IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 29, 2020, accepted March 22, 2020, date of publication April 1, 2020, date of current version April 14, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2984941

SPD Data Dictionary Learning Based on Kernel Learning and Riemannian Metric

RIXIN ZHUANG, ZHENGMING MA, WEIJIA FENG, AND YUANPING LIN
School of Electronics and Information Technology, Sun Yat-sen University, Guangzhou 510006, China

Corresponding author: Zhengming Ma (issmzm@mail.sysu.edu.cn)

This work was supported in part by the Natural Science Foundation of China through the Project Research on Nonlinear Alignment Algorithm of Local Coordinates in Manifold Learning under Grant 61773022.

ABSTRACT The use of regional covariance descriptors to generate feature data represented by Symmetric Positive Definite (SPD) matrices from images or videos has become increasingly common in machine learning. However, SPD data itself does not constitute a vector space, and dictionary learning involves a large number of linear operations, so dictionary learning cannot be performed directly on SPD data. For this reason, a more common method is to map the SPD data to the Reproducing Kernel Hilbert Space (RKHS). The so-called kernel learning is to find the most suitable RKHS for specific tasks. RKHS can be uniquely generated by a kernel function. Therefore, RKHS learning can also be considered as kernel learning. In this article, there are two main contributions. The first contribution is to propose a framework which based on Kernel Learning and Riemannian Metric (KLRM). Usually the learnable kernel function framework is to learn some parameters in the kernel function. The second contribution is dictionary learning by applying KLRM to SPD data. The SPD data is transformed into the RKHS generated by KLRM, and RKHS after training provides the most suitable working space for dictionary learning. Under the proposed framework, we design a positive definite kernel function, which is defined by the Log-Euclidean metric. This function can be transformed into a corresponding Riemannian kernel. The experimental results provided in this paper is compared with other state-of-the-art algorithms for SPD data dictionary learning and show that the proposed algorithm achieves better results.

INDEX TERMS Dictionary learning, symmetric positive definite matrix, reproducing Kernel Hilbert space, Log-Euclidean metric.

I. INTRODUCTION

Sparse representation is a very popular research direction in signal processing and computer vision problems. For a given set of data points $X = \{x_1, \cdots, x_N\}$, the main idea of sparse representation is to learn the dictionary set $D = \{d_1, \cdots, d_L\}$, so that $x_i (i = 1, \cdots, N)$ can be effectively represented by the basis (dictionary) through a small number of linear combinations of non-zero coefficients while minimizing the reconstruction error [1]. Most of the existing sparse coding algorithms are based on vector and linear operations in Euclidean space, that is, data points and dictionaries are elements in vector space. Sparse representation has been widely used in many machine learning related problems, such as face recognition [2], [3] and picture classification [4].

The associate editor coordinating the review of this manuscript and approving it for publication was Sotirios Goudos.

However, in many practical scenarios, the data points are usually on a Riemannian Manifold, such as SPD matrix space [5], [6], Stiefel manifolds and Grassmann manifolds [7], [8]. Existing dictionary learning and sparse coding algorithms based on vector space do not consider the inherent non-linear geometry of Riemannian manifolds, so they cannot be directly applied to the processing of Riemannian manifold data. The vector method simply by vectorizing the SPD matrix will cause severe encoding distortion [9]. Mehrtash Harandi et al proposed a model for converting high-dimensional SPD manifold to low-dimensional SPD manifold [10]. In a supervised case, the model consists of the affinity function and the similarity between two SPD matrices. In the unsupervised case, the model consists only of the similarity between two SPD matrices. In both cases, the transformation matrix requires orthogonality. The model is trained to obtain the optimal transformation matrix W, and

finally to obtain the optimal low-dimensional SPD manifold. Yaxin Peng et al proposed a semi-supervised metric learning framework, and used Hinge loss function and smooth loss function in combination with triplet constraints to construct two different models to obtain the optimal local metric [11]. In view of the poor performance of the SPD matrix data directly treated as Euclidean space's data, the SPD matrix is usually given a Riemannian geometry when analyzed, so that the SPD matrix can be regarded as an SPD manifold or a tensor manifold point. But this will also bring some difficulties. Like the nonlinearity of the Riemannian geometry makes it difficult to model sparse coding methods based on linear combinations. Therefore, it is necessary to extend the sparse coding and dictionary learning methods based on SPD manifold.

