Generalized Learning Riemannian Space Quantization: A Case Study on Riemannian Manifold of SPD Matrices

Fengzhen Tang, Mengling Fan, and Peter Tiňo

***Abstract*— Learning vector quantization (LVQ) is a simple and efficient classification method, enjoying great popularity. However, in many classification scenarios, such as electroencephalogram (EEG) classification, the input features are represented by symmetric positive-definite (SPD) matrices that live in a curved manifold rather than vectors that live in the flat Euclidean space. In this article, we propose a new classification method for data points that live in the curved Riemannian manifolds in the framework of LVQ. The proposed method alters generalized LVQ (GLVQ) with the Euclidean distance to the one operating under the appropriate Riemannian metric. We instantiate the proposed method for the Riemannian manifold of SPD matrices equipped with the Riemannian natural metric. Empirical investigations on synthetic data and real-world motor imagery EEG data demonstrate that the performance of the proposed generalized learning Riemannian space quantization can significantly outperform the Euclidean GLVQ, generalized relevance LVQ (GRLVQ), and generalized matrix LVQ (GMLVQ). The proposed method also shows competitive performance to the state-of-the-art methods on the EEG classification of motor imagery tasks.**

***Index Terms*— Generalized learning vector quantization (GLVQ), learning vector quantization (LVQ), Riemannian geodesic distances, Riemannian manifold.**

I. Introduction

LEARNING vector quantization (LVQ), introduced by Kohonen [1] in 1986, is a prototype-based supervised classification algorithm based on metric comparisons of data. The approach has enjoyed great popularity because of its simplicity, intuitive nature, and natural accommodation of multiclass classification problems. Unlike deep networks, the LVQ system is straightforward to interpret. The classifier constructed by LVQ is parameterized by a set of labeled prototypes living in the data space. The classification of an unknown instance takes place as an inference of the class of the closest prototype in terms of the involved metric. The learning rules of LVQ are typically based on intuitive Hebbian Learning. Thus, the implementation and realization of the method are very simple. Unlike many alternatives, such as the perceptron or support vector machines, which are in their basic form restricted to only two classes, LVQ can naturally deal with any number of classes without making the classification rule or learning algorithm more complicated. Indeed, LVQ has been used in a variety of applications, such as image and signal processing, the biomedical field and medicine, and industry [2].

In brain–computer interface (BCI), motor imagery is a very promising modality compared with other alternatives, as the subject voluntarily produces electroencephalogram (EEG) signal by imaging movements of different parts of the body, without external stimulus. The topological representation and band power change of brain signals during motor imagery tasks are well-known. Imaging movements of different body parts will activate (or deactivate) the activities of a different area in the motor cortex of the brain, e. g. roughly, imagination of right hand movement associates with C_3 electrode, left hand C_4, and foot C_z [3]. Thus, in motor imagery classification, the spatial covariance matrix of the EEG signal provides enough discriminative information for different classes. This is corroborated by the most commonly used common spatial pattern (CSP) [4] algorithm, which is completely based on the estimation of the spatial covariance matrices, where spatial filters can be derived to enhance the class separability. Thus, the EEG signal can be represented by the corresponding sample covariance matrix, summarizing spatial information in the signal with temporal content integrated out [5]–[7].

Current LVQ and its extensions are designed to deal with data items in the form of finite-dimensional real vectors. Nevertheless, covariance matrices are symmetric positive definite (SPD) and, as such, live in a curved manifold, rather than the flat Euclidean space. The structure information of the original matrix is useful and informative for the classification task. Directly applying the existing vector-based learning algorithms through vectorizing matrices into vectors may lead to poor generalization performance, as vectorization may destroy the crucial tensor structure of matrix data. Several

Manuscript received May 20, 2019; revised December 5, 2019; accepted March 2, 2020. This work was supported in part by the National Natural Science Foundation of China under Grant 61803369, in part by the CAS Pioneer Hundred Talents Program under Grant Y8A1220104, 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 Frontier Science Research Project of the Chinese Academy of Sciences under Grant QYZDY-SSW-JSC005. The work of Peter Tiňo was supported by the European Commission Horizon 2020 Innovative Training Network SUNDAIL under Project 721463. *(Corresponding author: Fengzhen Tang.)*

Fengzhen Tang and Mengling Fan are with the State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China, also with the Institute for Robotics and Intelligent Manufacturing, Chinese Academy of Sciences, Shenyang 110016, China, and also with the School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing 100049, China (e-mail: tangfengzhen@sia.cn; mengling@sia.ac.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 of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2020.2978514

approaches extending vector-based learning algorithms to SPD matrix data targeted for motor imagery classification have been suggested, e.g., the Fisher geodesic discriminant analysis (FGDA) [3], minimum distance to the Riemannian mean (MDRM) [8], and tangent space linear discriminant analysis (TSLDA) [8]. MDRM is a straightforward extension of the minimum distance to mean classification algorithm using the Riemannian geometric distance and the Riemannian geometric mean. It learns a cluster center (i.e., the Riemannian geometric mean) for each class of instances and predicts the class label of an unknown instance by finding the center with the shortest Riemannian geometric distance to the instance. The classification performance of MDRM was shown inferior to other alternatives [9]. Unlike the MDRM method, the FGDA and TSLDA methods, based on discriminant analysis, are intrinsically binary classifiers. To deal with the multiclass classification of SPD matrices, we would need to decouple the problem into multiple binary classification problems using appropriate heuristics (e.g., one-versus-one and or one-versus-the rest), thus not fully taking advantages of the overall picture of the multiclass structure. Both FGDA and TSLDA project SPD matrices from the training set onto the tangent space at their Riemannian geometric mean. Standard Euclidean Fisher linear discriminate analysis and standard Euclidean linear discriminate analysis are then applied in the tangent space, respectively. However, mapping data to a tangent space at a particular point gives a first-order approximation of the data, which can be distorted, especially in regions that are far from the origin of the tangent space.

In this article, we propose to extend the successful generalized LVQ (GLVQ) method [9] to a curved Riemannian manifold and specify the general framework for the Riemannian manifold of SPD matrices equipped with the Riemannian natural metric tensor. The proposed method is named generalized learning Riemannian space quantization (GLRSQ). To show the appropriateness of our proposed GLRSQ method, we also create simple and naive extensions of GLVQ, generalized relevance LVQ (GRLVQ) [10], and generalized matrix LVQ (GMLVQ) [11] to deal with SPD matrix classification through simply concatenating the upper triangle of SPD matrices into vectors and then applying the standard learning rule of GLVQ, GRLVQ, or GMLVQ targeted for vectors, totally ignoring the nonlinear structure of the Riemannian manifold. To ensure that the learned prototype is a positive definite, the updated vector is reshaped into a symmetric matrix, and the matrix is then projected onto its closest SPD matrix under the Frobenius norm. In contrast, the proposed GLRSQ is naturally formulated in the appropriate Riemannian space, obtaining superior generalization performance to the naive extension of LVQs on both synthetic and real-world data sets. In addition to controlled experiments, we verify the GLRSQ method on EEG classification of motor imagery tasks, obtaining competitive performance to the state-of-the-art methods on this problem.

The proposed GLRSQ method is different from the existing methods for SPD matrix data. In particular, our GLRSQ is implemented by the Riemannian stochastic gradient descent algorithm. Unlike TSLDA or FGDA, it does not need to approximate the data by projections to the tangent space at a particular point on the manifold. Note that the Riemannian stochastic gradient descent algorithm also introduces projections of SPD matrices to the tangent spaces but only at a local scale. The GLRSQ method shares some properties with MDRM, but it is potentially far more powerful than MDRM in that, if needed, GLRSQ can learn multiple representatives for each class, as opposed to only one center per class in MDRM. Even with one prototype per class, the GLRSQ method shows superior performance to MDRM since MDRM finds the class representatives in an unsupervised manner as class conditional means of the training data. Our method is also different from recent GLVQ variants with tangent distances for matrix-shaped data [12], [13]. The main motivation there is to deal with invariances in the data (e.g., image objects subject to rotations, scaling, and shifts) by automatically learning a lower dimensional tangent basis that can be used to calculate tangent distances between data points and class prototypes, providing the best class discrimination. It is assumed that the inherent data manifold dimensionality is much lower than that of the embedding data space. The data manifold is considered implicitly and exclusively based on the data sample. In contrast, we know the Riemannian manifold structure of SPD matrices and take full use of it when formulating the GLRSQ method. In connection to this article, it is worth mentioning that variants of vector quantization capable of operating on the Grassman manifolds have recently been developed [14], [15]. Grassman manifold $\mathcal{G}_n^k$ is a space parameterizing all k-dimensional linear subspaces of the n-dimensional embedding vector space. Each point on $\mathcal{G}_n^k$ can be thought of as a k-dimensional vector space represented through a representative $n \times k$ basis matrix. If $n = k$, the Grassman manifold is trivially a singleton—the embedding space itself. In our case, the manifold is the space of full rank squared $n \times n$ SPD matrices, requiring a very different metric structure from that used in Grassmannians.

The rest of this article is organized as follows. In Section II, we briefly introduce LVQ. In Section III, we derive the novel general framework of GLRSQ, and in Section IV, we specify the GLRSQ for the Riemannian manifold of SPD matrices equipped with the Riemannian natural metric. Experimental results are in Section V. Main findings and conclusions are presented in Section VI.

II. Learning Vector Quantization

Our approach is developed within the framework of LVQ [1]. In this section, we will briefly introduce the concept of LVQ.

Consider a training data set $(\boldsymbol{x}_i, y_i) \in \mathbb{R}^n \times \{1, \ldots, C\}$, $i = 1, \ldots, m$, where n is the input dimension, C is the number of classes, and m is the number of training examples. A typical LVQ classifier consists M ($M \geq C$) prototypes $\boldsymbol{w}_j \in \mathbb{R}^n$, labeled by $c(\boldsymbol{w}_j) \in \{1, \ldots, C\}$, $j = 1, 2, \ldots, M$. The classification is implemented in a winner-takes-all scheme. For a data point $\boldsymbol{x} \in \mathbb{R}^n$, the output class is determined by the class label of its closest prototype, i.e., $\hat{y}(\boldsymbol{x}) := c(\boldsymbol{w}_{j_*})$ such that $j_* = \arg\min_j d(\boldsymbol{x}, \boldsymbol{w}_j)$, where $d(\cdot, \cdot)$ is a distance measure in $\mathbb{R}^n$. Each labeled prototype $\boldsymbol{w}_j$ with label $c(\boldsymbol{w}_j)$

defines a receptive filed in the input space—a set of points that pick this prototype as their winner. The goal of learning the LVQ classifier is to adapt prototypes automatically such that the class labels of data points within the receptive field coincide with the label of the respective prototype as much as possible.

