_j d(\boldsymbol{x}, \boldsymbol{w}_j)$, where $d(\cdot, \cdot)$ is a distance measure in $\mathbb{R}^n$. The set of data points selecting the prototype $\boldsymbol{w}_j$ as their winner is defined as the receptive field of $\boldsymbol{w}_j$. The prototypes are learned automatically such that the data points in each prototype's receptive field with a different label are as few as possible.

GLVQ updates the prototypes through minimization of a well-delineated cost function by steepest gradient descent [3]. During the training process, a pair of prototypes is updated each time. Given an input $\boldsymbol{x}_i$ with label y_i, its closest correct prototype $\boldsymbol{w}_J$ (with the same label y_i) is propelled toward $\boldsymbol{x}_i$, while its closest incorrect prototype $\boldsymbol{w}_K$ (with a different label) is moved away from $\boldsymbol{x}_i$. The cost function of GLVQ is formulated as sum of normalized differences between the distances of instances to their correct prototypes and incorrect prototypes as follows:

$$E(\boldsymbol{w}) = \sum_{i=1}^{m} \Phi\left(\frac{d(\boldsymbol{x}_i, \boldsymbol{w}_J) - d(\boldsymbol{x}_i, \boldsymbol{w}_K)}{d(\boldsymbol{x}_i, \boldsymbol{w}_J) + d(\boldsymbol{x}_i, \boldsymbol{w}_K)}\right) \tag{1}$$

where $\Phi(\cdot)$ is a scaling function that increases monotonically, for example, the logistic function $\Phi(x) = 1/(1 + e^{-x})$ [3].

The above cost function of classic GLVQ takes the squared Euclidean distance, that is, $d(\boldsymbol{x}_i, \boldsymbol{w}_J) = (\boldsymbol{x}_i - \boldsymbol{w}_J)^T(\boldsymbol{x}_i - \boldsymbol{w}_J)$. Then, the learning rules are given as follows:

$$\Delta \boldsymbol{w}_J = \alpha \cdot 2\mu_K(\boldsymbol{x}_i - \boldsymbol{w}_J) \tag{2}$$

$$\Delta \boldsymbol{w}_K = -\alpha \cdot 2\mu_J(\boldsymbol{x}_i - \boldsymbol{w}_K) \tag{3}$$

where $\mu_K = ([2\Phi' d_K]/[(d_J + d_K)^2])$, $\mu_K = ([2\Phi' d_J]/[(d_J + d_K)^2])$, $d_J = d(\boldsymbol{x}_i, \boldsymbol{w}_J)$, $d_K = d(\boldsymbol{x}_i, \boldsymbol{w}_K)$, and $0 < \alpha < 1$ is the learning rate. According to the learning rules, the correct prototype $\boldsymbol{w}_J$ is dragged toward the instance $\boldsymbol{x}_i$ along the straight line connecting $\boldsymbol{w}_J$ with $\boldsymbol{x}_i$, while the incorrect prototype $\boldsymbol{w}_K$ is pushed away from the instance $\boldsymbol{x}_i$ along the straight line passing through $\boldsymbol{x}_i$ and $\boldsymbol{w}_K$.

GLVQ has been extended to incorporate metric learning. A diagonal matrix of relevance is considered in GRLVQ [1], while an adaptive matrix of relevance is used in GMLVQ [4]. Furthermore, local adaptive matrices are constructed to attach to individual prototypes, leading to LGMLVQ [4]. However, GLVQ and these extensions so far are only applicable to data points living in the Euclidean space.

The work presented in [17] extends GLVQ to directly deal with matrix data without any vectorization. The matrix data involved there are general matrices without any special properties and thus are still located in a Euclidean space. In contrast, we deal with SPD matrix data living on a curved Riemannian manifold. GLVQ has been extended to the manifold of SPD matrices equipped with AIRM, showing superior classification performance [13]. In the following sections, we extend GLVQ to data points living in an SPD Riemannian manifold with LEM.

III. Log-Euclidean Metric

Let $\mathbb{S}^+(n)$ represent the space of all real-valued $n \times n$ SPD matrices. It makes a Riemannian manifold if endowed with a Riemannian metric. LEM is a commonly used Riemannian metric on the SPD manifold $\mathbb{S}^+(n)$. It is derived by exploiting the Lie group structure under the group operation $\mathbf{S}_1 \odot \mathbf{S}_2 = \exp(\log \mathbf{S}_1 + \log \mathbf{S}_2)$, for $\mathbf{S}_1, \mathbf{S}_2 \in \mathbb{S}^+(n)$, where exp and log represent the matrix exponential function and the matrix logarithm, respectively.

[1] Representing the tangent space at the origin.

The well-studied LEM on Lie group of SPD matrices [6], [12] leads to the EM in the logarithm domain of SPD matrices. The geodesic between two points $\mathbf{X}_1, \mathbf{X}_2 \in \mathbb{S}^+(n)$ is the convex combination in the logarithmic domain

$$\gamma(t) = \exp(\log \mathbf{X}_1 + t(\log \mathbf{X}_2 - \log \mathbf{X}_1)), \ t \in [0, 1]. \quad (4)$$

By differentiating the above geodesic at $t = 0$, we can obtain the expression of the corresponding Riemannian logarithmic map, which gives the initial speed vector of the geodesic curve

$$\mathrm{Log}_{\mathbf{X}_1}(\mathbf{X}_2) = \mathbf{X}_1(\log \mathbf{X}_2 - \log \mathbf{X}_1). \quad (5)$$

With the initial speed vector $\mathbf{V} = \mathbf{X}_1(\log \mathbf{X}_2 - \log \mathbf{X}_1)$, the Riemannian exponential map returns to the point $\mathbf{X}_2$ on the manifold, that is, $\mathrm{Exp}_{\mathbf{X}_1}(\mathbf{V}) = \gamma(1) = \mathbf{X}_2$. Then, the exponential map induced by the LEM is given as follows:

$$\mathrm{Exp}_{\mathbf{X}_1}(\mathbf{V}) = \exp(\log \mathbf{X}_1 + \mathbf{X}_1^{-1}\mathbf{V}). \quad (6)$$

The LEM is defined as follows [6]:

$$\begin{aligned} \langle \mathbf{V}_1, \mathbf{V}_2 \rangle_{\mathbf{X}} &= \left\langle \mathbf{X}^{-1}\mathbf{V}_1, \mathbf{X}^{-1}\mathbf{V}_2 \right\rangle_{\mathbf{I}} \\ &= \mathrm{Tr}\left(\mathbf{X}^{-1}\mathbf{V}_1\mathbf{X}^{-1}\mathbf{V}_2\right) \end{aligned} \quad (7)$$

where Tr represents the trace operator. Under LEM, the squared geodesic distance between two SPD matrices is

$$\begin{aligned} \delta_{LE}(\mathbf{X}_1, \mathbf{X}_2) &= \langle \mathrm{Log}_{\mathbf{X}_1}(\mathbf{X}_2), \mathrm{Log}_{\mathbf{X}_1}(\mathbf{X}_2) \rangle_{\mathbf{X}_1} \\ &= \mathrm{Tr}\Big[(\log \mathbf{X}_2 - \log \mathbf{X}_1)^2\Big]. \end{aligned} \quad (8)$$

which corresponds to the Euclidean distance in the logarithmic domain.

Based on (4) and (6), the geodesic emitted at the point $\mathbf{X}$ in the direction of $\mathbf{V} \in \mathcal{T}_{\mathbf{X}}\mathbb{S}^+(n)$, that is, $\gamma_{LE}(0) = \mathbf{X}$, $\dot{\gamma}_{LE}(0) = \mathbf{V}$, can be expressed as

$$\gamma_{LE}(t) = \exp\Big(\log \mathbf{X} + t\mathbf{X}^{-1}\mathbf{V}\Big), \ t \in [0, 1]. \quad (9)$$

IV. GLVQ With LEM on the SPD Manifold

Although $S^+(n)$ is a convex subset of a vector space $\mathbb{R}^{n^2}$, it is *not* a vector space since the negation of a positive-definite matrix [a simple scalar multiplication on elements of $S^+(n)$] is not positive definite. Instead, $S^+(n)$ forms a manifold as it is locally homeomorphic to $R^{n(n+1)/2}$. The homeomorphism is provided by the vec operator that vectorizes the upper triangle of the SPD matrix. $\mathbb{S}^+(n)$ forms a *curved manifold* rather than a linear one as the boundary of the convex cone is curved.

Since $S^+(n)$ is a convex set, there is no obvious harm in evaluating distances in $S^+(n)$ with the usual Euclidean distance (e.g., when computing a set mean). However, the Euclidean distance cannot provide appropriate prototype updates. The prototype learning rule involves *minus* and so the application of the usual Euclidean distance may result in prototypes that are not positive definite. The work presented in [13] has already shown the inappropriateness of using EM for SPD matrices. Thus, we develop a new method, called GLVQ-LEM, by altering GLVQ to deal with SPD matrices under LEM.

Consider a training dataset $\{(\mathbf{X}_i, y_i)\}_{i=1}^m$, where $\mathbf{X}_i \in \mathbb{S}^+(n)$ represents the ith training instance and $y_i \in \{1, \ldots, C\}$ denotes the class of this instance. Assume the classifier is made of M ($M \geq C$) prototypes $\mathbf{W}_j \in \mathbb{S}^+(n)$ with labels $c_j \in \{1, \ldots, C\}$. As the data points are SPD matrices, the squared log-Euclidean distance is used to measure the difference between a prototype $\mathbf{W}_j$ and a training example $\mathbf{X}_i$, leading to the following cost function:

$$E(\mathbf{W}) = \sum_{i=1}^{m} \Phi(\mu(\mathbf{X}_i, \mathbf{W})) \quad (10)$$

$$\mu(\mathbf{X}_i, \mathbf{W}) = \frac{\delta_{LE}(\mathbf{X}_i, \mathbf{W}_J) - \delta_{LE}(\mathbf{X}_i, \mathbf{W}_K)}{\delta_{LE}(\mathbf{X}_i, \mathbf{W}_J) + \delta_{LE}(\mathbf{X}_i, \mathbf{W}_K)} \quad (11)$$

where $\mathbf{W}_J \in \mathbb{S}^+(n)$ represents the closest correct prototype to the training instance $\mathbf{X}_i \in \mathbb{S}^+(n)$, and $\mathbf{W}_K \in \mathbb{S}^+(n)$ denotes its closest incorrect prototype under the distance measure δ_{LE}.