In summary, there are mainly three ideas to solve the nonlinear problem of SPD manifold. The first is to directly model the SPD matrix manifold, that is, an SPD matrix is expressed as a linear combination of the SPD matrices in the dictionary set. The result of linear combination may not be the SPD matrix. In addition, the optimization problem based on the SPD manifold is difficult to solve. The second is to map the SPD matrix manifold data to the tangent space of an SPD matrix. The tangent space of the SPD manifold is a linear space. The third is to map the SPD matrix manifold data to RKHS. Huang et al proposed a novel metric learning method that can work directly on the logarithm of the SPD matrix. Specifically, the method mainly learns a tangent mapping, which can directly transform the logarithm of a matrix (rather than a vector) from the original tangent space to a more distinguishable new tangent space [12]. Modeling SPD manifold data onto RKHS has been a hot research direction in recent years [13], [14]. However, mapping SPD manifold to RKHS not only requires that the kernel functions based on SPD manifold meet Mercer's theory [15], but also requires that the kernel functions can retain the original geometry of the manifold while mapping, so it will requires the high performance of the kernel functions. The main effects of SPD manifold are Log-Euclidean kernel function [16] and Gaussian kernel function based on Bregman divergence [17]. However, the Log-Euclidean kernel function cannot reflect the internal geometry of the SPD manifold, and the Bregman divergence is only an approximation of the geodesic distance of the SPD manifold. Therefore, the existing two kernel functions have their own defects. So as to reflect the geometric structure of the manifold better and the geometric relationship information of the existing manifold data can be added to the kernel function, that is the method of geometric perception kernel learning.

This paper researches dictionary learning and sparse coding methods on SPD manifold, maps SPD manifold data to RKHS for processing, and innovates sparse coding methods on SPD manifold from kernel functions and dictionary learning. The main work and innovations of the paper are as follows:

1) A data-dependent kernel function learning method is proposed. The data dependency function adds a data dependency to the basic kernel function, and uses the data dependency to provide more information about the original data, so that the new kernel function can better reflect the geometric information of the original data. The parameters of the data dependencies in the data dependence kernel function can be obtained by combining learning with specific goals.
2) A dictionary learning and sparse coding method on SPD manifold based on data-dependent kernel learning is proposed. Apply this kernel learning to the Riemannian manifold dictionary learning and sparse coding method based on kernel method, optimize the three variables of kernel parameters, dictionary learning and sparse coding at the same time, learn the optimal kernel parameters and dictionary, and use the learned kernel parameters and dictionary so you can get better encoding results.
3) Experimental analysis. The KLRM-DL proposed in this paper were tested on three data sets, and compared with RSR, TSC, K-LE-DLSC and other algorithms. The validity of the algorithms was verified by comparing the classification accuracy of each algorithm.

The following content of the paper is arranged as follows. In Chapter 2, we will introduce the basic mathematical knowledge involved in this paper, including sparse coding, dictionary learning, Reproducing Kernel Hilbert Space and Riemannian manifold. In Chapter 3, important research results obtained by other researchers in the field of kernel learning will be introduced. In Chapter 4, we will describe in detail the kernel algorithms designed in this paper, including data-dependent kernel learning frameworks and dictionary learning and sparse coding methods on SPD manifold based on data-dependent kernel learning. In Chapter 5, we will solve each model proposed in Chapter 4, and give the algorithm block diagram of this paper. Then in Chapter 6, the experimental part of the thesis combines multiple mainstream data sets and compares with multiple mainstream algorithms. Finally, we will summarize the work of our thesis in Chapter 7.

II. NOTATIONS AND PRELIMINARIES

A. NOTATIONS

In this article, we use the lowercase letter x as a datapoint, the bold capital letter X as a set of the datapoint. The transpose, trace, inverse of the matrix are expressed as X^T, $tr(X)$ and X^{-1}. Sym^n_{++} denotes SPD manifold.

B. REPRODUCING KERNEL HILBERT SPACE (RKHS)

RKHS is a common mathematical platform for various kernel methods in machine learning. One can map data from the original data space to RKHS, and then complete various machine learning tasks on RKHS. The original data space can

be the familiar Euclidean space, or manifolds, such as SPD manifold, Grassmannian Manifolds, and so on.