A generalization of LVQ, termed GLVQ, was introduced in [9]. In GLVQ, the prototypes are updated based on the steepest descent method on a well-defined cost function. The cost function is determined so that the obtained learning rule satisfies the convergence condition. In the training phase of GLVQ, for each labeled input $\boldsymbol{x}_i$, a pair of prototypes will be updated. The closest prototype to $\boldsymbol{x}_i$ with the same label y_i (correct prototype) is rewarded by dragging it closer to $\boldsymbol{x}_i$, while the closest prototype with a different label (incorrect prototype) is penalized by pushing it away from $\boldsymbol{x}_i$.

We collect all the model parameters (i.e., labeled prototype positions) in vector $\boldsymbol{w}$ and denote by $\boldsymbol{w}_J$ and $\boldsymbol{w}_K$ the closest correct and incorrect prototypes to the training example $\boldsymbol{x}_i$, respectively. The GLVQ cost function is defined by

$$E(\boldsymbol{w}) = \sum_{i=1}^{m} \Phi(f(\boldsymbol{x}_i, \boldsymbol{w})) \tag{1}$$

$$f(\boldsymbol{x}_i, \boldsymbol{w}) = \frac{d(\boldsymbol{x}_i, \boldsymbol{w}_J) - d(\boldsymbol{x}_i, \boldsymbol{w}_K)}{d(\boldsymbol{x}_i, \boldsymbol{w}_J) + d(\boldsymbol{x}_i, \boldsymbol{w}_K)} \tag{2}$$

where $\Phi(\cdot)$ is a monotonically increasing scaling function [e.g., the identity $\Phi(x) = x$ or the logistic function $\Phi(x) = 1/(1 + e^{-x})$]. Interested reader can find more details on the meaning Φ in, e.g., [9] and [10].

For vector data, the squared Euclidean distance, i.e., $d(\boldsymbol{x}_i, \boldsymbol{w}_J) = |\boldsymbol{x}_i - \boldsymbol{w}_J|^2$ is commonly used in the cost function. The learning rule for prototypes can be obtained by minimizing the cost function based on stochastic steepest gradient descent algorithm. For a given instance $\boldsymbol{x}_i$, denoting $d_J = d(\boldsymbol{x}_i, \boldsymbol{w}_J)$ and $d_K = d(\boldsymbol{x}_i, \boldsymbol{w}_K)$, we obtain the following learning rules:

$$\Delta \boldsymbol{w}_J = \alpha(t) \frac{\partial \Phi}{\partial f} \frac{4 d_K}{(d_J + d_K)^2} (\boldsymbol{x}_i - \boldsymbol{w}_J) \tag{3}$$

$$\Delta \boldsymbol{w}_K = -\alpha(t) \frac{\partial \Phi}{\partial f} \frac{4 d_J}{(d_J + d_K)^2} (\boldsymbol{x}_i - \boldsymbol{w}_K) \tag{4}$$

where $0 < \alpha(t) < 1$ is the learning rate that satisfies $\sum_{t=1}^{\infty} \alpha(t) = \infty$ and $\sum_{t=1}^{\infty} \alpha^2(t) < \infty$.

The GLVQ algorithm is designed for data points living in Euclidean space. In Section III, we extend the generalized learning quantization algorithm to data points that live in a Riemannian Manifold.

III. Generalized Learning Riemannian Space Quantization on Riemannian Manifolds

In this section, we present the general framework of extending the GLVQ to a Riemannian manifold. Before we present the proposed method in detail, we briefly introduce several important concepts in the Riemannian manifold, including tangent spaces, geodesic curves, geodesic distance, and the Riemannian gradient. More detailed information can be found in the Appendix and in [16] and [17].

A. Riemannian Manifolds

A topological manifold (or simply manifold) is a topological space that is locally homeomorphic to the n-dimensional Euclidean space $\mathbb{R}^n$, where n is the dimension of the manifold. A differential manifold is a topological manifold that has a defined differential structure. To any point on such a manifold, we can attach the tangent space, which can be viewed as a vector space containing tangent vectors of all possible curves on the manifold passing through that point.

A Riemannian manifold is a differentiable manifold equipped with a smoothly varying inner product acting on tangent spaces. The family of inner products on all tangent spaces is referred to as the Riemannian metric tensor. With the definition of the inner product, the angles between tangent vectors from the same tangent space, as well as the length of each tangent vector, can be quantified. In particular, this allows us to measure the length of curves on the Riemannian manifolds by gluing together the lengths of infinitesimal scale tangent vectors along the curve. The (geodesic) distance between two distinct points on the manifold is then the shortest length among all curves connecting the two points on the manifold.

We denote a Riemannian manifold as $\mathcal{M}$ and its tangent space at point $\mathbf{X} \in \mathcal{M}$ by $\mathcal{T}_{\mathbf{X}}\mathcal{M}$. Note that we denote points on manifold $\mathcal{M}$ with bold capital letters (e.g., $\mathbf{X}$) since, in this article, we concentrate on the manifold of SPD matrices. Given any pair of vectors $\mathbf{V}, \mathbf{W} \in \mathcal{T}_{\mathbf{X}}\mathcal{M}$, their inner product $\langle \mathbf{V}, \mathbf{W} \rangle_{\mathbf{X}} \in \mathbb{R}$ is defined through the associated Riemannian metric tensor on $\mathcal{T}_{\mathbf{X}}\mathcal{M}$. A smooth mapping parameterized by $\gamma\colon \mathbb{R} \to \mathcal{M}\colon t \mapsto \gamma(t)$ is referred to as a curve in $\mathcal{M}$. Since the set of points forming the curve (the image of γ) can be parameterized in many ways, with a varying speed of parameterization, we will consider the parameterizations by arc length—so-called naturally parameterized curves. Loosely speaking, moving parameter t with constant speed in $[0, 1]$, the images $\gamma(t)$ should move with the constant speed with respect to the Riemannian metric. In other words, the ratio of the arc length $(\gamma(0), \gamma(t))$ to the total length of the curve is t.

Given two points $\mathbf{X}_1, \mathbf{X}_2 \in \mathcal{M}$, let $\gamma(t)\colon [0, 1] \to \mathcal{M}$ be a naturally parameterized curve connecting the two points, i.e., $\gamma(0) = \mathbf{X}_1$ and $\gamma(1) = \mathbf{X}_2$ with tangent vectors $\dot{\gamma}(t) \in \mathcal{T}_{\gamma(t)}\mathcal{M}$ along the way. The length of the curve γ is given by

$$L(\gamma) = \int_0^1 \sqrt{\langle \dot{\gamma}(t), \dot{\gamma}(t) \rangle_{\gamma(t)}} dt. \tag{5}$$

A curve that minimizes the distance between two points on the manifold is called a geodesic curve. Note that geodesics are not merely the curves of the shortest path between two points but they are also parameterized with "constant speed." Given two points, the geodesic curve connecting them may not be unique, as there can be a multitude of such geodesic curves, for example, geodesic curves connecting the north and south poles on a sphere. The length of any geodesic curve connecting $\mathbf{X}_1$ and $\mathbf{X}_2$ is referred to as the Riemannian geodesic distance between $\mathbf{X}_1$ and $\mathbf{X}_2$. If any two points can be

connected with a geodesic, the manifold is complete. However, without completeness, this is not necessarily true. In this article, we only consider geometrically complete Riemannian manifolds.

Given a point $\mathbf{X}$ on a curved manifold $\mathcal{M}$, the tangent space to $\mathcal{M}$ at $\mathbf{X}$ can usefully function as a vector space approximation of $\mathcal{M}$ only to within a certain neighborhood of $\mathbf{X}$ on $\mathcal{M}$. This is formalized by the notion of the injective radius. Loosely speaking, it defines the maximum radius of a ball $\mathcal{B}(\mathbf{X})$ in $\mathcal{M}$ centered at $\mathbf{X}$ so that there exists a diffeomorphism between $\mathcal{B}(\mathbf{X})$ and its image in the tangent space through the log map. For any point $\mathbf{X}' \in \mathcal{B}(\mathbf{X})$, there is the unique minimizing geodesic connecting $\mathbf{X}$ and $\mathbf{X}'$.

Given a smooth real-valued function $f(\mathbf{X})$ defined on $\mathcal{M}$, its Riemannian gradient $\nabla_{\mathbf{V}} f(\mathbf{X})$ at $\mathbf{X} \in \mathcal{M}$ in the direction of the vector $\mathbf{V} \in \mathcal{T}_{\mathbf{X}}\mathcal{M}$ measures the rate of change of the function f in the direction $\mathbf{V}$. Given a naturally parameterized smooth curve $\gamma\colon [0, 1] \to \mathcal{M}$, such that $\gamma(0) = \mathbf{X}$ and $\dot{\gamma}(0) = \mathbf{V}$, the composite function $f \circ \gamma\colon t \mapsto f(\gamma(t))$ is a smooth function from $\mathbb{R}$ to $\mathbb{R}$ with a well-defined classical derivative. Given the direction $\mathbf{V} \in \mathcal{T}_{\mathbf{X}}\mathcal{M}$, the Riemannian gradient $\nabla_{\mathbf{V}} f(\mathbf{X})$ is the unique tangent vector in $\mathcal{T}_{\mathbf{X}}\mathcal{M}$ such that

$$\langle \mathbf{V}, \nabla_{\mathbf{V}} f(\mathbf{X}) \rangle_{\mathbf{X}} = \frac{d}{dt} f(\gamma(t))\bigg|_{t=0}. \tag{6}$$

Thus, the computation of the Riemannian gradient can be performed through the calculation of the classical derivative of the composite function $f \circ \gamma$. The exponential map at point $\mathbf{X}$ denoted by $\mathrm{Exp}_{\mathbf{X}}$ is a map from the tangent space of the manifold $\mathcal{M}$ at point $\mathbf{X}$ to the manifold $\mathcal{M}$ itself, i.e., $\mathrm{Exp}_{\mathbf{X}}\colon \mathcal{T}_{\mathbf{X}}\mathcal{M} \to \mathcal{M}$, and the exponential map is defined by $\mathrm{Exp}_{\mathbf{X}}(\mathbf{V}) = \gamma(1)$. The exponential map provides a way to access to the corresponding point on the manifold, given a tangent vector.

B. Generalized Learning Riemannian Space Quantization on a General Riemannian Manifold

Consider a training data set $\{(\mathbf{X}_i, y_i)\}_{i=1}^m$, where $\mathbf{X}_i \in \mathcal{M}$ is the ith training point that lives on the Riemannian manifold $\mathcal{M}$ and $y_i \in \{1, \ldots, C\}$ is the label of this point. Assume that the classifier consists of M ($M \geq C$) prototypes $\mathbf{W}_j \in \mathcal{M}$ labeled by $c_j \in \{1, \ldots, C\}$. On a curved Riemannian manifold, the distance between a prototype $\mathbf{W}_j$ and a training instance $\mathbf{X}_i$ is the length of the geodesic curve between them, rather the straight line, as illustrated in Fig. 1. The closest correct prototype to an instance should be dragged toward it along the geodesic curve, rather than along the straight line between the prototype and the instance. Otherwise, the prototype may leave the manifold. In the same way, the closest incorrect prototype to the instance should be pushed away from the instance along the corresponding geodesic.