To obtain the learning rules of prototypes, the above cost function is minimized through stochastic gradient descent on the manifold $\mathbb{S}^+(n)$. The Riemannian gradient of the cost function with respect to the prototype on the manifold $\mathbb{S}^+(n)$ can be computed by calculating the classical derivative of the composite function composed of the cost function and the geodesic emitting from the prototype [13]. The gradient of the cost function associated with a training instance $\mathbf{X}_i$ can be computed as follows. Denote $\gamma_J(t)$ as a geodesic curve that connects $\mathbf{X}_i$ with its closest correct prototype $\mathbf{W}_J$, starting from the prototype $\mathbf{W}_J$ with an initial speed vector $\mathbf{V}_J \in T_{\mathbf{W}_J}\mathbb{S}^+(n)$ [see (9)]

$$\gamma_J(t) = \exp\Big(\log \mathbf{W}_J + t{\mathbf{W}_J}^{-1}\mathbf{V}_J\Big). \quad (12)$$

Then, we can obtain the cost function with respect to the instance $\mathbf{X}_i$ along this curve as follows:

$$\begin{aligned} &E_i(\gamma_J(t), \mathbf{W}_K) \\ &\quad = \Phi\left(\frac{\delta_{LE}(\mathbf{X}_i, \gamma_J(t)) - \delta_{LE}(\mathbf{X}_i, \mathbf{W}_K)}{\delta_{LE}(\mathbf{X}_i, \gamma_J(t)) + \delta_{LE}(\mathbf{X}_i, \mathbf{W}_K)}\right). \end{aligned} \quad (13)$$

Let δ_J and δ_K denote the distances $\delta_{LE}(\mathbf{X}_i, \mathbf{W}_J)$ and $\delta_{LE}(\mathbf{X}_i, \mathbf{W}_K)$, respectively, and $g_K = 2\Phi'\delta_K/(\delta_J + \delta_K)^2$. The Riemannian gradient of E_i in the direction $\mathbf{V}_J$ evaluated at $\mathbf{W}_J$ denoted by $\nabla_{\mathbf{V}_J} E_i$ is given as follows:

$$\nabla_{\mathbf{V}_J} E_i = -2g_K\mathbf{W}_J(\log \mathbf{X}_i - \log \mathbf{W}_J). \quad (14)$$

The detailed derivation of $\nabla_{\mathbf{V}_J} E_i$ is given in Appendix A-A.

Let $\gamma_K(t)$ be a geodesic curve that connects $\mathbf{X}_i$ with its closest incorrect prototype $\mathbf{W}_K$, starting from the prototype $\mathbf{W}_K$ with an initial speed vector $\mathbf{V}_K \in T_{\mathbf{W}_K}\mathbb{S}^+(n)$. The Riemannian gradient of E_i in the direction $\mathbf{V}_K$ evaluated at $\mathbf{W}_K$, denoted by $\nabla_{\mathbf{V}_K} E_i$ can be obtained similarly

$$\nabla_{\mathbf{V}_K} E_i = 2g_J\mathbf{W}_K(\log \mathbf{X}_i - \log \mathbf{W}_K) \quad (15)$$

where $g_J = 2\Phi'\delta_J/(\delta_J + \delta_K)^2$.

Then, the prototypes $\mathbf{W}_J$ and $\mathbf{W}_K$ are updated using the exponential map as follows:

$$\mathbf{W}_J^{\mathrm{new}} = \mathrm{Exp}_{\mathbf{W}_J}\big(-\alpha\nabla_{\mathbf{V}_J} E_i\big) \quad (16)$$

$$\mathbf{W}_K^{\mathrm{new}} = \mathrm{Exp}_{\mathbf{W}_K}\big(-\alpha\nabla_{\mathbf{V}_K} E_i\big). \quad (17)$$

Equation (16) indicates that the correct prototype $\mathbf{W}_J$ is dragged toward the instance $\mathbf{X}_i$ along the geodesic curve emitted from $\mathbf{W}_J$ in the direction of $-\nabla_{\mathbf{V}_J} E_i$ ($\mathbf{V}_J$), which

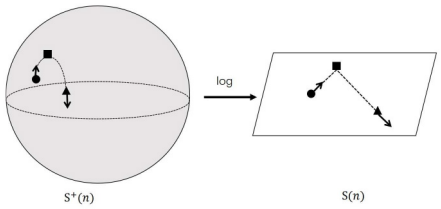


Fig. 1. Illustration of GLVQ on the SPD manifold endowed with LEM. The movements of prototype along the geodesic curve on the manifold corresponds to the movements along the straight line in the logarithm domain of the SPD matrices.

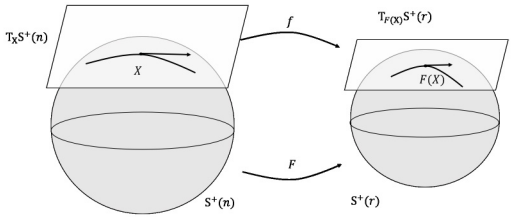


Fig. 2. Illustration of LEML [19]. LEML directly works on the logarithm of SPD matrices and learns a tangent map $f : \mathcal{T}_{\mathbf{X}}\mathbb{S}^+(n) \rightarrow \mathcal{T}_{\mathbf{X}}\mathbb{S}^+(r)$ from the original tangent space $\mathcal{T}_{\mathbf{X}}\mathbb{S}^+(n)$ to a possibly more discriminative tangent space $\mathcal{T}_{\mathbf{X}}\mathbb{S}^+(r)$. From this mapping, we can derive a manifold map $F : \mathbb{S}^+(n) \rightarrow \mathbb{S}^+(r)$. Thus, it implicitly maps the original manifold to a target one with more separability.

corresponds to the geodesic curve connecting $\mathbf{W}_J$ and $\mathbf{X}_i$. Analogously, (17) means that the incorrect prototype $\mathbf{W}_K$ is propelled away from the instance $\mathbf{X}_i$ in continuation of the geodesic curve connecting $\mathbf{X}_i$ with $\mathbf{W}_K$.

Taking the matrix logarithm on both left- and right-hand sides of (16) and (17), and denoting $\mathbf{T}_i = \log \mathbf{X}_i$, $\mathbf{U}_J = \log \mathbf{W}_J$, and $\mathbf{U}_K = \log \mathbf{W}_K$, the updating rule for prototypes can be rewritten in the logarithmic domain of SPD matrices

$$\mathbf{U}_J^{\text{new}} = \mathbf{U}_J + \alpha \cdot 2g_K(\mathbf{T}_i - \mathbf{U}_J) \tag{18}$$

$$\mathbf{U}_K^{\text{new}} = \mathbf{U}_K - \alpha \cdot 2g_J(\mathbf{T}_i - \mathbf{U}_K) \tag{19}$$

which exactly coincides with the update equations for the vector-valued data given by (2) and (3). The prototypes in the logarithm domain move toward or away from the instance in the logarithm domain along the straight line passing through the instance and the prototype.

As we mentioned before, LEM for SPD matrices corresponds to the EM in the logarithm domain of the SPD matrices. Therefore, under the LEM framework, the movement of the prototype along the geodesic curve on the manifold suggested in [13] corresponds to the movement of the prototype along the straight line in the logarithm domain of the SPD matrices, as shown in Fig. 1. Consequently, for the data of SPD matrices, we can transform it into the logarithmic domain and apply the standard GLVQ algorithm on the transformed data.

V. Metric Learning

In this section, we will enhance the discriminative power of our classifiers by further adapting the distance metric in the vector space of log-transformed SPD matrices. We will present two approaches. The first simplified approach adopts the usual ideas of metric learning in spaces of vectors as first-order tensors, obtained by vectorizing the log-transformed SPD matrices. However, this may distort the geometrical structure of the logarithm space derived from SPD matrices, as well as increase the computational complexity as it needs to learn a much larger distance metric than the data dimension. The second approach involves a full formulation of metric learning in the space of (log-transformed) SPD matrices, preserving their second-order tensor structure.

A. GMLVQ With LEM on the SPD Manifold

As we demonstrated in Section IV, the logarithm operation transforms the curved manifold into a flat space, where the classical Euclidean learning algorithm is applicable. Following this principle, we can also extend GMLVQ to the SPD manifold. The Mahalanobis distance function we need to learn is [18]

$$\delta_{LE}^{\Lambda}(\mathbf{X}, \mathbf{W}) = (\text{vec}(\log \mathbf{X}) - \text{vec}(\log \mathbf{W}))^T \cdot \Lambda \cdot (\text{vec}(\log \mathbf{X}) - \text{vec}(\log \mathbf{W})) \tag{20}$$

where $\mathbf{X} \in \mathbb{S}^+(n)$ represents a training instance, $\mathbf{W} \in \mathbb{S}^+(n)$ denotes a prototype, $\text{vec}(\cdot)$ denotes the operator that concatenates the upper triangular elements of the argument matrix into a vector, and $\Lambda \in \mathbb{S}^+(l)$ with $l = ([n(n+1)]/2)$ is the adaptive Mahalanobis matrix. The difference between the vectorized instance and prototype is weighted by the adaptive Mahalanobis matrix Λ, enabling separate weighting of individual dimensions, as well as accounting for pairwise interplay between the dimensions.

Following [18], we map the original SPD manifold $\mathbb{S}^+(n)$ to the vector space $\{\text{vec}(\log \mathbf{X}) | \mathbf{X} \in \mathbb{S}^+(n)\}$. Different from [18], which learns the Mahalanobis distance using information-theoretic metric learning in the vector space, we incorporate the distance learning in the LVQ system, more precisely, we apply standard GMLVQ in the vector space.

Similarly, we can restrict the global metric Λ to be a diagonal matrix and adapting GRLVQ for the classification of SPD matrices. We can also learn a local Λ_j for each prototype, extending LGMLQ for the classification of SPD matrices.

B. GLVQ With Log-Euclidean Metric Learning on the SPD Manifold

As shown in Fig. 2, we, following [19], learn a tangent map $f : \mathcal{T}_{\mathbf{X}}\mathbb{S}^+(n) \rightarrow \mathcal{T}_{\mathbf{X}}\mathbb{S}^+(r)$ projecting the original tangent space $\mathcal{T}_{\mathbf{X}}\mathbb{S}^+(n)$ to a possibly more discriminative tangent space $\mathcal{T}_{\mathbf{X}}\mathbb{S}^+(r)$ via $f(\log \mathbf{X}) = \Omega^T \log(\mathbf{X})\Omega$ with transformation $\Omega \in \mathbb{R}^{n \times r}$ of full rank in column, that is, $\text{rank}(\Omega) = r$, where $r \leq n$. With such Ω, the transformed point $\Omega^T \log(\mathbf{X})\Omega$ is a real symmetric matrix to form a valid tangent space. Then, the original manifold map $F : \mathbb{S}^+(n) \rightarrow \mathbb{S}^+(r)$ can be derived.

The geodesic distance between transformed data on the new resulting SPD manifold $\mathbb{S}^+(r)$ can be defined as follows:

$$\delta_{LE}^{\Omega}(\mathbf{X}, \mathbf{W}) = \left\| \Omega^T \log(\mathbf{X})\Omega - \Omega^T \log(\mathbf{W})\Omega \right\|_F^2 \tag{21}$$

which is the Frobenius norm of the difference matrix between the transformed instance and prototype in the logarithm domain.

The above distance function can be reparametrized as

$$\delta_{LE}^{\mathbf{Q}}(\mathbf{X}, \mathbf{W}) = \mathrm{Tr}\big[\mathbf{Q}(\log \mathbf{X} - \log \mathbf{W})(\log \mathbf{X} - \log \mathbf{W})\big] \quad (22)$$

where $\mathbf{Q} = (\Omega\Omega^T)^2$ is an $n \times n$ symmetric positive semidefinite matrix of rank r, taking a role of "reduced rank Mahalanobis matrix." The difference between the instance and prototype in the logarithm domain is weighed by the adaptive matrix $\mathbf{Q}$, casting different importances on each dimension of the difference matrix, as well as accounting for pairwise interplays between the dimensions. Note that when $r = n$, $\mathbf{Q}$ becomes an SPD matrix.

One crucial advantage of metric learning introduced in (22) over that in (20) is in that the dimensionality of $\mathbf{Q}$ is much smaller than that of Λ. Thus, it is more computationally efficient. Moreover, (22) always results in a valid tangent space, while (20) may not always and thus may distort the geometrical structure of the logarithm space derived from SPD matrices, resulting in degenerated performance.