When an inner product is defined on a linear space, the linear space can become an inner product space. Complete inner product space is a Hilbert space. RKHS is a special Hilbert space.

Definition 1: Let $H = (L^2(\Omega), \langle \bullet, \bullet \rangle)$ be a Hilbert space of functions and $k : \Omega \times \Omega \to \mathbb{R}$, if

1) For all $x \in \Omega$, $k_x(\bullet) = k(\bullet, x) \in H$;
2) For all $x \in \Omega$ and all $f \in H$,

$$f(x) = \langle f, k_x \rangle = \langle f(\bullet), k(\bullet, x) \rangle = \int_\Omega f(z)k(x, z)\, dz \tag{1}$$

then H is called RKHS and $k(x, y)$ is called the reproducing kernel of H.

Remark 1: $L^2(\Omega)$ and $\langle \bullet, \bullet \rangle$ are defined as follows:

$$\begin{aligned} L^2(\Omega) &= \{f | f : \Omega \to \mathbb{R}, \int_\Omega |f(x)|^2\, dx < +\infty\}, \\ \langle f, g \rangle &= \int_\Omega f(x)g(x)\, dx \end{aligned} \tag{2}$$

In practice, Ω is the so-called data space. Using the reproducing kernel $k(x, y)$, a transformation from the data space Ω to RKHS H can be defined as follows: $\phi : \Omega \to H$, for all $x \in \Omega$

$$\phi(x) = k(\bullet, x) \in H \tag{3}$$

According to the reproducing property of k, it can be easily proven that

$$\langle \phi(x), \phi(y) \rangle = k(x, y) \tag{4}$$

The above formula has been widely used in many machine learning algorithms based on RKHS.

In mathematics, it can be proved that an RKHS can be uniquely generated by using a kernel function such that the kernel function is exactly the reproducing kernel of the generated RKHS. It must be pointed out that kernel functions are different from reproducing kernels. Reproducing kernels are defined based on Hilbert space, while kernel functions are defined independently.

Definition 2: Let $k : \Omega \times \Omega \to \mathbb{R}$, if

1) For all $x, y \in \Omega$, $k(x, y) = k(y, x)$
2) For all integer N and all $x_1, \cdots, x_N \in \Omega$, the following matrix is positive definite:

$$\mathrm{K} = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_N) \\ \vdots & \ddots & \vdots \\ k(x_N, x_1) & \cdots & k(x_N, x_N) \end{bmatrix}$$

then the function k is called kernel function.

The process of generating an RKHS from a kernel function k is as follows:

1) Generate a linear space by using the kernel function:

$$\begin{aligned} H_k &= span\{k(\bullet, x) | x \in \Omega\} \\ &= \left\{ \sum_{i=1}^{N} \alpha_i k(\bullet, x_i) | x_i \in \Omega, \alpha_i \in \mathbb{R}, i \in \mathbb{Z}^+ \right\} \end{aligned} \tag{5}$$

2) Define an inner product on H_k : for all $f, g \in H_k$, since $f(\bullet) = \sum_{i=1}^{N} \alpha_i k(\bullet, x_i)$, $g(\bullet) = \sum_{j=1}^{M} \beta_j k(\bullet, y_j)$, then

$$\langle f, g \rangle = \begin{bmatrix} \alpha_1 & \cdots & \alpha_N \end{bmatrix} \begin{bmatrix} k(x_1, y_1) & \cdots & k(x_1, y_M) \\ \vdots & \ddots & \vdots \\ k(x_N, y_1) & \cdots & k(x_N, y_M) \end{bmatrix} \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_M \end{bmatrix} \tag{6}$$

3) Complete H_k to obtain the complete space of H_k, denoted as $\widetilde{H}_k$, then $\widetilde{H}_k$ is an RKHS and the kernel function k is exactly the reproducing kernel of $\widetilde{H}_k$.

Since an RKHS can be uniquely generated from a kernel function, the task of learning an RKHS can be translated into a task of learning a kernel function. This is what the algorithm proposed in this paper does.