The cost function of the GLVQ classifier on the Riemannian Manifold can be directly extended from the original GLVQ [i.e., (1) and (2)] by replacing the squared Euclidean distance d by the squared Riemannian distance δ

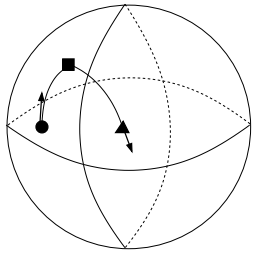

Fig. 1. Illustration of GLRSQ. The closest correct (circle) and incorrect (triangle) prototypes are, respectively, dragged toward and pushed away from the instance (square) along the geodesic curve, rather than the straight line between them.

$$E(\mathbf{W}) = \sum_{i=1}^{m} \Phi(\mu(\mathbf{X}_i, \mathbf{W})) \tag{7}$$

$$\mu(\mathbf{X}_i, \mathbf{W}) = \frac{\delta(\mathbf{X}_i, \mathbf{W}_J) - \delta(\mathbf{X}_i, \mathbf{W}_K)}{\delta(\mathbf{X}_i, \mathbf{W}_J) + \delta(\mathbf{X}_i, \mathbf{W}_K)} \tag{8}$$

where $\mathbf{W}_J \in \mathcal{M}$ and $\mathbf{W}_K \in \mathcal{M}$ are the closest correct and incorrect prototypes to the training point $\mathbf{X}_i$ in terms of δ, respectively. Again, the updating rule can be obtained by minimizing the abovementioned cost function through stochastic deepest gradient descent algorithm on the manifold $\mathcal{M}$.

Given an example $\mathbf{X}_i$, let $\gamma_J(t)$ be a geodesic curve connecting $\mathbf{X}_i$ with its closest correct prototype $\mathbf{W}_J$, starting in $\mathbf{W}_J$ with initial speed $\mathbf{V}_J \in \mathcal{T}_{\mathbf{W}_J}\mathcal{M}$. Analogously, let $\gamma_K(t)$ be a geodesic curve connecting $\mathbf{X}_i$ with its closest incorrect prototype $\mathbf{W}_K$, starting in $\mathbf{W}_K$ with initial speed $\mathbf{V}_K \in \mathcal{T}_{\mathbf{W}_K}\mathcal{M}$. The cost function for this example along the two curves reads

$$E(\gamma_J(t), \gamma_K(t)) = \Phi\left(\frac{\delta(\mathbf{X}_i, \gamma_J(t)) - \delta(\mathbf{X}_i, \gamma_K(t))}{\delta(\mathbf{X}_i, \gamma_J(t)) + \delta(\mathbf{X}_i, \gamma_K(t))}\right).$$

Let δ_J and δ_K denote the distances $\delta(\mathbf{X}_i, \mathbf{W}_J)$ and $\delta(\mathbf{X}_i, \mathbf{W}_K)$, respectively. Following (6), the Riemannian gradient $\nabla_{\mathbf{V}_J} E$ can be computed as:

$$\begin{aligned}\langle \mathbf{V}_J, \nabla_{\mathbf{V}_J} E \rangle_{\mathbf{W}_J} &= \frac{\partial E(\gamma_J(t), \gamma_K(t))}{\partial \gamma_J(t)}\bigg|_{t=0} \\ &= \Phi' \frac{2\delta_K}{(\delta_J + \delta_K)^2} \frac{d}{dt} \delta(\mathbf{X}_i, \gamma_J(t))\bigg|_{t=0}. \end{aligned} \tag{9}$$

Similarly, the Riemannian gradient $\nabla_{\mathbf{W}_K} E$ can be computed via

$$\begin{aligned}\langle \mathbf{V}_K, \nabla_{\mathbf{V}_K} E \rangle_{\mathbf{W}_K} &= \frac{\partial E(\gamma_J(t), \gamma_K(t))}{\partial \gamma_K(t)}\bigg|_{t=0} \\ &= \Phi' \frac{-2\delta_J}{(\delta_J + \delta_K)^2} \frac{d}{dt} \delta(\mathbf{X}_i, \gamma_K(t))\}\bigg|_{t=0}. \end{aligned} \tag{10}$$

Given a particular Riemannian manifold with a specifically defined Riemannian structure, the geodesics and time derivatives of the geodesic distance $(d\delta/d)t$ will be uniquely determined.

Once we have the Riemannian gradients $\nabla_{\mathbf{V}_J} E$ and $\nabla_{\mathbf{V}_K} E$, the prototypes $\mathbf{W}_J$ and $\mathbf{W}_K$ are updated following [18]:

$$\mathbf{W}_J^{\text{new}} = \mathrm{Exp}_{\mathbf{W}_J}(-\alpha(t) \nabla_{\mathbf{V}_J} E) \tag{11}$$

$$\mathbf{W}_K^{\text{new}} = \mathrm{Exp}_{\mathbf{W}_K}(-\alpha(t) \nabla_{\mathbf{V}_K} E) \tag{12}$$

where $\mathrm{Exp}_{\mathbf{W}}$ is the exponential map at $\mathbf{W}$, determined by the Riemannian metric associated with a specific Riemannian manifold [19], and $0 < \alpha(t) < 1$ is the learning rate that satisfies $\sum_{t=1}^{\infty} \alpha(t) = \infty$ and $\sum_{t=1}^{\infty} \alpha^2(t) < \infty$.

IV. Generalized Learning Riemannian Space Quantization on SPD Riemannian Manifold

In this section, we present the approach of extending the GLVQ to the specific Riemannian manifold consisting of SPD matrices. Before describing our method, we briefly introduce some basic concepts of this particular manifold and the calculation of Riemannian mean of SPD matrices that have been also introduced in [20]–[22].

A. Manifold of SPD Matrices

Let $\mathbb{S}(n)$ represent the space of all $n \times n$ symmetric matrices and $\mathbb{S}^+(n)$ denote the space of all $n \times n$ SPD matrices. The training points $\mathbf{X}_i \in \mathbb{S}^+(n), i = 1, \ldots, m$ are $n \times n$ SPD matrices. The $\mathbb{S}^+(n)$ space forms a curved manifold, known as Riemannian symmetric space [20] if each tangent space of $\mathbb{S}^+(n)$ is equipped with the Riemannian natural metric.

Let $\mathbf{X} \in \mathbb{S}^+(n)$, and the tangent space at point $\mathbf{X}$ is a space of symmetric matrices, given by $\mathcal{T}_{\mathbf{X}}\mathbb{S}^+(n) = \mathbb{S}(n) = \{\mathbf{V} \in \mathbb{R}^{n\times n} | \mathbf{V}^T = \mathbf{V}\}$. The Riemannian natural metric on the manifold of SPD matrices is defined by the local inner product; for any $\mathbf{V}_1, \mathbf{V}_2 \in \mathcal{T}_{\mathbf{X}}\mathbb{S}^+(n)$

$$\langle \mathbf{V}_1, \mathbf{V}_2 \rangle_{\mathbf{X}} = \mathrm{Tr}(\mathbf{V_1}\mathbf{X}^{-1}\mathbf{V_2}\mathbf{X}^{-1}) \tag{13}$$

where Tr represents the trace of the matrix. The inner product induces a norm for the tangent vectors on the tangent space, i.e., $\|\mathbf{V}\|_{\mathbf{X}}^2 = \langle \mathbf{V}, \mathbf{V} \rangle_{\mathbf{X}}$. Note that in identity matrix $\mathbf{I}$, this norm is reduced into Frobenius norm, that is

$$\langle \mathbf{V}, \mathbf{V} \rangle_{\mathbf{I}} = \|\mathbf{V}\|_F^2.$$

The geodesic curve at the point $\mathbf{X}$ in the direction of $\mathbf{V} \in \mathcal{T}_{\mathbf{X}}\mathbb{S}^+(n)$ can be expressed as

$$\gamma(t) = \mathbf{X}^{1/2} \exp(t\mathbf{X}^{-1/2}\mathbf{V}\mathbf{X}^{-1/2})\mathbf{X}^{1/2} \tag{14}$$

where exp is the exponential of matrix (see Appendix A). Obviously, this geodesic curve is entirely contained in the manifold. For any given pair $\mathbf{X}_1, \mathbf{X}_2 \in \mathbb{S}^+(n)$, we can find a geodesic curve $\gamma(t)$ connecting $\gamma(0) = \mathbf{X}_1$ with $\gamma(1) = \mathbf{X}_2$ by taking the initial velocity [22]

$$\dot{\gamma}(0) = \mathbf{X}_1^{1/2} \log(\mathbf{X}_1^{-1/2}\mathbf{X}_2\mathbf{X}_1^{-1/2})\mathbf{X}_1^{1/2} \in T_{\mathbf{X}_1}\mathbb{S}^+(n)$$

where log is the principal logarithm of matrix (see Appendix A). According to (5), the squared Riemannian geodesic distance between $\mathbf{X}_1$ and $\mathbf{X}_2$ can be explicitly computed as follows [21]:

$$\delta(\mathbf{X}_1, \mathbf{X}_2) = \|\log(\mathbf{X}_1^{-1}\mathbf{X}_2)\|_F^2 = \sum_{i=1}^{n} \log^2 \lambda_i \tag{15}$$

where $\lambda_i, i = 1, \ldots, n$ are the real eigenvalues of $\mathbf{X}_1^{-1}\mathbf{X}_2$. Note that the definition of the squared Riemannian distance is equivalent to $\delta(\mathbf{X}_1, \mathbf{X}_2) = \|\log(\mathbf{X}_1^{-1/2}\mathbf{X}_2\mathbf{X}_1^{-1/2})\|_F^2$ (see Appendix B). The main properties of the Riemannian geodesic distance are listed as follows.

1) $\delta(\mathbf{X}_1, \mathbf{X}_2) = \delta(\mathbf{X}_2, \mathbf{X}_1)$.
2) $\delta(\mathbf{X}_1^{-1}, \mathbf{X}_2^{-1}) = \delta(\mathbf{X}_1, \mathbf{X}_2)$.
3) $\delta(\mathbf{W}^T\mathbf{X}_1\mathbf{W}, \mathbf{W}^T\mathbf{X}_2\mathbf{W}) = \delta(\mathbf{X}_1, \mathbf{X}_2) \ \forall \ \mathbf{W} \in Gl(n)$.