With the distance function $\delta_{LE}^{\mathbf{Q}}$, we obtain the cost function

$$E(\mathbf{W}, \mathbf{Q}) = \sum_{i=1}^{m} \Phi(\mu(\mathbf{X}_i, \mathbf{W}, \mathbf{Q})) \quad (23)$$

$$\mu(\mathbf{X}_i, \mathbf{W}, \mathbf{Q}) = \frac{\delta_{LE}^{\mathbf{Q}}(\mathbf{X}_i, \mathbf{W}_J) - \delta_{LE}^{\mathbf{Q}}(\mathbf{X}_i, \mathbf{W}_K)}{\delta_{LE}^{\mathbf{Q}}(\mathbf{X}_i, \mathbf{W}_J) + \delta_{LE}^{\mathbf{Q}}(\mathbf{X}_i, \mathbf{W}_K)}. \quad (24)$$

Again, we can derive the learning rule of prototypes through minimization of the above cost function via stochastic deepest gradient descent on the manifold $\mathbb{S}^+(n)$.

The cost function of example $\mathbf{X}_i$ along the curve $\gamma_J(t)$ is

$$E_i^{\mathbf{Q}}(\gamma_J(t), \mathbf{W}_K, \mathbf{Q}) = \Phi\left(\frac{\delta_{LE}^{\mathbf{Q}}(\mathbf{X}_i, \gamma_J(t)) - \delta_{LE}^{\mathbf{Q}}(\mathbf{X}_i, \mathbf{W}_K)}{\delta_{LE}^{\mathbf{Q}}(\mathbf{X}_i, \gamma_J(t)) + \delta_{LE}^{\mathbf{Q}}(\mathbf{X}_i, \mathbf{W}_K)}\right). \quad (25)$$

Let $\delta_J^{\mathbf{Q}}$ and $\delta_K^{\mathbf{Q}}$ denote the distances $\delta_{LE}^{\mathbf{Q}}(\mathbf{X}_i, \mathbf{W}_J)$ and $\delta_{LE}^{\mathbf{Q}}(\mathbf{X}_i, \mathbf{W}_K)$, respectively. Let $f_K = 2\Phi'\delta_K^{\mathbf{Q}}/(\delta_J^{\mathbf{Q}} + \delta_K^{\mathbf{Q}})^2$ and $f_J = 2\Phi'\delta_J^{\mathbf{Q}}/(\delta_J^{\mathbf{Q}} + \delta_K^{\mathbf{Q}})^2$. The Riemannian gradient of $E_i^{\mathbf{Q}}$ in the direction $\mathbf{V}_J$ evaluated at point $\mathbf{W}_J$ denoted by $\nabla_{\mathbf{V}_J} E_i^{\mathbf{Q}}$ is given as follows:

$$\nabla_{\mathbf{V}_J} E_i^{\mathbf{Q}} = -2f_K \mathbf{W}_J \mathbf{Q}(\log \mathbf{X}_i - \log \mathbf{W}_J) \quad (26)$$

The detailed derivation of $\nabla_{\mathbf{V}_J} E_i^{\mathbf{Q}}$ is given in Appendix A-B.

Similarly, we can calculate the Riemannian gradient $\nabla_{\mathbf{V}_K} E_i^{\mathbf{Q}}$ utilizing the composite function composed of the cost function $E_i^{\mathbf{Q}}$ and the curve $\gamma_K(t)$, and obtain

$$\nabla_{\mathbf{W}_K} E_i^{\mathbf{Q}} = 2f_J \mathbf{W}_K \mathbf{Q}(\log \mathbf{X}_i - \log \mathbf{W}_K). \quad (27)$$

Then, we can have the learning rule for the prototypes in the logarithmic domain as follows:

$$\mathbf{U}_J^{\text{new}} = \mathbf{U}_J + \alpha \cdot 2f_K \mathbf{Q}(\mathbf{T}_i - \mathbf{U}_J) \quad (28)$$

$$\mathbf{U}_K^{\text{new}} = \mathbf{U}_K - \alpha \cdot 2f_J \mathbf{Q}(\mathbf{T}_i - \mathbf{U}_K) \quad (29)$$

The above prototype updating rules are different from those given by (18) and (19). The difference between the example and prototype in the logarithm domain is weighted by $\mathbf{Q}$, hopefully achieving better prototype position with more discriminative data-driven distance measure.

We then derive the learning rule for the distance matrix $\mathbf{Q}$. Two situations are considered and detailed here. We first restrict $\mathbf{Q}$ to be of full rank n, making it be an SPD matrix, where the LEM framework can be directly used. We then cast no such restriction on $\mathbf{Q}$, leaving it be a fixed-rank symmetric positive semidefinite matrix, where the quotient geometry with total space equipped with flat metric is utilized.

1) Learning $\mathbf{Q}$ *Using LEM:* Here, we consider $\mathbf{Q}$ to be an SPD matrix. The Riemannian gradient of the cost function with respect to $\mathbf{Q}$ can then be calculated by the same way as that with respect to the prototypes.

Let $\gamma_{\mathbf{Q}}(t)$ be a geodesic emitted from $\mathbf{Q}$ with initial speed $\mathbf{V}_{\mathbf{Q}} \in \mathcal{T}_{\mathbf{Q}}\mathbb{S}^+(n)$

$$\gamma_{\mathbf{Q}}(t) = \mathrm{Exp}(\log(\mathbf{Q}) + t\mathbf{Q}^{-1}\mathbf{V}_{\mathbf{Q}}). \quad (30)$$

The cost function of example $\mathbf{X}_i$ along the curve $\gamma_{\mathbf{Q}}(t)$ reads

$$E_i^{\mathbf{Q}}\big(\mathbf{W}_J, \mathbf{W}_K, \gamma_{\mathbf{Q}}(t)\big) = \Phi\left(\frac{\delta_{LE}^{\gamma_{\mathbf{Q}}}(\mathbf{X}_i, \mathbf{W}_J) - \delta_{LE}^{\gamma_{\mathbf{Q}}}(\mathbf{X}_i, \mathbf{W}_K)}{\delta_{LE}^{\gamma_{\mathbf{Q}}}(\mathbf{X}_i, \mathbf{W}_J) + \delta_{LE}^{\gamma_{\mathbf{Q}}}(\mathbf{X}_i, \mathbf{W}_K)}\right). \quad (31)$$

The Riemannian gradient of the cost function in the direction $\mathbf{V}_{\mathbf{Q}}$ evaluated at point $\mathbf{Q}$ denoted by $\nabla_{\mathbf{V}_{\mathbf{Q}}} E_i^{\mathbf{Q}}$ is given as

$$\nabla_{V_{\mathbf{Q}}} E_i^{\mathbf{Q}} = f_K \mathbf{Q}(\log \mathbf{X}_i - \log \mathbf{W}_J)^2 \mathbf{Q} - f_J \mathbf{Q}(\log \mathbf{X}_i - \log \mathbf{W}_K)^2 \mathbf{Q}. \quad (32)$$

The detailed derivation of $\nabla_{\mathbf{V}_Q} E_i^{\mathbf{Q}}$ is given in Appendix A-C.

Once we have the gradient, we can have the learning rule

$$\mathbf{Q}^{\text{new}} = \exp\Big\{\log \mathbf{Q} - \eta f_K(\mathbf{T}_i - \mathbf{U}_J)^2\mathbf{Q} + \eta f_J(\mathbf{T}_i - \mathbf{U}_K)^2\mathbf{Q}\Big\} \quad (33)$$

where $0 < \eta < 1$ is the learning rate. The metric tensor $\mathbf{Q}$ is updated by moving along the geodesic curve emitted at $\mathbf{Q}$ in the direction of $-f_K(\mathbf{T}_i - \mathbf{U}_J)^2\mathbf{Q} + f_J(\mathbf{T}_i - \mathbf{U}_K)^2\mathbf{Q}$. In this way, the gradient of the distances from both the closest correct prototype and the closest incorrect prototype provides updating force for $\mathbf{Q}$. The gradient of the distance from the closest correct prototype contributes a negative (scale with -1) driving force. Thus, the elements of $\mathbf{Q}$ are modified so that the distance from the closet correct prototype becomes smaller. The gradient of the distance from the closest incorrect prototype contributes a positive driving force. Thus, it forces elements of $\mathbf{Q}$ to change so that the distance from the closest incorrect prototype is increased.

2) Learning $\mathbf{Q}$ *Using Quotient Geometry With Flat Metric:* In the above section, the distance matrix $\mathbf{Q}$ is confined to be of full rank n, such that we can readily use the LEM framework to learn $\mathbf{Q}$. However, in some applications, the dimension of the input SPD matrices can be very high and some dimensions of the input may barely contribute to the classification task. For instance, in the BCI research, the number of electrodes used to record the brain signals can be very high [20], producing SPD matrices of very high dimensions. However, not all the electrodes contribute greatly for the identification

of brain states. In this case, we can restrict the rank of $\mathbf{Q}$ to be smaller than n, speeding up the computation and possibly improving the generalization performance. In this section, we consider $\mathbf{Q}$ of rank r, where $r \leq n$. $\mathbf{Q}$ then becomes a point in the space of $\mathbb{S}^+(r, n)$ containing all $n \times n$ fixed-rank positive semidefinite matrices of rank r.

The space of $\mathbb{S}^+(r, n)$ can be identified as a quotient manifold [21]. The quotient geometry comes from the matrix factorization $\mathbf{Q} = \mathbf{G}\mathbf{G}^T$ with $\mathbf{G} \in \mathbb{R}_*^{n\times r}$, where $\mathbb{R}_*^{n\times r}$ represents all invertible $n \times r$ matrices. Matrix factorization is invariant under rotation. In other words, multiplying $\mathbf{G}$ by a rotation matrix $\mathbf{O}$ produces the same $\mathbf{Q}$. Let $\mathbb{O}(r)$ represent the space containing all rotation matrices (linear operators preserving the norm). Then, we can have a set of equivalence classes $[\mathbf{G}] = \{\mathbf{GO},\ \text{s.t. } \mathbf{G} \in \mathbb{O}(r)\}$ which can be identified by the quotient $\mathbb{S}^+(r, n) \simeq \mathbb{R}_*^{n\times r}/\mathbb{O}(r)$.