C. SPARSE CODING

The so-called dictionaries finds a set of basic data $\{d_1, \cdots, d_L\}$, other data x can be approximated by a linear combination of this set of basic data:

$$\min_a \left\| x - \sum_{i=1}^{L} a_i d_i \right\|^2 + \lambda \mathrm{Sparse}(a) \tag{7}$$

where $a = [a_1, \cdots, a_L]$ and $\mathrm{Sparse}(a)$ is called the sparse regular term of dictionary encoding, which means that while the best linear approximation, the non-zero components of the approximation coefficient a_i are minimized We usually use 1-norm to represent the sparse regular term of sparse coding [17]:

$$\mathrm{Sparse}(a) = \|a\|_1 = \sum_{i=1}^{L} |a_i| \tag{8}$$

It is worth mentioning that $\mathrm{Sparse}(a)$ is the feature of sparse coding. Without $\mathrm{Sparse}(a)$ the problem (7) would become the problem of subspace projection. If $\{d_1, \cdots, d_L\}$ are orthonormal, then

$$\min_a \left\| x - \sum_{i=1}^{L} a_i d_i \right\|^2 \Rightarrow a_i = x^T d_i, \quad i = 1, \cdots, L \tag{9}$$

D. DICTIONARY LEARNING

The premise of sparse coding is to have a suitable dictionary. This dictionary is learned from samples. According to the specific application, collecting a certain number of fully representative samples and then use them for dictionary learning.

Let $X = \{x_1, \cdots, x_N\}$ be the learning sample to find the suitable dictionaries $\{d_1, \cdots, d_L\}$. The objective function of dictionary learning is:

$$\min_{d_1,\cdots,d_L} \sum_{i=1}^{N} \left[\left\| x_i - \sum_{j=1}^{L} a_{ij} d_j \right\|^2 + \lambda \text{Sparse}(A_{iRow}) \right] \tag{10}$$

where $A = \begin{bmatrix} a_{11} & \cdots & a_{1L} \\ \vdots & \ddots & \vdots \\ a_{N1} & \cdots & a_{NL} \end{bmatrix}$ is the sparse coding matrix. In the above objective function, the sparse coding matrix A is a by-product which is not needed, dictionary $\{d_1, \cdots, d_L\}$ is needed.

E. RIEMANNIAN MANIFOLD

In recent years, as a kind of data space for non-Euclidean data, Riemannian manifold has been more and more applied to machine learning. Riemannian manifold evolves from topological space.

Definition 3: Let M be a Hausdorff topological space with a countable topological basis. If any given $x \in M$, there exists a neighborhood U_x of x, U_x is homeomorphic to an open set $O_x \subset \mathbb{R}^n$ of the n-dimension Euclidean space $\mathbb{R}^n$, then M is called a n-dimensional topological manifold.

Remark 2: (U_x, ϕ_x) is called a local coordinate of M, where ϕ_x is the homeomorphic mapping between U_x and O_x. $\{(U_x, \phi_x) | x \in M\}$ is called locate coordinate system of M.

Definition 4: Let M be a topology manifold, if for all local coordinates (U, ϕ) and (V, δ) with $U \cup V \neq \varnothing$, if the mapping $\delta \circ \phi^{-1} : \phi(U \cup V) \to \delta(U \cup V)$ is infinitely differentiable, then M is called differential manifold.

Definition 5: Let M be a differential manifold, w be a second-order tensor field on M, i.e., for all $p \in M$, $w(p)$ is a second-order tensor on the tangent space of p, if w is symmetric, positive definite and smooth, then w is called a Riemannian metric and (M, w) is a Riemannian manifold.

Based on Riemannian metric, we can define geodesic distance on Riemannian manifold.

Definition 6: Let $p \in M, q \in M, \rho : [0, 1] \to M$, satisfy $\rho(0) = p, \rho(1) = q$ and ρ is smooth, then ρ represents a smooth curve from p to q on M.

Furthermore, for all $t \in (0, 1), \rho(t) \in M$, and $\rho'(t)$ is a tangent vector of $\rho(t)$ such that for all smooth function f defined on a neighborhood of $\rho(t)$,

$$\rho'(t)(f) = \frac{d(f \circ \rho)(\theta)}{d\theta}\Big|_{\theta=t} \tag{11}$$

Thus, the length of the curve ρ is defined as:

$$L(p) = \int_0^1 \sqrt{w(\rho(t))(\rho'(t), \rho'(t))} dt \tag{12}$$

Theorem 1: Let C_{pq} represents the set of all smooth curves between p and q, and

$$d(p, q) = min\{L(\rho) | \rho \in C_{pq}\} \tag{13}$$

then $d(p, q)$ satisfies the three axioms of distance and (M, d) becomes a distance space.