Here, $Gl(n)$ represents the general linear group, consisting of all nonsingular real matrices. The third property implies that the Riemannian geodesic distance between two SPD matrices is invariant by projection, making classifiers based on this distance measure robust to transformations.

In the Riemannian manifold of SPD matrices, given a point $\mathbf{X} \in \mathbb{S}^+(n)$, it is possible for each point in this space $\mathbf{X}_i \in \mathbb{S}^+(n)$ to identify a tangent vector $\mathbf{V}_i \in \mathbb{S}(n)$ such that $\mathbf{V}_i = \dot{\gamma}(0)$ is the initial speed vector of the geodesic $\gamma(t)$ between $\mathbf{X}$ and $\mathbf{X}_i$. The Riemannian Log map operator $\mathrm{Log}_{\mathbf{X}}$: $\mathbb{S}^+(n) \rightarrow \mathbb{S}(n)$ achieves the mapping from the manifold to the tangent space at $\mathbf{X}$, i.e., $\mathrm{Log}_{\mathbf{X}}(\mathbf{X}_i) = \mathbf{V}_i$. The Riemannian Exp map operator $\mathrm{Exp}_{\mathbf{X}}(\mathbf{V}_i) = \mathbf{X}_i$ allows to go back in the original manifold of SPD matrices $\mathbb{S}^+(n)$ in a one-to-one mapping. The Riemannian Exp and Log maps associated with the Riemannian natural metric given by (13) are expressed as follows:

$$\mathrm{Exp}_{\mathbf{X}}(\mathbf{V}_i) = \mathbf{X}^{1/2} \exp(\mathbf{X}^{-1/2}\mathbf{V}_i\mathbf{X}^{-1/2})\mathbf{X}^{1/2} \tag{16}$$

$$\mathrm{Log}_{\mathbf{X}}(\mathbf{X}_i) = \mathbf{X}^{1/2} \log(\mathbf{X}^{-1/2}\mathbf{X}_i\mathbf{X}^{-1/2})\mathbf{X}^{1/2}. \tag{17}$$

B. Riemannian Mean of SPD Matrices

Consider m SPD matrices $\mathbf{X}_1, \ldots, \mathbf{X}_m \in \mathbb{S}^+(n)$, the Riemannian mean of them is defined as follows [20], [21]:

$$\mathfrak{m} = \arg\min_{\mathbf{X}\in\mathbb{S}^+(n)} \sum_{i=1}^{m} \delta(\mathbf{X}, \mathbf{X}_i). \tag{18}$$

There is no general closed-form expression for finding $\mathfrak{m}$. Fletcher and Joshi [20] introduced a gradient descent algorithm to compute the mean. The solution of the minimization problem (18) has been proven to exist and be unique, justifying the validity of the definition [20]. The space $\mathbb{S}^+(n)$ does have nonpositive sectional curvature, and for a manifold with nonpositive sectional curvature, the mean is uniquely defined; thus, the Riemannian mean is uniquely defined. In particular, the gradient for finding the mean reads [21]

$$\nabla\rho = \mathbf{X}\sum_{k=1}^{m} \log\left(\mathbf{X}_k^{-1}\mathbf{X}\right) = -\sum_{i=1}^{m} \mathrm{Log}_{\mathbf{X}}(\mathbf{X}_k) \tag{19}$$

and the calculation of the Riemannian mean is listed in Algorithm 1.

C. Generalized Learning Riemannian Space Quantization of SPD Matrices

Consider a training data set $\{(\mathbf{X}_i, y_i)\}_{i=1}^m$, $\mathbf{X}_i \in \mathbb{S}^+(n)$, $y_i \in \{1, \ldots, C\}$. Assume that the corresponding GLVQ classifier consists of M ($M \geq C$) prototypes $\mathbf{W}_i \in \mathbb{S}^+(n)$ labeled by $c_i \in \{1, \ldots, C\}$. Then, the GLVQ algorithm for points on the Riemannian manifold of SPD matrices can be derived by substituting the Riemannian distance defined by (15) into the cost function $E(\mathbf{W})$ given by (7) and (8).

Algorithm 1 Riemannian Mean

Input: Points $\mathbf{X}_1, \ldots, \mathbf{X}_m \in \mathbb{S}^+(n)$, and a very small positive real number ϵ

Output: The intrinsic mean $\mathfrak{m} \in \mathbb{S}^+(n)$

Initialize $\mathfrak{m}_0 = I$
repeat
  $V_i = \frac{1}{m} \sum_{k=1}^{m} \mathrm{Log}_{\mathfrak{m}_i}(\mathbf{X}_k)$
  $\mathfrak{m}_{i+1} = \mathrm{Exp}_{\mathfrak{m}_i}(\mathbf{V}_i)$
until $\| \mathbf{V}_i \|_F < \epsilon$

To obtain the update rule of the prototypes, we need to compute the Riemannian gradient of the cost function defined on the manifold of SPD matrices. Recall that, for a given instance $\mathbf{X}_i \in \mathbb{S}^+(n)$, we denote its closest correct prototype [under the Riemannian geodesic distance defined by (15)] as $\mathbf{W}_J \in \mathbb{S}^+(n)$ and its closest incorrect prototype as $\mathbf{W}_K \in \mathbb{S}^+(n)$. The two geodesic curves on $\mathbb{S}^+(n)$ emitting from $\mathbf{W}_J$ and $\mathbf{W}_K$ toward $\mathbf{X}_i$ with the initial speed $\mathbf{V}_J = \dot{\gamma}(0)$ and $\mathbf{V}_K = \dot{\gamma}(0)$ are specified by

$$\gamma_J(t) = \mathbf{W}_J^{1/2} \exp\left(t \mathbf{W}_J^{-1/2} \mathbf{V}_J \mathbf{W}_J^{-1/2}\right) \mathbf{W}_J^{1/2}$$

and

$$\gamma_K(t) = \mathbf{W}_K^{1/2} \exp\left(t \mathbf{W}_K^{-1/2} \mathbf{V}_K \mathbf{W}_K^{-1/2}\right) \mathbf{W}_K^{1/2}$$

respectively. Substituting the abovementioned two curves into (9) and (10), we can obtain the Riemannian gradients of the cost function with respect to $\mathbf{W}_J$ and $\mathbf{W}_K$ as follows:

$$\nabla_{\mathbf{W}_J} E = -\Phi' \frac{4\delta_K}{(\delta_J + \delta_K)^2} \mathrm{Log}_{\mathbf{W}_J}(\mathbf{X}_i) \tag{20}$$

$$\nabla_{\mathbf{W}_K} E = \Phi' \frac{4\delta_J}{(\delta_J + \delta_K)^2} \mathrm{Log}_{\mathbf{W}_K}(\mathbf{X}_i) \tag{21}$$

where δ_J and δ_K denote the distances $\delta(\mathbf{X}_i, \mathbf{W}_J)$ and $\delta(\mathbf{X}_i, \mathbf{W}_K)$, respectively. The detailed calculation is given in Appendix C. Once we have the gradients, we can update the prototypes via (11) and (12) with $\mathrm{Exp}_{\mathbf{W}}$ defined by (16). The algorithm of GLVQ in the Riemannian SPD matrix space is summarized by Algorithm 2. Following the general practice of the LVQ algorithm that the class prototypes are initialized with the mean of examples from each class plus some random fluctuation, we initialize the prototypes of each class with the Riemannian mean of examples from that class plus small random perturbation.

V. Experimental Results

The performance of our proposed approach is evaluated on both synthetic data sets and real-world motor imagery data sets. For all LVQ-based algorithms, the logistic scaling function $\Phi(x) = 1/(1 + e^{-x})$ was used, and the learning rates are continuously reduced in the course of learning. We use the schedule $\alpha(t) = (n/100) \cdot 0.01^{t/T}$,, where n is the dimension of the manifold, T denotes the number of sweeps through the training data, and $t = 1, \ldots, T$.

Algorithm 2 GLRSQ of SPD Matrices

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

Output: M labeled prototypes $(\mathbf{W}_1, c_1), \ldots, (\mathbf{W}_M, c_M)$, where $\boldsymbol{W}_i \in \mathbb{S}^+(n)$ and $c_i \in \{1, \ldots, C\}$

Initialize $\mathbf{W}_i$ by the Riemannian mean of examples labeled by c_i plus small random perturbation.
while a stopping criterion is not reached **do**
  Randomly select a training example $(\mathbf{X}_i, y_i)$
  Identify the closest correct prototype $\mathbf{W}_J$ and the closest incorrect prototype $\mathbf{W}_K$ utilizing Riemannian geodesic distances
  $\mathbf{V}_J = \alpha(t)\ \Phi' \frac{4\delta_K}{(\delta_J + \delta_K)^2} \mathrm{Log}_{\mathbf{W}_J}(\mathbf{X}_i)$
  $\mathbf{V}_K = -\alpha(t)\ \Phi' \frac{4\delta_J}{(\delta_J + \delta_K)^2} \mathrm{Log}_{\mathbf{W}_K}(\mathbf{X}_i)$
  $\mathbf{W}_J \leftarrow \mathrm{Exp}_{\mathbf{W}_J}(\mathbf{V}_J)$
  $\mathbf{W}_K \leftarrow \mathrm{Exp}_{\mathbf{W}_K}(\mathbf{V}_K)$
end while

A. Baseline Methods

The baseline methods naively utilize the Euclidean distance between matrices to measure the distance between the instance and the prototype, i.e., $\delta_F(\mathbf{X}, \mathbf{W}) = \|\mathbf{X} - \mathbf{W}\|_F^2$, where $\|\cdot\|_F^2$ is the Frobenius norm of a matrix and $\|\mathbf{A}\|_F^2 = \mathrm{Tr}(\mathbf{A}\mathbf{A}^T) = \sum_{i,j} a_{ij}^2$. Replacing the distance in the cost function given by (7) and (8) with δ_F and as the derivative of Frobenius norm

$$\frac{\partial \| \mathbf{W} \|_F^2}{\partial \mathbf{W}} = 2\mathbf{W}$$

we can obtain the updating rule of prototypes as follows [23]:

$$\Delta \mathbf{W}_J = \alpha(t) \Phi' \frac{4\delta_K}{(\delta_J + \delta_K)^2} (\mathbf{X}_i - \mathbf{W}_J) \tag{22}$$

$$\Delta \mathbf{W}_K = -\alpha(t) \Phi' \frac{4\delta_J}{(\delta_J + \delta_K)^2} (\mathbf{X}_i - \mathbf{W}_K). \tag{23}$$

Comparing the learning rules for prototypes in (22) and (23) with those in (3) and (4), we find that (unsurprisingly) GLVQ learning rule for SPD matrices using Euclidean distance recovers the structure of the original GLVQ learning rule for vectors.