The tangent space $T_{[\mathbf{G}]}\mathbb{S}^+(r, n)$ at $[\mathbf{G}]$ contains all directions that induces no displacement in the equivalence class $[\mathbf{G}]$. A tangent vector $\xi_{[\mathbf{G}]} \in T_{[\mathbf{G}]}\mathbb{S}^+(r, n)$ corresponds to a unique horizontal vector $\bar{\xi}_{\mathbf{G}} \in T_{\mathbf{G}}\mathbb{R}_*^{n\times r}$ that is orthogonal to the equivalence class $[\mathbf{G}]$. The EM bestowed on the total space $\mathbb{R}_*^{n\times r}$ induces a Riemannian metric in the quotient space [21]

$$g_{[\mathbf{G}]}\big(\xi_{[\mathbf{G}]}, \zeta_{[\mathbf{G}]}\big) \triangleq \bar{g}_{\mathbf{G}}\big(\bar{\xi}_{\mathbf{G}}, \bar{\zeta}_{\mathbf{G}}\big) = \mathrm{Tr}\Big(\bar{\xi}_{\mathbf{G}}^T\ \bar{\zeta}_{\mathbf{G}}\Big). \tag{34}$$

Here, the inner product of two tangent vectors in the quotient space is the standard inner product of their corresponding horizontal vectors. With this geometry, the exponential mapping is

$$\mathrm{Exp}_{\mathbf{G}}\ \big(\bar{\xi}_{\mathbf{G}}\big) = \big[\mathbf{G} + \bar{\xi}_{\mathbf{G}}\big] = \mathbf{G} + \bar{\xi}_{\mathbf{G}} \tag{35}$$

leading to a straight geodesic expressed as follows:

$$\gamma_{\mathbf{G}}(t) = \mathrm{Exp}_{\mathbf{G}}\ \big(t\bar{\xi}_{\mathbf{G}}\big) = \mathbf{G} + t\bar{\xi}_{\mathbf{G}}. \tag{36}$$

The horizontal gradient of a function g defined in the quotient space is the unique vector $\overline{\nabla g(\mathbf{G})}$ satisfying

$$\bar{g}_{\mathbf{G}}\Big(\bar{\xi}_{\mathbf{G}}, \overline{\nabla g(\mathbf{G})}\Big) = \frac{\mathrm{d}}{\mathrm{d}t} g(\gamma_{\mathbf{G}}(t))|_{t=0}. \tag{37}$$

The calculation of the horizontal gradient can then be performed through computation of the classical derivative of the composite function $g \circ \gamma$.

Since $\mathbf{Q}$ is decomposed as the matrix factorization of $\mathbf{G}$, the geodesic distance $\delta_{LE}^{\mathbf{Q}}$ parametrized by $\mathbf{Q}$ can be reparameterized by $\mathbf{G}$ as

$$\delta_{LE}^{\mathbf{Q}}(\mathbf{X}, \mathbf{W}) = \delta_{LE}^{\mathbf{G}\mathbf{G}^T}(\mathbf{X}, \mathbf{W}) = \mathrm{Tr}\big[\mathbf{G}\mathbf{G}^T(\mathbf{T} - \mathbf{U})(\mathbf{T} - \mathbf{U})\big]$$

where $\mathbf{T} = \log \mathbf{X}$ and $\mathbf{U} = \log \mathbf{W}$. Using the formula of the above geodesics, the distance along the geodesic

$$\delta_{LE}^{\gamma_{\mathbf{G}}} = \delta_{LE}^{\gamma_{\mathbf{G}}\gamma_{\mathbf{G}}^T} = \mathrm{Tr}\big[\gamma_{\mathbf{G}}(t)(\gamma_{\mathbf{G}}(t))^T(\mathbf{T} - \mathbf{U})(\mathbf{T} - \mathbf{U})\big]. \tag{38}$$

The horizontal gradient of the cost function $E_i^{\mathbf{Q}}$ at $\mathbf{G}$ is given as follows:

$$\overline{\nabla_{\bar{\xi}_{\mathbf{G}}} E_i^{\mathbf{Q}}} = 2f_K(\mathbf{T}_i - \mathbf{U}_J)^2\mathbf{G} - 2f_J(\mathbf{T}_i - \mathbf{U}_K)^2\mathbf{G}. \tag{39}$$

The detailed derivation of $\overline{\nabla_{\bar{\xi}_{\mathbf{G}}} E_i^{\mathbf{Q}}}$ is given in Appendix A-D.

Algorithm 1 GLVQ With LEML for SPD Matrices

Input: m training instances $(\mathbf{X}_1, y_1), \ldots, (\mathbf{X}_m, y_m)$, where $\mathbf{X}_i \in \mathbb{S}^+(n)$ and $y_i \in \{1, \ldots, C\}$

Output: M labeled prototypes $(\mathbf{U}_1, c_1), \ldots, (\mathbf{U}_M, c_M)$, where $\mathbf{U}_i \in \mathbb{S}(n)$, and $c_i \in \{1, \ldots, C\}$, and metric $\mathbf{Q} \in \mathbb{S}^+(n)$ (GLVQ-LEML-LEM) or $\mathbf{Q} \in \mathbb{S}^+(r, n)$ (GLVQ-LEML-FM)

for i=1 to m **do**
  $\mathbf{T}_i = \log \mathbf{X}_i$
end for
Randomly initialize $\mathbf{U}_j$, and initialize $\mathbf{Q}$ as identity matrix
while a stopping criterion is not reached **do**
  Choose an instance $(\mathbf{T}_i, y_i)$ randomly
  Identify the closest correct $\mathbf{U}_J$ and incorrect prototypes $\mathbf{U}_K$ respectively, utilizing the distance measure $\delta_{LE}^{\mathbf{Q}} = \mathrm{Tr}[\mathbf{Q}(\mathbf{T}_i - \mathbf{U})(\mathbf{T}_i - \mathbf{U})]$
  Update prototypes $\mathbf{U}_J$ and $\mathbf{U}_K$ according to Eqs. (28) and (29), respectively
  Update distance matrix $\mathbf{Q}$ according to Eq. (33), leading to the implementation of the GLVQ-LEML-LEM method, or according to Eqs. (40) and (41), resulting in the realization of the GLVQ-LEML-FM method
end while

Consequently, the updating rule of $\mathbf{Q}$ is given as follows:

$$\mathbf{G} \leftarrow \mathbf{G} - 2\eta\Big(f_K(\mathbf{T}_i - \mathbf{U}_J)^2 - f_J(\mathbf{T}_i - \mathbf{U}_K)^2\Big)\mathbf{G} \tag{40}$$

$$\mathbf{Q} \leftarrow \mathbf{G}\mathbf{G}^T. \tag{41}$$

Similarly, the matrix $\mathbf{Q}$ is updated along geodesic due to the matrix factorization. The factor $\mathbf{G}$ is updated along a straight line. It changes so that the distance from the closest correct prototype becomes smaller, while the distance from the closest incorrect prototype increases. Note that this updating rule for $\mathbf{Q}$ is quite similar to the way the metric is learned in GMLVQ [4].

The learning procedure of GLVQ-LEML (including both GLVQ-LEML-LEM and GLVQ-LEML-FM) is summarized in Algorithm 1.

VI. Experiments

The proposed methods were verified on multiple datasets of different natures, including two synthetic datasets, three image datasets, and one motor imagery dataset. In this article, the scaling function Φ was set as $\Phi(x) = 1/(1 + e^{-x})$. Continuously decreasing learning rates in the learning process were used. The learning rate of prototypes followed the decreasing schedule $\alpha(t) = (n\xi/100)0.01^{t/T}$, where ξ represents the number of prototypes per class, n denotes the rank of the input matrix, T represents the number of training epochs, and $t = 1, \ldots, T$. The learning rate of metric $\mathbf{Q}$ followed the schedule: $\eta(t) = (n/1000)0.01^{t-t_0/T-t_0}$, where t_0 represents the time when to start learn $\mathbf{Q}$. To make the learning algorithm stable, we used smaller learning rates for $\mathbf{Q}$ compared to that of prototypes and started to learn $\mathbf{Q}$ when the learning of prototypes was stable. In this article, we set $t_0 = 1$.

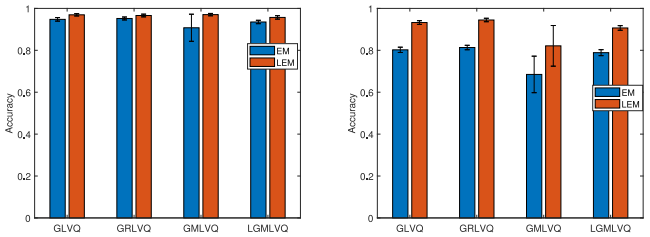


Fig. 3. Accuracy comparison between different metrics in the framework of LVQs on synthetic datasets. (a) SynI. (b) SynII.

A. Synthetic Data

To directly compare our proposed methods with the method presented in [13], two synthetic datasets created in [13] were used, namely: 1) SynI and 2) SynII. Both synthetic datasets contain four categories consisting SPD matrices of size 10×10. For SynI, class 1 and class 2 shared a set of eigenvalues, class 3 and class 4 shared another set of eigenvalues, class 1 and class 3 shared a set of eigenvectors, while class 2 and class 4 shared another set of eigenvectors. For SynII, all classes shared one set of eigenvectors but each class had its own set of eigenvalues. Both of the two datasets contain three subsets, for training, validation, and test, respectively. Each subset contains 250 data points per class. The generating process was repeated for 30 times. Averaged results over the 30 runs were reported. The detailed generating process can be found in [13].

All algorithms involved in this section were realized in MATLAB and run using a single core Intel 3.4-GHz CPU under a 64-bit Linux system. They were all trained for 100 epochs. The number of prototypes per class was chosen from 1 to 5 based on the performance on the validation dataset, if not specified.

We first compared LVQs under LEM with those under EM for classifying SPD matrices. As mentioned in [13], under EM, the learned prototypes cannot always stay positive definite. In this case, whenever a prototype update leaves $S^+(n)$, we project it back onto $S^+(n)$ by setting the nonpositive eigenvalues to a tiny positive value. The results are depicted in Fig. 3, showing that LEM significantly improves the classification performance.

We then compared our proposed methods developed under LEM with GLVQ-AIRM [13] in terms of both performance and computational time. Table I shows the mean accuracy of the studied Riemann GLVQ methods across 30 runs together with standard derivation ($\pm$).

Table I suggests that the proposed methods under LEM obtained comparable performance to GLVQ-AIRM on the synthetic datasets.

As shown in Table I, GLVQ-LEML-LEM obtained slightly better classification accuracy than GLVQ-LEML-FM. To better understand the difference between the two approaches, we compared their learned metrics. Fig. 4 shows the eigenvalues of the two metrics, respectively. The eigenvalues of the metric learned in GLVQ-LEML-FM spread more broadly than that in GLVQ-LEML-LEM. This suggests that GLVQ-LEML-FM is more likely to drive the weights of some

TABLE I
MEAN TEST ACCURACY OF THE RIEMANN GLVQ METHODS ACROSS 30 RUNS TOGETHER WITH STANDARD DERIVATION ($\pm$). THE PERFORMANCE BETTER THAN GLVQ-AIRM IS MARKED IN BOLDFACE

Method	SynI	SynII
GLVQ-AIRM	97.53 $\pm$ 0.58	92.13 $\pm$ 0.72
GLVQ-LEM	96.94 $\pm$ 0.54	**93.26 $\pm$ 0.89**$_\bullet$
GRLVQ-LEM	96.62 $\pm$ 0.64	**94.47 $\pm$ 0.82**$_\bullet$
GMLVQ-LEM	97.04 $\pm$ 0.56	82.13 $\pm$ 9.70
LGMLVQ-LEM	95.70 $\pm$ 0.82	90.67 $\pm$ 1.10
GLVQ-LEML-LEM	**97.59 $\pm$ 0.58**	**92.76 $\pm$ 0.78**$_\bullet$
GLVQ-LEML-FM	97.08 $\pm$ 0.60	**92.39 $\pm$ 0.94**$_\circ$

$\bullet$: Statistically significantly better than GLVQ-AIRM with a level of significance of $\alpha = 0.05$.
$\circ$: Statistically significantly better than GLVQ-AIRM with a level of significance of $\alpha = 0.1$.