Remark 3: The distance $d(p, q)$ is often called geodesic distance. Although geodesic distances are defined from Riemannian metric, however, what we use in practical applications is geodesic distances, not Riemannian metric.

A Riemannian manifold often used in many machine learning applications. The commonly-used geodesic distances on Sym^n_{++} are as follows:

1) For all $x, y \in Sym^n_{++}$, the so-called Log-Euclidean distance is as follows:

$$\delta_{G1}(x, y) = \|log(x) - log(y)\|_F \tag{14}$$

where $\|A\|_F = \sqrt{tr(A^T A)}$.

2) For all $x, y \in Sym^n_{++}$, the so-called affine invariant geodesic distance is as follows:

$$\delta_{G2}(x, y) = \left\| log(x^{-\frac{1}{2}} y x^{-\frac{1}{2}}) \right\|_F \tag{15}$$

Remark 4: $x \in Sym^n_{++}$, then $x = u \times diag(\lambda_1, \cdots, \lambda_n) \times u^T$, where $\lambda_i > 0$ are the eigenvalues of x, u consists of the orthonormal eigenvectors of x, then $log(x)$ is defined as $log(x) = u \times diag(log(\lambda_1), \cdots, log(\lambda_n)) \times u^T$.

III. RELATED WORKS

In recent years, Sym^n_{++} has been widely used in computer vision, such as pedestrian detection [18], texture classification [13], [19], target recognition [5], motion recognition [20], and face recognition [21]. Sparse coding has been a research hotspot because of its good representation ability. However, the existing sparse coding algorithms are mostly based on Euclidean space. The sparse coding methods of SPD manifold needs further study.

A. KERNEL LEARNING

In machine learning problems, some problems may not be modeled directly in Euclidean space, or the model may not be valid. Typically, data is mapped to RKHS can solve this problem. Mapping data from the original space to RKHS requires defining the kernel function, which the kernel function determines the impact of the model on RKHS. This makes kernel learning a key problem, because kernel function defines the relationship between the original data points and the data on RKSH. In order to make kernel function on RKHS better reflect the geometric properties of the original data, geometric perception kernel learning (GKL) has attracted people's attention. The effect of a model based on geometric perception kernel depends on the selected geometric perception kernel and other information used, which may include category information or surrounding spatial information. According to the information sources used, the geometric perception kernel learning methods can be divided into two categories: one is the geometric perception kernel learning method based on single-source information, and the other is the geometric perception kernel learning method based on multi-source information.

The kernel learning method based on single source information only focuses on the geometric structure information of the data itself and does not use other information. Diffusion kernel were proposed by Kondor and Lafferty [22]. Lafferty further explains the diffusion kernel that are generated from manifold data through thermodynamic equations [23]. It can better reflect the geometric structure of data. The research of Smola and Kondor [24] shows that the spectrum of Laplace graph can generate a series of geometric perception kernel after various filtering functions, including regularized Laplace kernel, diffusion kernel, random walk kernel and arccosine kernel. In addition, common manifold based data dimensionality reduction methods such as ISOMAP [25], Local Linear Embedding (LLE) [26], Laplacian Eigenmap (LE) [27] can also be unified in the framework of geometric perception kernel [28], [29]. These data reduction methods based on manifold learning can be understood as kernel principal component analysis or the Nyström formula of constructing kernel functions.

Based on multi-source information, the geometric perception kernel learning method not only uses the geometric structure information of the data, but also uses the category information and the surrounding spatial information. Combining the geometric structure information with the surrounding spatial information, Sindhwani's research [30] shows that the standard kernel function can directly calculate the kernel matrix of the new data points by using the geometric structure information of the existing data. Sindhwani's idea was born out of a primitive RKHS. A new RKHS is determined by the distribution of the data sample. Let RKHS $H = (S(\omega), \langle \bullet, \bullet \rangle_H)$ generated as a kernel function k, $w_i (i = 1, \cdots, N)$ be a sample of the data space Ω, and $\omega = \{\omega_1, \cdots, \omega_N\} \subseteq \Omega$ is the sample of the data space, so for any $f, g \in S(\omega)$,