By closely examining the GLVQ learning rule for SPD matrices in Euclidean space [i.e., (22) and (23)], we find it is equivalent to vectorize the upper triangle of the SPD matrices and apply the standard GLVQ learning rule for vectors since $\mathbf{X}_i - \mathbf{W}_K$ obtains the element difference between matrix $\mathbf{X}_i$ and $\mathbf{W}_K$. Following this idea, we can also vectorize the upper triangle of the SPD matrices and apply standard GRLVQ [10] and GMLVQ [11]. However, there is one problem. Even though we can initialize the prototypes with SPD matrices, the updated prototypes are not guaranteed to be positive definite. Therefore, for each update of the prototype, we reshape the vectorized prototype into a symmetric matrix and check whether the matrix is positive definite. If the matrix leaves out of SPD space, we project the prototype into its closest SPD matrix by eliminating the negative eigenvalues.

Besides the abovementioned LVQ baseline methods, we also compare our method with the minimal distance to the Riemannian mean method, which learns a cluster center for each

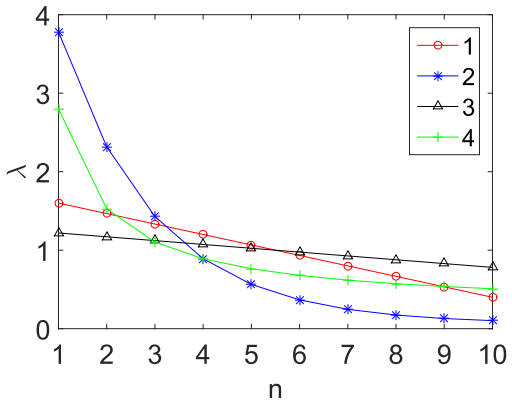

Fig. 2. Plot of the four sets of eigenvalues.

class of instances [8] on synthetic data sets. With respect to motor imagery data sets, besides the abovementioned four methods, we also compare the performance of our method to the state-of-the-art methods in the literature.

B. Synthetic Data

We first generated synthetic data set to verify our proposed approach. The instances are generating according to the following equations:

$$\mathbf{X} = \sum_{i}^{n} \lambda_i \boldsymbol{u}_i \boldsymbol{u}_i^T \tag{24}$$

where λ_i represents the ith eigenvalue of $\mathbf{X}$ and $\boldsymbol{u}_i$ is the corresponding eigenvectors. Here, we choose $n = 10$.

We designed four sets of eigenvalues. The first set of eigenvalues is from a linearly decreasing function

$$\tilde{\xi}^1(t) = 13 - t, \quad t = 1, \ldots, n.$$

The second set of eigenvalues follows an exponentially decreasing function:

$$\tilde{\xi}^2(t) = 1 + 100 \exp(-0.5\ t), \quad t = 1, \ldots, n.$$

The third set of eigenvalues also follows from a linearly decreasing function but with different slope:

$$\tilde{\xi}^3(t) = 13 - 0.5t, \quad t = 1, \ldots, n.$$

The fourth set of eigenvalues follows from a reciprocal function:

$$\tilde{\xi}^4(t) = \frac{1}{t}, \quad t = 1, \ldots, n.$$

For all four sets of eigenvalues, the mean of the eigenvalues is normalized to 1, that is

$$\xi^i(t) \ = n\tilde{\xi}^i(t) \ \Big/ \sum_{q=1}^{n} \tilde{\xi}^i(q)$$

where $i = 1, \ldots, 4$. The four sets of eigenvalues are plotted in Fig. 2.

We also designed four sets of eigenvectors (basis). To that end, we generated four $n \times n$ random real matrices [each element generated i.i.d. from $\mathcal{N}(0, 1)$]. Then, for each matrix, the Gram–Schmidt orthogonalization was used to obtain the orthogonal basis—the set of eigenvectors. We denote the four sets of eigenvectors by $\{\boldsymbol{v}_1^i, \ldots, \boldsymbol{v}_n^i\}, i = 1, 2, 3, 4$.

TABLE I

DESCRIPTIONS OF SYNTHETIC DATA SETS. l AND k DENOTE THE NUMBER OF SETS OF EIGENVALUES AND EIGENVECTORS THAT ARE USED TO GENERATE THE DATA, RESPECTIVELY. C DENOTES THE NUMBER OF CLASSES, n DENOTES THE RANK OF THE SPD MATRIX, #TRAIN REPRESENTS THE NUMBER OF TRAINING INSTANCES, AND #VALIDATION DENOTES THE NUMBER OF VALIDATION INSTANCES, WHILE #TEST IS THE NUMBER OF TEST INSTANCES

Name	l	k	C	n	#Train	#Validation	#Test
SynI	2	2	4	10	$250 * 4$	$250 * 4$	$250 * 4$
SynII	4	1	4	10	$250 * 4$	$250 * 4$	$250 * 4$

Two synthetic data sets were generated, named SynI and SynII. For SynI, the first two sets of eigenvalues and the first two eigenvectors are combined to produce four classes of instances. The first two classes share the first set of eigenvalues but each with different sets of eigenvectors. The remaining two classes share the second set of eigenvalues and also each with different sets of eigenvectors. When instances were generated, random noise were injected in both its eigenvalues and eigenvectors. The eigenvalues $\lambda(t), t = 1, \ldots, n$ of the instance were created according to uniform distribution $U(\xi^1(t) - \epsilon, \xi^1(t) + \epsilon)$ or $U(\xi^2(t) - \epsilon, \xi^2(t) + \epsilon)$, depending on its class label. Here, we chose $\epsilon = 0.1$. The eigenvectors $\boldsymbol{u}_t, t = 1, \ldots, n$ of the instance were the orthonormalized version of $\boldsymbol{v}_t^1 + \boldsymbol{\epsilon}$ or $\boldsymbol{v}_t^2 + \boldsymbol{\epsilon}$ through the Gram–Schmidt orthogonalization depending on its label, where $\boldsymbol{\epsilon}$ follows $\mathcal{N}(0, \sigma^2\mathbf{I})$. Here, we chose $\sigma = 0.3$. Once the eigenvalues and eigenvectors of the instance were obtained, the instance can be acquired by (24).

The SynII is also of four classes and was generated using the four sets of eigenvalues and one set of eigenvectors. Instances of each class have their own set of eigenvalues but share the common eigenvectors. Each instance was created following the same procedure of SynI.

The synthetic data sets are summarized in Table I. For both the SynI and synII, a training set, a validation set, and a test set were generated independently. All three sets contained 250 instances per class. The generating process of each data set was repeated 30 times, and the following results are the average results over 30 runs.

The number of prototypes per class and training epochs was selected from $\{1, 2, 3\}$ and $\{20, 50, 100\}$, respectively, based on the validation set performance. The test set classification performance (averaged accuracy over 30 runs with standard deviation) of different LVQ approaches on synthetic data sets SynI and synII is presented in Tables II and III, respectively.[1] In all experiments, GLRSQ performs significantly better than the other three LVQ approaches and the MDRM method with $p < 0.05$ via the nonparametric Wilcoxon signed-rank test [24].

[1]The selected number of prototypes per class and training epochs are listed in Table VIII.

TABLE II

COMPARISON OF ACCURACY AMONG DIFFERENT LVQ APPROACHES ON DATA SET SYNI. AVERAGED ACCURACY OVER 30 RUNS ALONG WITH STANDARD DEVIATION IS GIVEN

m/C	GLVQ	GRLVQ	GMLVQ	MDRM	GLRSQ
50	0.9028 ± 0.0108	0.9035 ± 0.0108	0.9137 ± 0.0140	0.9671 ± 0.0063	0.9681 ± 0.0072
150	0.9246 ± 0.0070	0.9377 ± 0.0066	0.9320 ± 0.0072	0.9721 ± 0.0073	0.9738 ± 0.0062
250	0.9306 ± 0.0088	0.9490 ± 0.0069	0.9361 ± 0.0078	0.9730 ± 0.0059	0.9750 ± 0.0054
Mean	0.9193	0.9301	0.9258	0.9684	0.9723

TABLE III

PERFORMANCE COMPARISON AMONG DIFFERENT LVQ METHODS ON DATA SET SYNII. AVERAGED ACCURACY OVER 30 RUNS ALONG WITH STANDARD DEVIATION IS GIVEN

m/C	GLVQ	GRLVQ	GMLVQ	MDRM	GLRSQ
50	0.7105 ± 0.0234	0.7625 ± 0.0140	0.7838 ± 0.0142	0.8717 ± 0.0120	0.8759 ± 0.0122
150	0.7497 ± 0.0189	0.8167 ± 0.0110	0.8121 ± 0.0160	0.8974 ± 0.0081	0.9037 ± 0.0075
250	0.7621 ± 0.0187	0.8340 ± 0.0130	0.8191 ± 0.0155	0.9040 ± 0.0083	0.9137 ± 0.0095
Mean	0.7408	0.8044	0.8050	0.8910	0.8978

C. Motor Imagery Data Sets

Two popular multiclass motor imagery EEG data sets were used to evaluate our proposed approach, namely, BCI competition III data set IIIa [25], [26] and BCI competition IV data set 2a [27].The two data sets (BCI III IIIa and BCI IV 2a) contain EEG signals extracted using $n = 60$ and $n = 22$ electrodes, respectively. The EEG signals will be transformed into spatial covariance matrices (the hypothesis is that for the given tasks, the spatial covariance matrix provides discriminative information of brain states). Suppose that the ith trial of preprocessed EEG signal is given as follows[2]:

$$\boldsymbol{E}_i = [\boldsymbol{e}(t_i), \ldots, \boldsymbol{e}(t_i + l - 1)] \in \mathbb{R}^{n \times l} \tag{25}$$

where n and l denote the number of channels and sampled points, respectively. Each trial of EEG signal is represented by the sample covariance matrix computed as follows:

$$\mathbf{X}_i = \frac{1}{l-1}\mathbf{E}_i\mathbf{E}_i^T. \tag{26}$$

Thus, elements of the two data sets live in $P(60)$ and $P(22)$, respectively.

1) Description of Data Sets:

a) BCI III IIIa: BCI competition III data set IIIa consists of EEG recordings from three subjects. Each subject performed four different types of motor imagery tasks, including left hand, right hand, foot, or tongue movements according to a cue; 60 channels were recorded with a 64-channel EEG amplifier from Neuroscan, using the left mastoid for reference and the right mastoid as ground. The EEG was sampled with 250 Hz. The subject sat in a relaxing chair with armrests. After the presentation of a cue, the subject was asked to perform the indicated imagery task for 4 s. The experiment consists of several runs (at least 6) with 40 trials each. Each of the four cues was displayed ten times within each run in a randomized order. The time interval of the processed data was restricted to the time segment comprised between 0.5 and 2.5 s starting from the cue instructing the user to perform the mental task. EEG signals from each trial were bandpass filtered by a fifth-order Butterworth filter in the 10-30-Hz frequency band to analyze the μ and β rhythms. The preprocessed segmented EEG signals were then used to obtain the sample covariance matrices by (26).