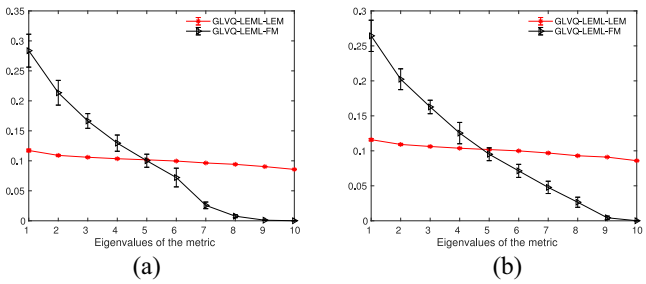


Fig. 4. Eigenvalues of the learned metric on synthetic datasets, with five prototypes per class. (a) SynI dataset. (b) SynII dataset.

dimensions toward 0 and it may obtain better performance if the dimensions of the data are large and some dimensions of the data contribute little to the classification task. However, GLVQ-LEML-LEM restricts the metric to be positive definite, confining the weight of each dimension within a small range. Therefore, GLVQ-LEML-LEM obtained better performance when the dimensionality of the data is small and all dimensions contribute greatly to the classification task.

Fig. 5 presents the classification accuracy, training time, and test time of the methods as a function of the number of prototypes per class. As we can see from Fig. 5(a) and (b), the methods appear to be remarkably robust to the danger of overfitting caused by the increased number of class prototypes, with the exception of GMLVQ With LEM (GMLVQ-LEM).

Fig. 5(c) and (d) shows that the proposed methods were computationally efficient. Both training and test time of all the algorithms linearly depend on the number of prototypes per class. The training of GLVQ-LEML (using either LEM or FM) and GMLVQ-LEM were both slower than that of GLVQ-LEM, as they need to learn the metric besides the prototypes. As we expected, the training of GLVQ with LEM learning (GLVQ-LEML) is faster than GMLVQ-LEM, since the metric in GLVQ-LEML is of size $n \times n$, while the metric in GMLVQ-LEM is of size $([n(n+1)]/2) \times ([n(n+1)]/2)$. However, the test of GLVQ-LEML is slightly slower than the other approaches under LEM, as the computation of the geodesic distance δ_{LE}^{Q} is more complicated than δ_{LE}. As we expected, the computational time of our methods under LEM is much faster than that of GLVQ-AIRM, except LGMLVQ-LEM which leans one local metric for each prototype.

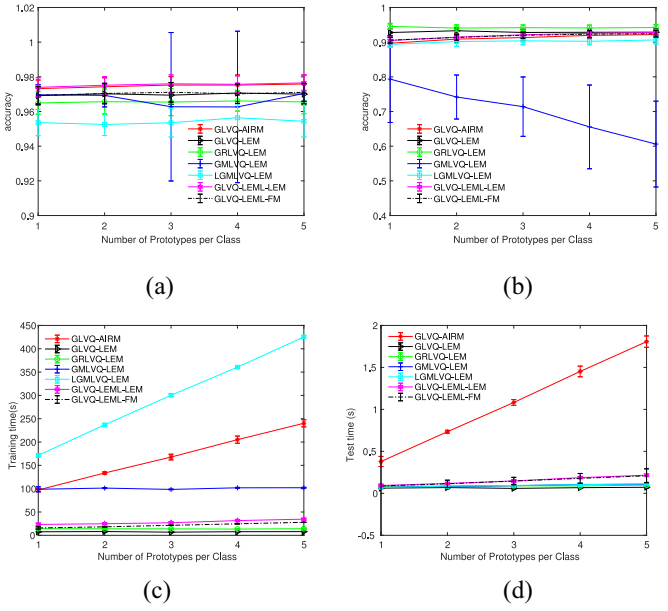


Fig. 5. Performances as a function of the number of prototypes per class. (a) Accuracy for SynI. (b) Accuracy for SynII. (c) Training time for SynI. (d) Test time for SynI.

Experiments on two synthetic datasets indicate the proposed GLVQ methods under LEM can achieve comparable performance to that under AIRM. Moreover, the proposed methods under LEM are much faster than GLVQ-AIRM.

B. Image Classification

The ETH-80 dataset [22] was used to evaluate our methods. It contains eight categories. Each category consists of ten objects. Each object has 41 images.

Following [9], we randomly chose 21 images of each object to train the classifier and used the remaining images to verify the classification performance. A single 5×5 covariance descriptor was used to represent each image. The features $[x, y, I, |I_x|, |I_y|]$ were used to calculate the covariance descriptor, where x and y represent the locations of the pixel and I, I_x, and I_y denote the corresponding intensity and derivatives. The splitting of the training and test set was randomly and independently repeated for 20 times, and thus the training and test procedure was repeated for 20 runs. The averaged performance over 20 runs is reported.

We compared our proposed methods with other LEM-based methods in the literature, particularly, K-means with LEM (KM-LEM) [9], kernel KM-LEM (KKM-LEM) [9], minimum distance to Riemannian mean with LEM (MDRM-LEM), and LEM learning (LEML) [19] on the ETH-80 dataset. The training epochs and the number of prototypes per class were selected by five-fold cross-validation on the training set. For training epochs, we experimented with 20, 50, 100, 200, and 500 iterates, respectively, while for the number of prototypes per class, we enumerated from 1 prototype per class to 10 prototypes per class. For LEML, the parameter η is chosen from $[0.01, 0.1, 1, 10, 100]$ and ζ is tuned from 0.1 to 0.5 by five-fold cross-validation on the training set.

TABLE II
COMPARISON OF OUR PROPOSED METHODS WITH OTHER LEM-BASED METHODS IN THE LITERATURE. THE RESULTS OF KM-LE, KKM-LE, AND MDRM-LE ARE TAKEN FROM [9]. THE BEST PERFORMANCE IS MARKED IN BOLDFACE

Methods	Accuracy
KM-LEM	58.31
KKM-LEM	71.44
MDRM-LEM	63.16
GLVQ-LEM	$83.07 \pm 1.11_{\bullet}$
GRLVQ-LEM	$83.84 \pm 0.93_{\bullet}$
GMLVQ-LEM	$82.08 \pm 1.77_{\bullet}$
LGMLVQ-LEM	$81.19 \pm 1.10_{\bullet}$
GLVQ-AIRM	$84.56 \pm 0.70_{\circ}$
LEML	$82.73 \pm 1.08_{\bullet}$
GLVQ-LEML-LEM	$\mathbf{85.27 \pm 1.41}$
GLVQ-LEML-FM	$84.01 \pm 3.03_{\circ}$

●: Statistically significant worse than GLVQ-LEML-LEM with a level of significance of $\alpha = 0.05$.
○: Statistically significant worse than GLVQ-LEML-LEM with a level of significance of $\alpha = 0.1$.

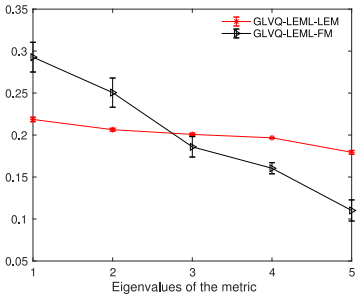


Fig. 6. Eigenvalues of the learned metrics on ETH-80. The analyzed number of classes was 8. Five prototypes were learned for each class. The method were trained epochs was 100.

Experimental results are given in Table II. The results show that both GLVQ-LEML-LEM and GLVQ-LEML-FM obtained significantly better performance than GLVQ-LEM, GRLVQ-LEM, GMLVQ-LEM, and LGMLVQ-LEM, while obtained slightly better performance than GLVQ-AIRM and LEML. These methods are much better than KM-LEM, KKM-LEM, and MDRM-LEM.

Table II also shows that the performance of GLVQ-LEML-LEM is slightly better than that of GLVQ-LEML-FM. We again compared the metrics learned by the two different approaches. As we can see in Fig. 6, the eigenvalues of both metrics are far from 0, indicating all dimensions of the data contribute greatly to the task. This confirms the findings in the experiments on the synthetic datasets, that GLVQ-LEML-LEM will obtain better performance if all dimensions of the data contribute greatly to the task. Again, the eigenvalues of the metric learned by GLVQ-LEML-FM spread slightly broader than those learned by GLVQ-LEML-LEM.

It can be argued that when the size of available training data is large, specialized Riemannian manifold-based methods of the kind presented here may not be needed. Indeed, for example, deep convolutional neural networks (CNNs) and related approaches can implicitly learn to handle the underlying structure of the input space. However, there are still situations where only limited sample sizes are available, for instance, in EEG data classification [11]. In such situations,

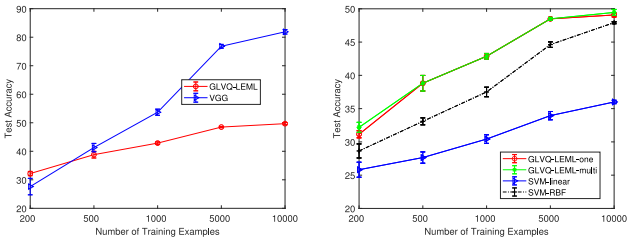
it is necessary to consider carefully the structure of the input space. To support this argument, we performed a set of controlled experiments on a downsampled large dataset, with a gradually increasing sample size. The dataset, CIFAR-10 [23], contains 50 000 training and 10 000 test images organized in ten classes. We randomly drew 200, 500, 1000, 5000, and 10000 images from the training set (in a stratified manner) to train our methods, as well as the VGG net [23], a representative deep CNN method. The methods were then verified on the hold-out set of 10 000 test images. The random downsampling of training images was repeated five times. Average performance together with standard derivation across the five runs was reported.

Each VGG network was trained for 100 epochs. The learning rate was annealed according to the exponential decay schedule, that is, $\alpha(t) = 0.01 * 0.99^t$. The architecture was chosen from 11, 13, 16, and 19 layers (detailed configuration is given by A in [23, Table 1]) via five-fold cross-validation only using the training split. As for our methods, we used GLVQ-LEML-LEM, since GLVQ-LEML-LEM obtained better performance than others on the image dataset. We trained for 100 training epochs. The number of prototypes per class was picked from 1 to 5 also by five-fold cross-validation on the training set. Following [18], each image is represented by 9×9 covariance descriptor calculated from the features $[x, y, R, G, B, |I_x|, |I_y|, |I_{xx}|, |I_{yy}|]$, where x and y are pixel locations and I, I_x, and I_y are corresponding intensity and derivatives, I_{xx} and I_{yy} are corresponding second-order partial derivative.[2]

The mean test accuracy curves as functions of the number of training examples, are presented in Fig. 7(a). As expected, when the training sample size is small, our dedicated method performs better than deep CNNs. However, as the size of the training sample increases, the accuracy of the VGG net increases sharply. We observed that the training accuracy of VGG net on the small training sample (200 training instances) is also very small (46.5%). Therefore, the inferior performance is not due to overfitting. Note that CIFAR-10 contains ten classes. Our proposed method trained with 20 examples per class obtained an accuracy of 32.2%, three times as many as the random guess (10%).

To further demonstrate the effectiveness of our proposed methods in the case of small training samples, we compared GLVQ-LEML-LEM with SVM. For SVM, both linear and spherical Gaussian kernels were used. The penalty parameter C and kernel width γ were both selected from {0.01, 0.1, 1, 10, 100,1000}, respectively, by five-fold cross-validation on the training set. The inputs were normalized to zero mean and unit variance per dimension. To provide a fair comparison, the number of prototypes per class was either fixed to one (in the Euclidean case, this setting corresponds to a linear classification boundary between two classes), or was allowed to be determined by cross-validation from $\{1, 2, 3, 4, 5\}$. We refer to the former and latter cases as GLVQ-LEML-one and GLVQ-LEML-multi, respectively. The models were trained

[2]We have tried using 5×5 covariance descriptor. The performance of our method gives better performance when using 9×9 covariance descriptor.