$$\langle f, g \rangle_{\widetilde{H}} = \langle f, g \rangle_H + f^T(\omega) M g(\omega) \tag{16}$$

where $f(\omega) = \begin{bmatrix} f(\omega_1) \\ \vdots \\ f(\omega_N) \end{bmatrix} \in \mathbb{R}^N$, $g(\omega) = \begin{bmatrix} g(\omega_1) \\ \vdots \\ g(\omega_N) \end{bmatrix} \in \mathbb{R}^N$, M is a symmetric positive semidefinite matrix. Sindhwani proved $\widetilde{H} = (S(\omega), \langle \bullet, \bullet \rangle_{\widetilde{H}})$ that it also is an RKHS and derived the form of the corresponding kernel function with $\widetilde{H}$:

$$\widetilde{k}(x_i, x_j) = k(x_i, x_j) - \mu k_{x_i}^T (I + MK)^{-1} M k_{x_j} \tag{17}$$

where K is the kernel matrix generated by the data sample ω, $k_{x_i}^T$ is the kernel vector generated by the x_i and data sample. M is the regular matrix based on the data sample, and is usually represented using Laplace mapping.

$\widetilde{k}(x_i, x_j)$ is a data-dependent kernel, which takes into account the distribution of the original data. This allows it to better reflect the geometry of the original data than the original kernel.

B. DICTIONARY LEARNING AND SPARSE CODING METHODS

Applying the sparse coding algorithm to SPD manifold, the problem can be transformed into how to use the linear representation of the SPD matrix to remain an SPD matrix. The eigenvalue of SPD matrix is always greater than 0, but SPD manifold is not linear space. This means that sparse coding of SPD matrix will face two problems. The first problem is how to solve the linear combination of SPD matrix still is SPD matrix, and the second problem is how to measure the relationship between two SPD matrices. According to the dictionary learning of SPD manifold and the sparse coding model, we can roughly divide the solution into three types:

1) Dictionary learning and sparse coding which inherent in SPD manifold;
2) SPD manifold dictionary learning and sparse coding method mapped to tangent space;
3) SPD manifold dictionary learning and sparse coding method based on kernel method.

1) DICTIONARY LEARNING AND SPARSE CODING METHODS INHERENT IN SPD MANIFOLD

The idea of sparse coding inherent in SPD manifold is to directly model on SPD manifold, that is, the dictionaries is also SPD matrices. The original data to be encoded is represented by a linear combination of dictionaries. And then the Logdet divergence or Frobenius matrix norm is used to measure the reconstruction error. For an SPD matrix, the linear combination of them is not guaranteed to be an SPD matrix. There are two solutions, the first is to ignore the positive definite condition of the matrix, and the second is to set the limit of the positive definite condition. If the positive definite condition is ignored, it is equivalent to ignoring the fact that the eigenvalue of the matrix is greater than 0, which is equivalent to the fact that the model becomes relatively simple to solve the objective function when calculate in symmetric matrix space. However, the original SPD manifold structure was ignored. The constraint of positive definite condition makes the objective function become non-convex, and it takes a long time to solve.

Sivalingam proposed Tensor Sparse Coding (TSC) for positive definite matrices [31], which represents a positive definite matrix as a linear combination of positive definite matrices, and then restricts the combined matrix to positive definite, that is, dictionary set is $D = \{d_1, \cdots, d_L\} \subseteq Sym_{++}^n$, X can be expressed as $\hat{X} = \sum_{i=1}^{L} y_i d_i$, where y_i is the combination coefficient, and then the Logdet divergence is used to measure the reconstruction error:

$$D_d(\hat{X}, X) = tr(X^{-1}\hat{X}) - log[det(X^{-1}\hat{X})] - d \tag{18}$$

The solution of the sparse coding objective function becomes the MAXDET optimization problem, which can be solved by the interiorpoint algorithm. In addition, Sivalingam further proposed a Tensor Dictionary Learning algorithm based on Logdet divergence [32]. However, the