[2]After preprocessing, $\boldsymbol{X}_i$ becomes zero mean.

TABLE IV

COMPARISON OF FOUR DIFFERENT LVQ METHODS IN TERMS OF CLASSIFICATION ACCURACY ON DATA SET IIIA; 30-FOLD CROSS VALIDATION WAS USED. MEAN ACCURACY (± STANDARD DEVIATION) IS GIVEN

Subject	GLVQ	GRLVQ	GMLVQ	GLRSQ
K3	0.7333 ± 0.1374	0.3389 ± 0.0846	0.5861 ± 0.1340	0.9139 ± 0.0941
K6	0.5000 ± 0.1353	0.4152 ± 0.1613	0.4041 ± 0.1563	0.6417 ± 0.1633
L1	0.5167 ± 0.1790	0.5250 ± 0.1408	0.4750 ± 0.1369	0.7917 ± 0.1285
Mean	0.5833	0.4255	0.4884	0.7824

TABLE V

COMPARISON BETWEEN OUR METHOD AND OTHER METHODS IN THE LITERATURE ON BCI COMPETITION III DATA SET IIIA IN TERMS OF KAPPA VALUE

	mean kappa	K3	K6	L1
1^{st}	**0.7926**	0.8222	**0.7556**	**0.8000**
2^{nd}	0.6872	0.9037	0.4333	0.7111
GLRSQ	0.6765	0.8519	0.4778	0.7000
3^{rd}	0.6272	**0.9481**	0.4111	0.5222
MDRM	0.6222	0.8222	0.3556	0.6889
GLVQ	0.4481	0.5778	0.3444	0.4222
GMLVQ	0.3716	0.4815	0.1889	0.4444
GRLVQ	0.2654	0.1407	0.2444	0.4111

TABLE VI

COMPARISON OF FOUR DIFFERENT LVQ METHODS IN TERMS OF CLASSIFICATION ACCURACY ON BCI IV 2A DATA SET; 30-FOLD CROSS VALIDATION WAS USED

Subject	GLVQ	GRLVQ	GMLVQ	GLRSQ
S1	0.6518 ± 0.1031	0.6315 ± 0.0868	0.5893 ± 0.0809	0.8508 ± 0.0764
S2	0.3753 ± 0.1256	0.2961 ± 0.1066	0.3001 ± 0.0933	0.555 ± 0.0852
S3	0.6772 ± 0.0993	0.4186 ± 0.0823	0.5090 ± 0.1112	0.8955 ± 0.0591
S4	0.4855 ± 0.1189	0.3550 ± 0.0853	0.4596 ± 0.1098	0.6632 ± 0.1011
S5	0.3180 ± 0.1018	0.3078 ± 0.1132	0.3083 ± 0.1002	0.5013 ± 0.1191
S6	0.4701 ± 0.1000	0.3933 ± 0.1096	0.395 ± 0.1089	0.5965 ± 0.0924
S7	0.5951 ± 0.0925	0.5268 ± 0.1043	0.5327 ± 0.1247	0.8483 ± 0.0909
S8	0.6818 ± 0.1219	0.4956 ± 0.1165	0.6002 ± 0.1212	0.8722 ± 0.059
S9	0.6894 ± 0.1009	0.4754 ± 0.0912	0.6879 ± 0.0977	0.8508 ± 0.0766
Mean	0.5494	0.4430	0.4869	0.7371

b) BCI IV 2a: BCI competition IV data set 2a consists of EEG signals from nine healthy subjects who were performing four different motor imagery tasks, i.e., the imagination of the movement of the left hand, right hand, both feet, and tongue. The signals were recorded by placing 22 electrodes distributed over the sensorimotor area of the subject at a sampling rate of 250 Hz. For each subject, two sessions were recorded on two different days, each containing 288 trials with 72 trials per class. At each trial, a cue was given in the form of an arrow pointing either to the left, right, down, or up, corresponding to one of the four classes, to prompt the subject to perform the corresponding motor imagery task. The motor imagination lasted 4 s from the presence of cue until the end of the motor imagery task. The time interval of the processed data was restricted to the time segment comprised between 0.5 and 2.5 s starting from the cue instructing the user to perform the mental task. As before, EEG signals from each trial were bandpass filtered by a fifth-order Butterworth filter in the 10–30-Hz frequency band.

2) Results: For each LVQ method, the number of prototypes per class and training epochs were selected from $\{1, 2\}$ and $\{20, 50, 100\}$, respectively, using fivefold cross validation on the training fold.

a) Results of data set BCI III IIIa: We first compared our GLRSQ with the other three LVQ methods using 30-fold

TABLE VII

PERFORMANCE COMPARISON BETWEEN OUR METHOD AND THE STATE-OF-THE-ART METHODS IN TERMS OF KAPPA VALUE ON BCI COMPETITION IV DATA SET 2A

	mean kappa	S1	S2	S3	S4	S5	S6	S7	S8	S9
TSSM+LDA [28]	**0.593**	0.77	0.33	**0.77**	*0.51*	0.35	0.36	*0.71*	0.72	**0.83**
GLRSQ	*0.586*	**0.79**	0.32	*0.76*	**0.55**	0.34	*0.36*	0.66	0.70	*0.79*
WaSF ConvNet [29]	0.58	0.63	0.32	0.75	0.44	**0.60**	**0.38**	0.69	0.71	0.73
1^{st}	0.57	0.68	**0.42**	0.75	0.48	*0.40*	0.27	**0.77**	**0.75**	0.61
TSLDA [8]	0.57	0.74	0.38	0.72	0.50	0.26	0.34	0.69	0.71	0.76
MDRM [8]	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
GLVQ	0.35	0.50	0.16	0.50	0.31	0.13	0.25	0.33	0.51	0.44
3^{rd}	0.31	0.38	0.18	0.48	0.33	0.07	0.14	0. 29	0. 49	0.44
GMLVQ	0.31	0.40	0.15	0.36	0.29	0.06	0.20	0.33	0.50	0.55
GRLVQ	0.24	0.40	0.08	0.28	0.16	0.06	0.15	0.31	0.42	0.34

cross validation. The results reported are averaged results over 30 runs. The mean classification accuracy together with standard deviation is given in Table IV. From Table IV, we can see that the proposed Riemannian GLVQ outperformed the three alternatives significantly ($p < 0.03$ using the nonparametric Wilcoxon signed-rank test). Interestingly, GLVQ performed better than both GRLVQ and GMLVQ. One possible reason is that the training data size is rather small. GLVQ is much simpler and has much fewer learning parameters than the other two approaches. We also empirically observed that the updated prototypes of GLVQ are less likely to leave out of the SPD space.

We then compared the performance of our method with the results published on the BCI competition III website and the MDRM algorithm [8]. The method was trained using the training set and tested on the separated test set. The selected number of prototypes per class and training epochs are given in Table IX. Kappa values are used to measure the performance of the methods. The results are given by Table V. From Table V, we can see that the performance of our proposed GLRSQ is quite close to the second-best performance and significantly better than the third-best method. The first two best results were delivered by methods both related to CSP, suggesting the validity of using spatial covariance matrices. However, EEG signals in this data set consist of 60 channels, resulting in very high dimensional spatial covariance matrices, i.e., 60×60, compared with the number of training instances. This might be one reason that our GLRSQ failed to beat the methods related to CSP.

b) BCI IV 2a: In this same way, we first compared our GLRSQ with the other three LVQ methods using 30-fold cross validation on this data set. The results reported are averaged results over 30 runs. The mean classification accuracy together with standard deviation is given in Table VI. From Table VI, we can see that the GLRSQ methods significantly outperformed the other three LVQ methods ($p < 2.1 \times 10^{-5}$ using the nonparametric Wilcoxon signed-rank test). Again, we found that GLVQ obtained better results than GRLVQ and GMLVQ.

We further compared the performance of our GLRSQ with three winning published on BCI competition IV and the results of state-of-the-art methods in the literature. Table VII gives the results in terms of kappa values as it was done for the BCI completion IV. The selected number of prototypes per class and training epochs of our methods are given in Table IX. From Table VII, we can see that our GLRSQ obtained much better performance than the first winner of BCI competition IV. Our GLRSQ method beat the first winner on five subjects over the nine subjects. Our GLRSQ is marginally better than the second winner of BCI competition IV. We also compared our method with the TSLDA method [8], the MDRM method [8], and two recently published methods: one method is termed TSSM + LDA that uses tangent space of submanifold (TSSM) learning for dimension reduction and linear discriminant analysis (LDA) as final classifier [28], and the other method is named WaSF ConvNet that utilizes a deep convolution network (ConvNet) with wavelet and spatial filter (WaSF) kernels to learn joint space–time–frequency features for motor imagery classification [29]. From Table VII, we can see that our GLRSQ method significantly beats the MDRM method and also outperformed the TSLDA method. Our GLRSQ method shows superior performance to the deep WaSF ConvNet method too. One possible reason is that the size of training data (subject-specific training instances) in BCI is rather small compared to that in computer vision. Consequently, the deep ConvNet-based methods failed to obtain unbeatable performance in BCI, such as in computer vision. Our methods obtained comparable performance to the TSSM + LDA method, as this method takes advantages of submanifold learning, suggesting that the dimension reduction of the covariance matrices may lead to improved classification performance. Our future work will incorporate the Riemannian manifold dimension reduction in our Riemannian LVQ framework.

VI. CONCLUSION

We have proposed a new approach to classify data points living in the curved Riemannian manifolds within the LVQ framework and specifies the approach for data points that live in the Riemannian manifold of SPD matrices. This approach extends the existing GLVQ method in the Euclidean space to the Riemannian space.

The standard GLVQ can be directly applied to classifying the SPD matrices by reshaping the upper triangle of the SPD matrices into vectors. The prototype can be kept positive definite by the proper initialization with the SPD matrix and the projection into its closest SPD matrix. By the same adaptation, GRLVQ and GMLVQ can also be applied to classify the SPD matrices. However, through treating the SPD matrix

space as the Riemannian manifold and using the Riemannian geodesic distance instead of the Frobenius norm, the prototype is guaranteed to be an SPD matrix without any heuristic manipulation. We also find that GLVQ with the Riemannian geodesic distance delivers better performance than GRLVQ and GMLVQ in classifying SPD matrices, suggesting that the Riemannian geodesic distance is more appropriate than the Mahalanobis distance in the case of SPD matrix classification.