(a) (b)

Fig. 7. Performance curves as functions of the number of training examples on the CIFAR-10 dataset. GLVQ-LEML-one and GLVQ-LEML-multi refer to our model with one and cross-validated number of prototypes per class, respectively. SVM with linear and spherical Gaussian kernels are referred to as SVM-linear and SVM-RBF, respectively. (a) Comparison with CNN. (b) Comparison with SVM.

for 100 epochs. Fig. 7(b) shows that our GLVQ-LEML-LEM model consistently outperformed SVM (linear or nonlinear), confirming the effectiveness of our proposed methods.

The VGG net takes full raw images as inputs, while our method takes highly summarized covariance structures of images as inputs. To provide a thorough comparison, we further compared our method with a deep network called SPD matrix network (SPDNet) [24] that also takes covariance descriptors as inputs. Following [24], the popular Acted Facial Expression in Wild (AFEW) [25] was used. The AFEW dataset consists of 1345 videos of seven facial expressions performed by actors in movies in real-world scenarios. The videos were divided into three subsets for training, validation, and test, respectively. Ground-truth class labels of the test set were unavailable. We therefore following [24] reported the results on the hold-out validation set. We preprocessed the data the same as in [24]. Training videos were segmented into 1747 small clips. Each clip was represented by a covariance matrix of size 400×400.

Since the SPD matrix is high dimensional, we used our GLVQ-LEML-FM to perform the comparison experiments. As for the SPDNet, the best architecture (SPDnet-3BiRe) reported in [24] was used. The training epochs of both methods were set as 100. The number of prototypes of our method was chosen from 1 to 3 via five-fold cross-validation on the training set.

Analogously to our previous experiment, we gradually increased the training sample size starting from 350, through 525, 700, 875, to 1050 randomly subsampled instances from the training set. Each random downsampling process was repeated for five times. We reported the mean accuracy on the hold-out validation set over five runs together with standard deviations. The hold-out validation set results are given in Fig. 8. Note that, in Fig. 8, the accuracy of the SPDNet that was trained with the entire 1747 training instances was taken from [24], thus no standard derivation was provided.

Fig. 8 shows that the performance discrepancy between our method and the SPDnet is rather small. Our method provided slightly better performance when the training sample is small. The overall performance of our method was comparable to that of the SPDNet method. Note that the AFEW dataset contains seven classes. The proposed method trained with 50 examples per class, obtained an accuracy of 28.4%, twice as many as

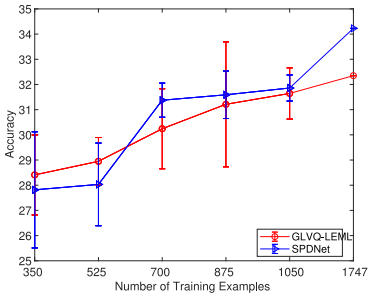


Fig. 8. Performance changes as a function of the number of training examples on the AFEW dataset.

the random guess (1/7). Given that the data is complex and high dimensional, the performance of our proposed method is acceptable. More importantly, the SPDNet did not provide better performance than that of our method either.

C. Motor Imagery Datasets

We finally evaluated our proposed approaches on the dataset 2a in BCI competition IV (IV-2a) [26]. This dataset consists of EEG signals recorded from nine healthy subjects using 22 electrodes at a sampling rate of 250 Hz. During recording, the subject was cued to execute one of four motor imagery tasks for 4 s, including imagining left-hand movement, right-hand movement, both feet movement, and tongue movement. For each subject, two sessions were recorded on different days. Each session contains 288 trials with 72 trials per class. In each trial, the segment of 2 s starting from 0.5 s after the presence of the cue was extracted for analysis. The extracted segments were bandpass filtered by a fifth-order Butterworth filter in the 10–30-Hz frequency band and then transformed into spatial covariance matrices of size 22×22 following the same procedure as in [13] and [27].

The performance of our proposed methods was compared with three winning results in terms of kappa value distributed at the website of BCI competition IV and other existing results presented in the literature. Kappa is a widely used performance metric in motor imagery classification tasks [28]. It is calculated as $\mathcal{K} = (P_a - P_c)/(1 - P_c)$, where P_a is the classification accuracy and P_c is the classification accuracy of random guess. Since the data contains four balanced classes, we have $P_c = 1/4$.

For our presented methods, the number of prototypes per class and training epochs were chosen from $\{1, 2, 3\}$ and $\{20, 50, 100\}$, respectively, by five-fold cross-validation on the training fold.

Table III compares our proposed methodologies with the state-of-the-art methods. The methods labeled by 1st, 2nd, and 3rd represent the three winning results on the IV-2a dataset distributed on the website of BCI competition IV [29]. Minimal distance to Riemannian mean (MDRM) [30] learns a cluster center for each class of instances by computing their Riemannian geometric mean under the AIRM. The class label of an unknown instance is obtained by finding its AIRM-closest center. Apart from using a different Riemannian metric, our proposed methods can learn multiple representatives for each class. Thus, our methods are more flexible and potentially more powerful than MDRM. Tangent space linear discriminant analysis (TSLDA) [30] is an extension of Euclidean LDA to the Riemannian manifold of SPD matrices equipped with the AIRM. TSLDA projects SPD matrices from the training set onto the tangent space at their Riemannian mean and then applies the standard Euclidean linear discriminant analysis in the tangent space. This treatment of mapping data to a tangent space at a particular point gives a first-order approximation of the data manifold, which can introduce significant distortions, especially in regions that are far from the origin of the tangent space. Our proposed methods do not utilize this global tangent space projection. Instead, our methods utilize the Riemannian stochastic gradient descent algorithm, which only involves local tangent space projections and thus can avoid large-scale distortions. Moreover, TSLDA inherits the binary nature of LDA. To deal with multiclass classification of SPD matrices, TSLDA needs to decouple the problem into multiple binary classification problems using appropriate heuristics (e.g., one-versus-one). Instead, our methods can naturally deal with multiclass classification problems. Tangent space of submanifold learning followed by linear discriminant analysis (TSSM+LDA) [31] is an extension of TSLDA with submanifold learning. This method first learns an optimal map from the original Riemannian space of SPD matrices to the Riemannian submanifold through jointly diagonalizing the Riemannian geometric means of the two class data. The optimal map transforms the original SPD matrices into lower dimensional SPD matrices where tangent space projection is performed and then Euclidean LDA is applied. Wavelet-spatial convolution network (WaSF ConvNet) [28] is a deep learning approach that learns joint space–time–frequency features using Morlet wavelet-like kernels and spatial kernels. To enable the deep nets work for small training data, cropped training, early stopping, and subject-to-subject weight transfer were used.

Table III shows that our proposed methods obtained comparable performance to the state-of-the-art methods. Interestingly, the proposed GLVQ-LEML-FM method obtained the best performance among our proposed approaches, significantly beating the proposed GLVQ-LEML-LEM method. We observed that many eigenvalues of the learned metric using GLVQ-LEML-FM were close to 0 (see Fig. 9), indicating that only a few dimensions of the data contributed greatly to the classification task. This confirms the finding that not all the recorded electrodes provided significantly useful information [32]. However, GLVQ-LEML-LEM restricted the distance metric to be positive definite, violating the nature of the data. Consequently, it obtained inferior performance to that of the GLVQ-LEML-FM method.

Compared to the three winning results distributed on the website of BCI competition IV, our methods obtained superior performance. The performance of our proposed GLVQ-LEML-FM method was significantly better than the 1st winner of BCI competition IV. The GLVQ-LEML-FM method outperformed the 1st winner on five subjects over the nine subjects. The GLVQ-LEML-LEM method obtained slightly worse performance than the 1st winner, but obtained marginally better performance than the 2nd winner. Our other proposed methods are all marginally better than the 3rd winner.

TABLE III
PERFORMANCE COMPARISON BETWEEN OUR METHODS AND EXISTING RESULTS IN TERMS OF KAPPA VALUE ON THE IV-2A DATASET. METHODS ARE ARRANGED IN A DESCENDING ORDER ACCORDING TO THEIR PERFORMANCES. 1ST, 2ND, AND 3RD REPRESENT THE THREE WINNING RESULTS DISTRIBUTED ON THE WEBSITE OF BCI COMPETITION IV [29]. OUR METHODS ARE MARKED IN BOLDFACE

	mean kappa	S1	S2	S3	S4	S5	S6	S7	S8	S9
TSSM+LDA[31]	0.59	0.77	0.33	0.77	0.51	0.35	0.36	0.71	0.72	**0.83**
GLVQ-AIRM	0.59	0.79	0.32	0.76	**0.55**	0.34	0.36	0.69	0.71	*0.80*
WaSF ConvNet [28]	0.58	0.63	0.32	0.75	0.44	**0.60**	**0.38**	0.69	0.71	0.73
GLVQ-LEML-FM	0.58	**0.80**	0.31	**0.81**	**0.55**	0.35	0.32	0.65	0.69	0.74
1^{st}	0.57	0.68	0.42	0.75	0.48	0.40	0.27	**0.77**	**0.75**	0.61
TSLDA[30]	0.57	0.74	0.38	0.72	0.50	0.26	0.34	0.69	0.71	0.76
GLVQ-LEML-LEM	0.56	0.78	0.30	0.76	0.51	0.33	0.31	0.65	0.68	0.74
GLVQ-LEM	0.55	0.74	0.26	0.78	0.50	0.31	0.30	0.69	0.67	0.70
LGMLVQ-LEM	0.53	0.70	0.27	0.73	0.45	0.30	0.26	0.69	0.63	0.75
MDRM [30]	0.52	0.75	0.37	0.66	0.53	0.29	0.27	0.56	0.58	0.68
2^{nd}	0.52	0.69	0.34	0.71	0.44	0.16	0.21	0.66	0.73	0.69
GRLVQ-LEM	0.52	0.70	0.21	**0.81**	0.39	0.21	0.29	0.69	0.64	0.74
GMLVQ-LEM	0.51	0.72	0.25	0.78	0.46	0.24	0.23	0.55	0.63	0.72
3^{rd}	0.31	0.38	0.18	0.48	0.33	0.07	0.14	0. 29	0. 49	0.44

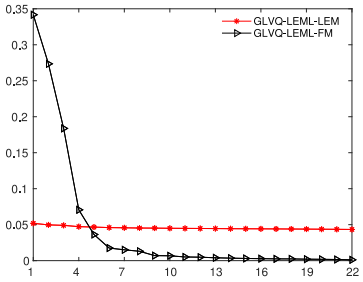


Fig. 9. Eigenvalues of the learned metric on subject 1 of the BCIIV2a dataset. Three prototypes were learned for each class. The method were trained for 100 epochs.

Compared to other manifold-related methods in the literature, particularly, GLVQ-AIRM, TSLDA, MDRM, and TSSM+LDA, our proposed method obtained comparable results. The aforementioned methods are all developed on the SPD manifold equipped with AIRM. The GLVQ-LEM method obtained worse performance than that of the GLVQ-AIRM method, suggesting that AIRM is more suitable for EEG data. The AIRM preserves the geodesic distance in different domains. To be more specific, the distance between two EEG trials will be the same in both the signal domain and source domain. Then, the calculation using the spatial covariance matrix computed from the recorded EEG signal is equivalent to that from the real source. The noise or artifact will not affect the computation. However, LEM does not have such nice property. Thus, AIRM is more suitable than LEM when dealing with EEG data. Incorporating metric learning in the GLVQ based on LEM, that is, GLVQ-LEML-FM, the method reaches comparable performance to that of GLVQ-AIRM. Moreover, the LEM-based method is much computationally efficient than AIRM-based methods.

Compared with WaSF ConvNet, our GLVQ-LEML-FM obtained comparable performance. This may because the number of training instances per subject is small. More importantly, our GLVQ-LEML-FM method is computationally faster than WaSF ConvNet, especially during the test. During the situation where a quick response is needed, such as real-time control, our method might be a nicer choice.

VII. CONCLUSION

In this article, we have developed novel methods for the classification of SPD matrix-formed data in the framework of LVQ. We have first extended Euclidean GLVQ to the space of SPD matrices by exploiting the LEM-induced geodesic distance (GLVQ-LEM).

We then extend GLVQ-LEM with metric learning in two different approaches. The first approach is simply to adapt metric in the space of vectorized log-transformed SPD matrices. This approach may distort the geometrical structure of the logarithm space derived from SPD matrices, as well as result in low computational efficiency since it needs to learn a much larger distance metric than the data dimension. The second approach learns a tangent map directly projecting the matrix logarithms in the original tangent space to a possibly more discriminative tangent space. The distance metric is much smaller, leading to improved computational efficiency.

Empirical experiments conducted on multiple datasets of different nature, that is, synthetic data, image data, and EEG classification data, showed the good performance of our methods. Our proposed methods, particularly, GLVQ-LEML, obtained competitive performances compared to other existing methods. This work, following our previous study of extending GLVQ to the manifold of SPD matrices equipped with AIRM [13], shows that LVQ methods can be competitive learning methods dedicated to Riemannian manifolds. In the future, we, therefore, intend to follow two lines. One line is to extend GLVQ to other types of Riemannian manifolds. The other line is to extend other LVQ methods using the framework presented in this article.

APPENDIX A
DERIVATION OF RIEMANNIAN GRADIENTS

A. Derivation of the Riemannian Gradient $\nabla_{\mathbf{V}_J} E_i$

Since the trace operator is invariant under cyclic permutations, that is, $\mathrm{Tr}(A_1A_2A_3) = \mathrm{Tr}(A_2A_3A_1) = \mathrm{Tr}(A_3A_2A_1)$, the Riemannian gradient $\nabla_{\mathbf{V}_J} E_i$ can be calculated as follows:

$$\left\langle \mathbf{V}_J, \nabla_{\mathbf{V}_J} E_i \right\rangle_{\mathbf{W}_J} = \frac{\mathrm{d}}{\mathrm{d}t} E_i(\gamma_J(t), \mathbf{W}_K)\Big|_{t=0}$$

$$
\begin{aligned}
&= g_K \frac{\mathrm{d}}{\mathrm{d}t}\delta_{LE}(\mathbf{X}_i, \gamma_J(t))\Big|_{t=0} \\
&= g_K \frac{\mathrm{d}}{\mathrm{d}t}\mathrm{Tr}\Big[(\log \mathbf{X}_i - \log \gamma_J(t))^2\Big]\Big|_{t=0} \\
&= -2g_K \mathrm{Tr}\Big[(\log \mathbf{X}_i - \log \gamma_J(t))\frac{\mathrm{d}}{\mathrm{d}t}\log \gamma_J(t)\Big]\Big|_{t=0} \\
&= -2g_K \mathrm{Tr}\Big[(\log \mathbf{X}_i - \log \mathbf{W}_J)\mathbf{W}_J{}^{-1}\mathbf{V}_J\Big] \\
&= -2g_K \mathrm{Tr}\Big[\mathbf{W}_J{}^{-1}\mathbf{V}_J\mathbf{W}_J^{-1}\mathbf{W}_J(\log \mathbf{X}_i - \log \mathbf{W}_J)\Big] \\
&= \langle \mathbf{V}_J, -2g_K \mathbf{W}_J(\log \mathbf{X}_i - \log \mathbf{W}_J)\rangle_{\mathbf{W}_J}. \qquad (42)
\end{aligned}
$$

Consequently, the gradient $\nabla_{\mathbf{V}_J} E_i$ is given as follows:

$$\nabla_{\mathbf{V}_J} E_i = -2g_K \mathbf{W}_J(\log \mathbf{X}_i - \log \mathbf{W}_J). \qquad (43)$$

B. Derivation of the Riemannian Gradient $\nabla_{\mathbf{V}_J} E_i^{\mathbf{Q}}$

The Riemannian gradient $\nabla_{\mathbf{V}_J} E_i^{\mathbf{Q}}$ can be computed as

$$
\begin{aligned}
\Big\langle \mathbf{V}_J, \nabla_{\mathbf{V}_J} E_i^{\mathbf{Q}}\Big\rangle_{\mathbf{W}_J} &= \frac{\mathrm{d}}{\mathrm{d}t} E_i^{\mathbf{Q}}(\gamma_J(t), \mathbf{W}_K, \mathbf{Q})\Big|_{t=0} \\
&= f_K \frac{\mathrm{d}}{\mathrm{d}t}\delta_{LE}^{\mathbf{Q}}(\mathbf{X}_i, \gamma_J(t))\Big|_{t=0} \\
&= f_K \frac{\mathrm{d}}{\mathrm{d}t}\mathrm{Tr}\Big[\mathbf{Q}(\log \mathbf{X}_i - \log \gamma_J(t))^2\Big]\Big|_{t=0} \\
&= -2f_K \mathrm{Tr}\Big[\mathbf{Q}(\log \mathbf{X}_i - \log \gamma_J(t))\frac{\mathrm{d}\log \gamma_J(t)}{\mathrm{d}t}\Big]\Big|_{t=0} \\
&= -2f_K \mathrm{Tr}\Big[\mathbf{Q}(\log \mathbf{X}_i - \log \mathbf{W}_J)\mathbf{W}_J{}^{-1}\mathbf{V}_J\Big] \\
&= \langle \mathbf{V}_J, -2f_K \mathbf{W}_J\mathbf{Q}(\log \mathbf{X}_i - \log \mathbf{W}_J)\rangle_{\mathbf{W}_J}. \qquad (44)
\end{aligned}
$$

Consequently

$$\nabla_{\mathbf{V}_J} E_i^{\mathbf{Q}} = -2f_K \mathbf{W}_J\mathbf{Q}(\log \mathbf{X}_i - \log \mathbf{W}_J). \qquad (45)$$

C. Derivation of the Riemannian Gradient $\nabla_{\mathbf{V}_\mathbf{Q}} E_i^{\mathbf{Q}}$

The initial speed vector $\mathbf{V}_\mathbf{Q}$ is unknown. The minimum of the cost function can be achieved when the initial speed vector of the curve is parallel to the gradient of the cost function. Hence, the Riemannian gradient of the cost function in the direction $\mathbf{V}_\mathbf{Q}$ evaluated at point $\mathbf{Q}$ denoted by $\nabla_{\mathbf{V}_\mathbf{Q}} E_i^{\mathbf{Q}}$ can still be computed as

$$
\begin{aligned}
\Big\langle \mathbf{V}_\mathbf{Q}, \nabla_{\mathbf{V}_\mathbf{Q}} E_i^{\mathbf{Q}}\Big\rangle_{\mathbf{Q}} &= \frac{\mathrm{d}}{\mathrm{d}t} E_i^{\mathbf{Q}}(\mathbf{W}_J, \mathbf{W}_K, \gamma_\mathbf{Q}(t)\Big|_{t=0} \\
&= f_K \frac{\mathrm{d}}{\mathrm{d}t}\delta_{LE}^{\gamma_\mathbf{Q}}(\mathbf{X}_i, \mathbf{W}_J)\Big|_{t=0} - f_J \frac{\mathrm{d}}{\mathrm{d}t}\delta_{LE}^{\gamma_\mathbf{Q}}(\mathbf{X}_i, \mathbf{W}_K)\Big|_{t=0}. \qquad (46)
\end{aligned}
$$

Note that the curve $\gamma_\mathbf{Q}(t)$ at the $t = 0$ reaches the point $\mathbf{Q}$, that is, $\gamma_\mathbf{Q}(0) = \mathbf{Q}$. Thus, $\delta_{LE}^{\gamma_\mathbf{Q}}(\mathbf{X}_i, \mathbf{W}_J) = \delta_J^{\mathbf{Q}}$ at $t = 0$.

As trace operator is invariant under cyclic permutations, if the operatee is products of three symmetric matrices, any permutation is allowed, we can have

$$
\begin{aligned}
&\frac{\mathrm{d}}{\mathrm{d}t}\delta_{LE}^{\gamma_\mathbf{Q}}(\mathbf{X}_i, \mathbf{W})\Big|_{t=0} \\
&= \frac{\mathrm{d}}{\mathrm{d}t}\mathrm{Tr}\Big[\gamma_\mathbf{Q}(t)(\log \mathbf{X}_i - \log \mathbf{W})^2\Big]\Big|_{t=0} \\
&= \mathrm{Tr}\Big[(\log \mathbf{X}_i - \log \mathbf{W})^2 \frac{\mathrm{d}\gamma_\mathbf{Q}(t)}{\mathrm{d}t}\Big]\Big|_{t=0} \\
&= \mathrm{Tr}\Big[(\log \mathbf{X}_i - \log \mathbf{W})^2 \mathbf{V}_\mathbf{Q}\Big] \\
&= \mathrm{Tr}\Big(\mathbf{Q}\mathbf{Q}^{-1}\mathbf{V}_\mathbf{Q}\mathbf{Q}^{-1}\mathbf{Q}(\log \mathbf{X}_i - \log \mathbf{W})^2\Big) \\
&= \mathrm{Tr}\Big(\mathbf{Q}^{-1}\mathbf{V}_\mathbf{Q}\mathbf{Q}^{-1}\mathbf{Q}(\log \mathbf{X}_i - \log \mathbf{W})^2\mathbf{Q}\Big) \\
&= \Big\langle \mathbf{V}_\mathbf{Q}, \mathbf{Q}(\log \mathbf{X}_i - \log \mathbf{W})^2\mathbf{Q}\Big\rangle_{\mathbf{Q}}. \qquad (47)
\end{aligned}
$$

Therefore, we can have

$$
\begin{aligned}
&\Big\langle \mathbf{V}_\mathbf{Q}, \nabla_{V_\mathbf{Q}} E_i^{\mathbf{Q}}\Big\rangle_{\mathbf{Q}} \\
&= f_K\Big\langle \mathbf{V}_\mathbf{Q}, \mathbf{Q}(\log \mathbf{X}_i - \log \mathbf{W}_J)^2\mathbf{Q}\Big\rangle_{\mathbf{Q}} \\
&\quad - f_J\Big\langle \mathbf{V}_\mathbf{Q}, \mathbf{Q}(\log \mathbf{X}_i - \log \mathbf{W}_K)^2\mathbf{Q}\Big\rangle_{\mathbf{Q}}.
\end{aligned}
$$

Consequently

$$
\begin{aligned}
\nabla_{V_\mathbf{Q}} E_i^{\mathbf{Q}} &= f_K\mathbf{Q}(\log \mathbf{X}_i - \log \mathbf{W}_J)^2\mathbf{Q} \\
&\quad - f_J\mathbf{Q}(\log \mathbf{X}_i - \log \mathbf{W}_J)^2\mathbf{Q}. \qquad (48)
\end{aligned}
$$