Our proposed GLRSQ has inherited the intuitive and simple nature of LVQ methods. Furthermore, it can naturally deal with the multiclass classification of SPD matrices. More importantly, the approach is an online learning algorithm that can perform life-long learning, and the test is very fast, as it only needs to compare the test instance to several prototypes. Empirical experiments have been conducted on EEG data classification to verify the good performance of our method. Our proposed method provides an alternative efficient and effective EEG decoding method for online BCI.

However, the training sample in the BCI research is usually small, while the number of electrodes can be high,[3] resulting in high-dimensional and possibly ill-conditioned SPD matrices. However, not all the recorded electrodes may provide equally useful information [30]. Hence, in our future work, we will consider learning of lower dimensional SPD manifold relevant for the given classification task along the lines of [31]. This can be expressed as an optimization problem on a Grassmann manifold [31], [32].

Appendix A
Matrix Exponential and Logarithms

The exponential of a matrix $\mathbf{X}$ is given by the convergent series

$$\exp(\mathbf{X}) = \sum_{k=0}^{\infty} \frac{1}{k!}\mathbf{X}^k. \tag{27}$$

For symmetric matrix $\mathbf{X} \in \mathbb{S}(n)$, the exponential of the matrix $\mathbf{X}$ can be computed via eigenvalue decomposition of $\mathbf{X}$

$$\mathbf{X} = \mathbf{U}\ \mathrm{diag}(\lambda_1, \ldots, \lambda_n)\ \mathbf{U}^T \tag{28}$$

where $\lambda_1, \ldots, \lambda_n$ are eigenvalues of $\mathbf{X}$, $\mathbf{U}$ is the matrix of eigenvectors of $\mathbf{X}$, and diag denotes the diagonal matrix with the arguments of diag being the diagonal elements of the matrix. Note that as $\mathbf{X}$ is symmetric, $\mathbf{U}^{-1} = \mathbf{U}^T$. Then, the exponential of the matrix of $\mathbf{X}$ is

$$\exp(\mathbf{X}) = \mathbf{U}\ \mathrm{diag}(\exp(\lambda_1), \ldots, \exp(\lambda_n))\ \mathbf{U}^T. \tag{29}$$

Logarithm of a matrix $\mathbf{P}$ is the solution of the matrix equation $\exp(\mathbf{X}) = \mathbf{P}$. When $\mathbf{P}$ does not have eigenvalues in the closed negative real line, there exists a unique real logarithm, called the principal logarithm, denoted by $\log(\mathbf{P})$, defined by the convergent series

$$\log(\mathbf{P}) = -\sum_{k=1}^{\infty} \frac{(\mathbf{I}-\mathbf{P})^k}{k}. \tag{30}$$

[3]See http://bnci-horizon-2020.eu/database/datasets

We have the following properties.

1) If $\mathbf{A}$ and $\mathbf{B}$ are both positive-definite matrices, then

$$\mathrm{Tr}(\log(\mathbf{AB})) = \mathrm{Tr}(\log(\mathbf{A})) + \mathrm{Tr}(\log(\mathbf{B})). \tag{31}$$

2) If positive-definite matrices $\mathbf{A}$ and $\mathbf{B}$ commute, i.e., $\mathbf{AB} = \mathbf{BA}$, then

$$\log(\mathbf{AB}) = \log(\mathbf{A}) + \log(\mathbf{B}). \tag{32}$$

3) Substituting $\mathbf{B} = \mathbf{A}^{-1}$ in (32), we have

$$\log(\mathbf{A}^{-1}) = -\log(\mathbf{A}). \tag{33}$$

For SPD matrix $\mathbf{P}$, the logarithm can be computed by

$$\log(\mathbf{P}) = \mathbf{U}\ \mathrm{diag}(\log(\lambda_1), \ldots, \log(\lambda_n))\ \mathbf{U}^T$$

where $\lambda_1, \ldots, \lambda_n$ are eigenvalues of $\mathbf{P}$, and $\mathbf{U}$ is the matrix of eigenvectors of $\mathbf{P}$. The exponential of any symmetric matrix is a positive-definite symmetric matrix, and the inverse of exponential (i.e., the principal logarithm) of any positive-definite symmetric matrix is a symmetric matrix, i.e., $\forall\ \mathbf{P} \in \mathbb{S}^+(n)$, $\log(\mathbf{P}) \in \mathbb{S}(n)$ and $\forall\ \mathbf{S} \in \mathbb{S}(n)$, $\exp(\mathbf{S}) \in \mathbb{S}^+(n)$.

Note that for all square matrices $\mathbf{A}$ and $\mathbf{B}$ and all scalars c, we have the following properties with respect to Tr operator:

$$\mathrm{Tr}(\mathbf{A}+\mathbf{B}) = \mathrm{Tr}(\mathbf{A}) + \mathrm{Tr}(\mathbf{B}) \tag{34}$$

$$\mathrm{Tr}(c \cdot \mathbf{A}) = c \cdot \mathrm{Tr}(\mathbf{A}). \tag{35}$$

Appendix B
Proof of Equivalent Definition of Riemannian Geodesic Distance

According to the property of trace operator given by (34), we have

$$\begin{aligned}\delta(\mathbf{X}_1, \mathbf{X}_2) &= \left\| \log\left(\mathbf{X}_1^{-1/2}\mathbf{X}_2\mathbf{X}_1^{-1/2}\right)\right\|_F^2 \\ &= \mathrm{Tr}\Big[\left(\log\left(\mathbf{X}_1^{-1/2}\mathbf{X}_2\mathbf{X}_1^{-1/2}\right)\right)^2\Big] \\ &= 2\mathrm{Tr}\Big[\log\left(\mathbf{X}_1^{-1/2}\mathbf{X}_2\mathbf{X}_1^{-1/2}\right)\Big].\end{aligned}$$

As the matrix $\mathbf{X}_1$ is SPD, $\mathbf{X}_1^{-1/2}$ is also SPD. Since $\mathbf{X}_2$ is also SPD, according to the properties given by (31) and (33), we have

$$\begin{aligned}&\mathrm{Tr}\Big[\log\left(\mathbf{P}_1^{-1/2}\mathbf{P}_2\mathbf{P}_1^{-1/2}\right)\Big] \\ &\quad= -\frac{1}{2}\mathrm{Tr}[\log(\mathbf{P}_1)] + \mathrm{Tr}[\log(\mathbf{P}_2)] - \frac{1}{2}\mathrm{Tr}[\log(\mathbf{P}_1)] \\ &\quad= -\mathrm{Tr}[\log(\mathbf{P}_1)] + \mathrm{Tr}(\log(\mathbf{P}_2)] \\ &\quad= \mathrm{Tr}\Big[\log(\mathbf{P}_1^{-1}\mathbf{P}_2)\Big]\end{aligned}$$

consequently, we have

$$\delta(\mathbf{X}_1, \mathbf{X}_2) = \left\| \log\left(\mathbf{X}_1^{-1/2}\mathbf{X}_2\mathbf{X}_1^{-1/2}\right)\right\|_F^2 = \left\| \log\left(\mathbf{X}_1^{-1}\mathbf{X}_2\right)\right\|_F^2.$$

TABLE VIII
SELECTED NUMBER OF PROTOTYPES PER CLASS AND TRAINING EPOCHS ON SYNTHETIC DATA SETS

	GLVQ		GRLVQ		GMLVQ		GLRSQ	
Data set	# prototype	#epochs	# prototype	#epochs	# prototype	#epochs	# prototype	#epochs
SynI (50)	2.40 ± 0.77	23.00 ± 9.15	2.70 ± 0.47	76.33 ± 28.46	2.53 ± 0.63	54.00 ± 33.18	2.33 ± 0.71	51.67 ± 30.18
SynI (150)	2.13 ± 0.82	30.67 ± 18.74	2.83 ± 0.38	75.67 ± 27.00	2.73 ± 0.45	67.67 ± 36.55	2.80 ± 0.41	57.67 ± 38.66
SynI (250)	2.20 ± 0.76	32.00 ± 14.95	2.77 ± 0.43	86.67 ± 22.49	2.40 ± 0.62	57.67 ± 32.87	2.57 ± 0.63	64.00 ± 31.91
SynII (50)	2.60 ± 0.50	87.33 ± 23.92	2.73 ± 0.52	62.33 ± 31.26	2.57 ± 0.68	38.33 ± 30.18	2.60 ± 0.50	96.67 ± 12.69
SynII (150)	2.40 ± 0.77	85.67 ± 24.73	2.90 ± 0.31	50.67 ± 30.73	2.50 ± 0.51	35.00 ± 25.43	2.90 ± 0.31	81.67 ± 24.51
SynII (250)	2.27 ± 0.74	88.33 ± 21.51	2.90 ± 0.31	55.67 ± 27.50	2.70 ± 0.60	31.67 ± 18.95	2.90 ± 0.31	81.67 ± 24.51

TABLE IX
SELECTED NUMBER OF PROTOTYPES PER CLASS AND TRAINING EPOCHS ON MOTOR IMAGERY CLASSIFICATION DATA SETS

	GLVQ		GRLVQ		GMLVQ		GLRSQ	
Data set	# prototype	#epochs	# prototype	#epochs	# prototype	#epochs	# prototype	#epochs
K3 (IIIa)	1	100	2	100	1	50	2	100
K6 (IIIa)	2	50	1	50	2	50	1	50
L1 (IIIa)	1	100	1	100	2	50	1	100
S1 (IV2a)	1	100	2	100	2	50	1	20
S2 (IV2a)	1	100	1	20	2	100	1	20
S3 (IV2a)	1	100	1	20	2	50	1	100
S4 (IV2a)	1	100	2	50	2	100	2	50
S5 (IV2a)	2	100	1	100	2	20	2	20
S6 (IV2a)	1	100	1	100	2	50	2	20
S7 (IV2a)	1	100	1	100	2	20	1	50
S8 (IV2a)	1	100	2	20	2	20	1	20
S9 (IV2a)	1	100	2	100	1	20	1	100

APPENDIX C
CALCULATION OF RIEMANNIAN GRADIENT

The derivative of the Riemannian distance $\delta(\mathbf{X}_i, \gamma_J(t)$ with respect to t is as follows [20], [21]:

$$\begin{aligned}\frac{d}{dt}\delta(\mathbf{X}_i, \gamma_J(t))\Big|_{t=0} &= \frac{d}{dt}\| \log\left(\mathbf{X}_i^{-1/2}\gamma_J(t)\ \mathbf{X}_i^{-1/2}\right)\|_F^2\Big|_{t=0} \\ &= \frac{d}{dt}\mathrm{Tr}\Big[\left(\log\left(\mathbf{X}_i^{-1/2}\gamma_J(t)\ \mathbf{X}_i^{-1/2}\right)\right)^2\Big]\Big|_{t=0} \\ &= 2\mathrm{Tr}\Big[\mathbf{V}_J \log\left(\mathbf{X}_i^{-1}\mathbf{W}_J\right)\mathbf{W}_J^{-1}\Big] \\ &= -2\mathrm{Tr}\Big[\mathbf{V}_J \log\left(\mathbf{W}_J^{-1}\mathbf{X}_i\right)\mathbf{W}_J^{-1}\Big]. \end{aligned} \tag{36}$$