D. Rimannian Gradient $\overline{\nabla_{\bar{\xi}_\mathbf{G}} E_i^{\mathbf{Q}}}$

As $\mathbf{T}$ and $\mathbf{U}$ are both symmetric, $\partial\mathrm{Tr}(\mathbf{X}) = \mathrm{Tr}(\partial\mathbf{X})$ and in general for suitable $\mathbf{B}$, $([\partial\mathrm{Tr}(\mathbf{B}\mathbf{X}\mathbf{X}^T)]/\partial\mathbf{X}) = (\mathbf{B} + \mathbf{B}^T)\mathbf{X}$ we have

$$
\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\delta_{LE}^{\gamma_\mathbf{Q}}(\mathbf{X}_i, \mathbf{W})\Big|_{t=0} &= \frac{\mathrm{d}}{\mathrm{d}t}\delta_{LE}^{\gamma_\mathbf{G}\gamma_\mathbf{G}^T}(\mathbf{X}_i, \mathbf{W})\Big|_{t=0} \\
&= \frac{\mathrm{d}}{\mathrm{d}t}\mathrm{Tr}\Big[\gamma_\mathbf{G}(t)(\gamma_\mathbf{G}(t))^T(\mathbf{T}_i - \mathbf{U})^2\Big]\Big|_{t=0} \\
&= \mathrm{Tr}\Big[2(\gamma_\mathbf{G}(t))^T(\mathbf{T}_i - \mathbf{U})^2\frac{\mathrm{d}\gamma_\mathbf{G}(t)}{\mathrm{d}t}\Big]\Big|_{t=0} \\
&= \mathrm{Tr}\Big[2\mathbf{G}^T(\mathbf{T}_i - \mathbf{U})^2\bar{\xi}_\mathbf{G}\Big] \\
&= \bar{g}_\mathbf{G}\Big(\bar{\xi}_\mathbf{G}, 2(\mathbf{T}_i - \mathbf{U})^2\mathbf{G}\Big). \qquad (49)
\end{aligned}
$$

Thus, the horizontal gradient of the cost function $E_i^{\mathbf{Q}}$ at $\mathbf{G}$ can be computed

$$
\begin{aligned}
\bar{g}_\mathbf{G}\Big(\bar{\xi}_\mathbf{G}, \overline{\nabla_{\bar{\xi}_\mathbf{G}} E_i^{\mathbf{Q}}}\Big) &= \frac{\mathrm{d}}{\mathrm{d}t} E_i^{\mathbf{Q}}(\gamma_J(t), \mathbf{W}_K, \mathbf{Q})\Big|_{t=0} \\
&= f_K\bar{g}_\mathbf{G}\Big(\bar{\xi}_\mathbf{G}, 2(\mathbf{T}_i - \mathbf{U}_J)^2\mathbf{G}\Big) \\
&\quad - f_J\bar{g}_\mathbf{G}\Big(\bar{\xi}_\mathbf{G}, 2(\mathbf{T}_i - \mathbf{U}_K)^2\mathbf{G}\Big). \qquad (50)
\end{aligned}
$$

Consequently, the horizontal gradient of the cost function at $\mathbf{G}$ is given

$$\overline{\nabla_{\bar{\xi}_\mathbf{G}} E_i^{\mathbf{Q}}} = 2f_K(\mathbf{T}_i - \mathbf{U}_J)^2\mathbf{G} - 2f_J(\mathbf{T}_i - \mathbf{U}_K)^2\mathbf{G}. \qquad (51)$$

REFERENCES

[1] B. Hammer and T. Villmann, "Generalized relevance learning vector quantization," *Neural Netw.*, vol. 15, nos. 8–9, pp. 1059–1068, 2002.

[2] T. Kohonen, "Learning vector quantization for pattern recognition," Helsinki Univ. Technol., Espoo, Finland, Rep. TKKF-A601, 1986.

[3] A. Sato and K. Yamada, "Generalized learning vector quantization," in *Advances in Neural Information Processing Systems*, D. Touretzky, M. Mozer, and M. Hasselmo, Eds., vol. 8. Cambridge, MA, USA: MIT Press, 1996, pp. 423–429.

[4] P. Schneider, P. Biehl, and B. Hammer, "Adaptive relevance matrices in learning vector quantization," *Neural Comput.*, vol. 21, no. 12, pp. 3532–3561, 2009.

[5] X. Penne, P. Fillard, and N. Ayache, "A Riemannian framework for tensor computing," *Int. J. Comput. Vis.*, vol. 66, no. 1, pp. 41–66, 2006.

[6] V. Arsigny, P. Filllard, X. Pennec, and N. Ayache, "Geometric means in a novel vector space structure on symmetric positive-definite matrices," *SIAM J. Matrix Anal. Appl.*, vol. 29, no. 1, pp. 328–347, 2007.

[7] P. T. Fletcher, C. Lu, S. M. Pizer, and S. Joshi, "Principal geodesic analysis for the study of nonlinear statistics of shape," *IEEE Trans. Med. Imag.*, vol. 23, no. 8, pp. 995–1005, Aug. 2004.

[8] Z. Cui, W. Li, D. Xu, S. Shan, X. Chen, and X. Li, "Flowing on Riemannian manifold: Domain adaptation by shifting covariance," *IEEE Trans. Cybern.*, vol. 44, no. 12, pp. 2264–2273, Dec. 2014.

[9] S. Jayasumana, R. Hartley, M. Salzmann, H. Li, and M. Harandi, "Kernel methods on Riemannian manifolds with gaussian RBF kernels," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 12, pp. 2464–2477, Dec. 2015.

[10] F. Yger, M. Berar, and F. Lotte, "Riemannian approaches in brain-computer interfaces: A review," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 25, no. 10, pp. 1753–1762, Oct. 2017.

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

[12] V. Arsigny, P. Fillard, X. Pennec, and N. Ayache, "Log-Euclidean metrics for fast and simple calculus on diffusion tensors," *Magn. Reson. Med.*, vol. 56, no. 2, pp. 411–421, 2006.

[13] F. Tang, M. Fan, and P. Tiňo, "Generalized learning Riemannian space quantization: A case study on Riemannian manifold of SPD matrices," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 281–292, Jan. 2021.

[14] H.-J. Ye, D.-C. Zhan, Y. Jiang, and Z.-H. Zhou, "What makes objects similar: A unified multi-metric learning approach," *IEEE Trans Pattern Anal. Mach. Intell.*, vol. 41, no. 5, pp. 1257–1270, May 2019.

[15] J. Yu, Y. Rui, Y. Y. Tang, and D. Tao, "High-order distance-based Multiview stochastic learning in image classification," *IEEE Trans. Cybern.*, vol. 44, no. 12, pp. 2431–2442, Dec. 2014.

[16] J. Yu, X. Yang, F. Gao, and D. Tao, "Deep multimodal distance metric learning using click constraints for image ranking," *IEEE Trans. Cybern.*, vol. 47, no. 12, pp. 4014–4024, Dec. 2017.

[17] A. Bohnsack, K. Domaschke, M. Kaden, M. Lange, and T. Villmann, "Learning matrix quantization and relevance learning based on Schatten-p-norms," *Neurocomputing*, vol. 192, pp. 104–114, Jun. 2016.

[18] R. Vemulapalli and D. W. Jacobs, "Riemannian metric learning for symmetric positive definite matrices," 2015, *arXiv:1501.02393*.

[19] Z. Huang, R. Wang, S. Shan, X. Li, and X. Chen, "Log-euclidean metric learning on symmetric positive definite manifold with application to image set classification," in *Proc. ICML*, 2015, pp. 720–729.

[20] F. Qi *et al.*, "Spatiotemporal-filtering-based channel selection for single-trial EEG classification," *IEEE Trans. Cybern.*, vol. 51, no. 2, pp. 558–567, Feb. 2021.

[21] G. Meyer, S. Bonnabel, and R. Sepulchre, "Regression on fixed-rank positive Semidefinite matrices: A Riemannian approach," *J. Mach. Learn. Res.*, vol. 12, pp. 593–625, Feb. 2011.

[22] B. Leibe and B. Schiele, "Analyzing appearance and contour based methods for object categorization," in *Proc. CVPR*, 2003, p. 409.

[23] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. ICLR*, 2015, pp. 1–14.

[24] Z. Huang and L. Van Gool, "A Riemannian network for SPD matrix learning," in *Proc. AAAI*, 2017, pp. 2036–2042.

[25] A. Dhall, R. Goecke, J. Joshi, K. Sikka, and T. Gedeon, "Emotion recognition in the wild challenge 2014: Baseline, data and protocol," in *Proc. ICMI*, 2014, pp. 461–466.

[26] C. Brunner, R. Leeb, G. R. Müller-Putz, A. Schlogl, and G. Pfurtscheller, *BCI Competition 2008—Graz Data Set A*, Graz Univ. Technol., Graz, Austria, 2008, pp. 136–142.

[27] Y. Zhang, C. S. Nam, G. Zhou, J. Jin, X. Wang, and A. Cichocki, "Temporally constrained sparse group spatial patterns for motor imagery BCI," *IEEE Trans. Cybern.*, vol. 49, no. 9, pp. 3322–3332, Sep. 2019.

[28] D. Zhao, F. Tang, B. Si, and X. Feng, "Learning joint space-time-frequency features for EEG decoding on small labeled data," *Neural Netw.*, vol. 114, pp. 67–77, Jun. 2019.

[29] M. Tangermann *et al.*, "Review of the BCI competition IV," *Front. Neurosci.*, vol. 6, p. 55, Jul. 2012.

[30] A. Barachant, S. Bonnet, M. Congedo, and C. Jutten, "Multiclass brain-computer interface classification by Riemannian geometry," *IEEE Trans. Biomed. Eng.*, vol. 59, no. 4, pp. 920–928, Apr. 2012.

[31] X. Xie, Z. L. Yu, H. Lu, Z. Gu, and Y. Li, "Motor imagery classification based on bilinear sub-manifold learning of symmetric positive-definite matrices," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 25, no. 6, pp. 504–516, Jun. 2017.

[32] F. Tang, L. Adam, and B. Si, "Group feature selection with multi-class support vector machine," *Neurocomputing*, vol. 317, pp. 42–49, Nov. 2018.

Fengzhen Tang received the Ph.D. degree in computer science from the University of Birmingham, Birmingham, U.K., in 2015.

She is an Associate Researcher with the State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang, China. Her research interests include machine learning, computational neuroscience, robotics, and signal processing.

Peter Tiňo received the Ph.D. degree from the Institute of Control Theory and Robotics, Slovak Academy of Sciences, Bratislava, Slovakia, in 1997.

He is a Professor and the Chair of Complex and Adaptive Systems with the School of Computer Science, University of Birmingham, Birmingham, U.K. His research interests include artificial intelligence, machine learning, statistical pattern recognition, and natural computation.

Haibin Yu (Senior Member, IEEE) received the B.S. and M.S. degrees in automation engineering and the Ph.D. degree in control theory and control engineering from Northeastern University, Shenyang, China, in 1984, 1987, and 1997, respectively.

He is currently the President and a Professor with the Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang. His research interests include wireless sensor networks, industrial communication and networked control, industrial automation, and intelligent manufacturing.