Using the definition of the Riemannian inner product given by (13), (36) can be written as the inner product as follows:

$$\frac{d}{dt}\delta(\mathbf{X}_i, \gamma_J(t))\Big|_{t=0} = \left\langle \mathbf{V}_J, -2\mathbf{W}_J \log\left(\mathbf{W}_J^{-1}\mathbf{X}_i\right)\right\rangle_{\mathbf{W}_J}.$$

As $\log(A^{-1}BA) = A^{-1}(\log B)A$, for all $A \in Gl(n)$ and for all $\mathbb{S}^+(n)$, the above equation can be rewritten as follows:

$$\begin{aligned}&\frac{d}{dt}\delta(\mathbf{X}_i, \gamma_J(t))\Big|_{t=0} \\ &\quad= \left\langle \mathbf{V}_J, -2\mathbf{W}_J \log\left(\mathbf{W}_J^{-1/2}\mathbf{W}_J^{-1/2}\mathbf{X}_i\mathbf{W}_J^{-1/2}\mathbf{W}_J^{1/2}\right)\right\rangle_{\mathbf{W}_J} \\ &\quad= \left\langle \mathbf{V}_J, -2\mathbf{W}_J^{1/2} \log\left(\mathbf{W}_J^{-1/2}\mathbf{X}_i\right)\mathbf{W}_J^{-1/2}\right)\mathbf{W}_J^{1/2}\rangle_{\mathbf{W}_J} \\ &\quad= \langle \mathbf{V}_J, -2\mathrm{Log}_{\mathbf{W}_J}(\mathbf{X}_i)\rangle_{\mathbf{W}_J}. \end{aligned} \tag{37}$$

Similarly, we can obtain the derivative of the Riemannian distance $\delta(\mathbf{X}_i, \gamma_K(t))$ with respect to t

$$\begin{aligned}\frac{d}{dt}\delta(\mathbf{X}_i, \gamma_K(t))\Big|_{t=0} &= -2\mathrm{Tr}\Big[\mathbf{V}_K \log\left(\mathbf{W}_K^{-1}\mathbf{X}_i\right)\mathbf{W}_K^{-1}\Big] \\ &= \left\langle \mathbf{V}_K, -2\mathbf{W}_K \log\left(\mathbf{W}_K^{-1}\mathbf{X}_i\right)\right\rangle_{\mathbf{W}_K} \\ &= \langle \mathbf{V}_K, -2\mathrm{Log}_{\mathbf{W}_K}(\mathbf{X}_i)\rangle_{\mathbf{W}_K}. \end{aligned} \tag{38}$$

Substituting (37) and (38) into (9) and (10), respectively, we can obtain

$$\begin{aligned}\langle \mathbf{V}_J, \nabla_{\mathbf{W}_J} E\rangle_{\mathbf{W}_J} &= \Phi' \frac{2\delta_K}{(\delta_J+\delta_K)^2}\langle \mathbf{V}_J, -2\mathrm{Log}_{\mathbf{W}_J}(\mathbf{X}_i)\rangle_{\mathbf{W}_J} \\ &= \left\langle \mathbf{V}_J, \Phi' \frac{-4\delta_K}{(\delta_J+\delta_K)^2}\mathrm{Log}_{\mathbf{W}_J}(\mathbf{X}_i)\right\rangle_{\mathbf{W}_J} \end{aligned} \tag{39}$$

and

$$\begin{aligned}\langle \mathbf{V}_K, \nabla_{\mathbf{W}_K} E\rangle_{\mathbf{W}_K} &= \frac{-2\delta_J}{(\delta_J+\delta_K)^2}\langle \mathbf{V}_J, -2\mathrm{Log}_{\mathbf{W}_K}(\mathbf{X}_i)\rangle_{\mathbf{W}_K} \\ &= \left\langle \mathbf{V}_K, \Phi' \frac{4\delta_J}{(\delta_J+\delta_K)^2}\mathrm{Log}_{\mathbf{W}_K}(\mathbf{X}_i)\right\rangle_{\mathbf{W}_K} \end{aligned} \tag{40}$$

respectively. Consequently, we can obtain the Riemannian gradients of the cost function with respect to $\mathbf{W}_J$ and $\mathbf{W}_K$ as follows:

$$\nabla_{\mathbf{W}_J} E = -\Phi' \frac{4\delta_K}{(\delta_J+\delta_K)^2}\mathrm{Log}_{\mathbf{W}_J}(\mathbf{X}_i) \tag{41}$$

$$\nabla_{\mathbf{W}_K} E = \Phi' \frac{4\delta_J}{(\delta_J+\delta_K)^2}\mathrm{Log}_{\mathbf{W}_K}(\mathbf{X}_i). \tag{42}$$

APPENDIX D
OPTIMAL PARAMETERS

The optimal parameters used in the synthetic experiments are listed in Table VIII, while those used in the motor imagery classification are given in Table IX.

REFERENCES

[1] T. Kohonen, "Learning vector quantization," in *The Handbook of Brain Theory and Neural Networks*. Cambridge, MA, USA: MIT Press, 1995, pp. 537–540.

[2] D. Nova and P. A. Estévez, "A review of learning vector quantization classifiers," *Neural Comput. Appl.*, vol. 25, nos. 3–4, pp. 511–524, Sep. 2014.

[3] A. Barachant, S. Bonnet, M. Congedo, and C. Jutten, "Riemannian geometry applied to BCI classification," in *Latent Variable Analysis and Signal Separation*, V. Vigneron, V. Zarzoso, E. Moreau, R. Gribonval, and E. Vincent, Eds. Berlin, Germany: Springer, 2010, pp. 629–636.

[4] B. Blankertz, R. Tomioka, S. Lemm, M. Kawanabe, and K.-R. Müller, "Optimizing spatial filters for robust EEG single-trial analysis," *IEEE Signal Process. Mag.*, vol. 25, no. 1, pp. 41–56, Dec. 2008.

[5] 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.

[6] 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, Jul. 2017.

[7] P. Gaur, R. B. Pachori, H. Wang, and G. Prasad, "A multi-class EEG-based BCI classification using multivariate empirical mode decomposition based filtering and Riemannian geometry," *Expert Syst. Appl.*, vol. 95, pp. 201–211, Apr. 2018.

[8] 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, Oct. 2012.

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

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

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

[12] S. Saralajew and T. Villmann, "Adaptive tangent distances in generalized learning vector quantization for transformation and distortion invariant classification learning," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2016, pp. 2672–2679.

[13] S. Saralajew, D. Nebel, and T. Villmann, "Adaptive Hausdorff distances and tangent distance adaptation for transformation invariant classification learning," in *Proc. Int. Conf. Neural Inf. Process.*, 2016, pp. 362–371.

[14] T. Villmann *et al.*, "Searching for the origins of life–detecting RNA life signatures using learning vector quantization," in *Proc. Int. Workshop Self-Organizing Maps*. Cham, Switzerland: Springer, 2019, pp. 324–333.

[15] M. Kirby and C. Peterson, "Visualizing data sets on the Grassmannian using self-organizing mappings," in *Proc. 12th Int. Workshop Self-Organizing Maps Learn. Vector Quantization, Clustering Data Vis. (WSOM)*, Jun. 2017, pp. 1–6.

[16] M. Spivak, *A Comprehensive Introduction to Differential Geometry*, vol. 1, 3rd ed. Berkeley, CA, USA: Publish or Perish, 1999.

[17] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*. Princeton, NJ, USA: Princeton Univ. Press, 2007.

[18] S. Bonnabel, "Stochastic gradient descent on Riemannian manifolds," *IEEE Trans. Autom. Control*, vol. 58, no. 9, pp. 2217–2229, Sep. 2013.

[19] 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.

[20] P. T. Fletcher and S. Joshi, "Principal geodesic analysis on symmetric spaces: Statistics of diffusion tensors," in *Computer Vision and Mathematical Methods in Medical and Biomedical Image Analysis*, M. Sonka, I. A. Kakadiaris, and J. Kybic, Eds. Berlin, Germany: Springer, 2004, pp. 87–98.

[21] M. Moakher, "A differential geometric approach to the geometric mean of symmetric positive-definite matrices," *SIAM J. Matrix Anal. Appl.*, vol. 26, no. 3, pp. 735–747, Jan. 2008.

[22] X. Duan, H. Sun, and X. Zhao, "Riemannian gradient algorithm for the numerical solution of linear matrix equations," *J. Appl. Math.*, vol. 2014, no. 2, pp. 1–7, 2014.

[23] K. B. Petersen and M. S. Pedersen, *The Matrix Cookbook*. Lyngby, Denmark: Technical Univ. of Denmark, 2008, p. 14.

[24] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bull.*, vol. 1, no. 6, pp. 80–83, Dec. 1945.

[25] B. Blankertz *et al.*, "The BCI competition 2003: Progress and perspectives in detection and discrimination of EEG single trials," *IEEE Trans. Biomed. Eng.*, vol. 51, no. 6, pp. 1044–1051, Jun. 2004.

[26] B. Blankertz *et al.*, "The BCI competition III: Validating alternative approaches to actual BCI problems," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 14, no. 2, pp. 153–159, Jun. 2006.

[27] M. Tangermann *et al.*, "Review of the BCI competition IV," *Frontiers Neurosci.*, vol. 6, no. 55, pp. 1–31, 2012.

[28] 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.

[29] 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.

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

[31] M. T. Harandi, M. Salzmann, and R. Hartley, "From manifold to manifold: Geometry-aware dimensionality reduction for SPD matrices," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 17–32.

[32] P.-A. Absil, R. Mahony, and R. Sepulchre, "Riemannian geometry of Grassmann manifolds with a view on algorithmic computation," *Acta Appl. Math.*, vol. 80, no. 2, pp. 199–220, Jan. 2004.

Fengzhen Tang is currently 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.

Mengling Fan is currently pursuing the Ph.D. degree with the State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences, University of Chinese Academy of Sciences, Beijing, China. His research focuses on machine learning and neural computation.

Peter Tiňo was a Research Fellow with the Neural Computing Research Group, Aston University, Birmingham, U.K., and the Austrian Research Institute for AI, Vienna, Austria. He is currently a Professor and the Chair in Complex and Adaptive Systems with the School of Computer Science, University of Birmingham, Birmingham. His research interests include artificial intelligence, machine learning, statistical pattern recognition, and natural computation.