computational complexity of the above two methods is too high. Sra and Cherian used the Frobenius matrix norm as the error metric to learn a matrix dictionary with rank-1 to represent the SPD matrix [33]. However, existing research results [17], [34] show that the Frobenius matrix norm is simply to vectorize the SPD matrix and then calculate the vector norm value between two vectorized SPD matrices. The Frobenius norm is not a good measure because it ignores the geometry of the SPD manifold. Anoop proposed that the SPD matrix represents a linear combination of the SPD dictionary matrix and strictly requires that the combination coefficient is greater than or equal to 0 [1], for $x_i \in Sym_+^n$, the dictionary set is $D = \{d_1, \cdots, d_L\} \subseteq Sym_+^n$ and $y_i \in \mathbb{R}^+$, where y_i is the combination coefficient, and then the Riemannian geodesic distance is used to measure the reconstruction error, the objective function established is:

$$\min_{D, y \in \mathbb{R}_+^L} \frac{1}{2} \sum_{i=1}^{n} d_R^2(x_i, Dy_i) + \sum_{i=1}^{n} \text{Sparse}(y_i) + \Omega(D) \quad (19)$$

where $Dy_i = \sum_{j=1}^{L} y_{ij} D_j$, Sparse(•) is the sparse regular term and $\Omega(D)$ is the regular term for the dictionary D. And $d_R^2(x_i, Dy_i) = \left\| log(X^{-\frac{1}{2}} Y X^{-\frac{1}{2}}) \right\|_F$ is geodesic distance. Similarly, because the coding coefficient is restricted to be greater than or equal to 0, the objective function is also non-convex, and optimization requires a large time overhead.

2) SPD MANIFOLD DICTIONARY LEARNING AND SPARSE CODING METHODS MAPPED TO TANGENT SPACE

The sparse coding and dictionary learning model in Euclidean space cannot be directly applied to SPD manifold. In order to apply the existing vector-based sparse coding algorithm to the SPD matrix manifold, one idea is to map the SPD matrix manifold data onto the vector space. There are two mapping spaces available. The first is the tangent space of the mean point of the SPD manifold data, which can map one SPD matrix to the tangent vector of the tangent space of another SPD matrix. The tangent space of SPD matrix is linear space, and the tangent vector on the tangent space is symmetric matrix. The second is Exp mapping, which can map the tangent vector on the tangent space of an SPD matrix to the SPD manifold. Using these two mappings, Zhang mapped the region covariance feature to the tangent space, and then obtained the sparse representation [35] by vectrization. Guo maps the SPD matrix manifold to the tangent space [36], and then the SPD matrix of each mapping on the tangent space can be represented by linear combinations of the SPD matrices of other mappings. Yuan calculated the sparse representation by mapping SPD manifold to tangent space [37], and then applied it to motion recognition problem. Although the method based on Log-Euclidean mapping can be applied to SPD manifold by mapping SPD matrix manifold data to linear space, the sparse coding method based on Euclidean vectors can be applied to SPD manifold. More importantly, the tangent space of a data point of an SPD matrix manifold can only retain the local geometric structure of the neighborhood of that point, so if all data points of an SPD matrix manifold are mapped to the tangent space of the same point. The tangent space cannot reflect the geometric relationship between all points, so learning dictionary may not be optimal, which will affect the effect of sparse coding.

3) SPD MANIFOLD DICTIONARY LEARNING AND SPARSE CODING METHODS MAPPED TO RKHS

SPD manifold are nonlinear space. Another approach is to map the SPD matrix to the RKHS. First, we need to define a kernel function that it will produce a unique RKHS.

Harandi uses the Stein kernel function to map the SPD matrix manifold data to RKHS [13], [17], and then builds a model on RKHS to solve the problems of sparse coding and dictionary learning. The training data $X = \{x_1, \cdots, x_N\} \subseteq Sym_{++}^n$ and $D = \{d_1, \cdots, d_L\} \subseteq Sym_{++}^n$ the initialized dictionary are mapped to the RKHS by the non-linear mapping generated by the Stein kernel function:

$$X = \{x_1, \cdots, x_N\} \xrightarrow{\varphi} \varphi(X) = \{\varphi(x_1), \cdots, \varphi(x_N)\} \quad (20)$$

$$D = \{d_1, \cdots, d_L\} \xrightarrow{\varphi} \varphi(D) = \{\varphi(d_1), \cdots, \varphi(d_L)\} \quad (21)$$

Then $\varphi(X)$ can be represented the linear combination of $\varphi(D)$ represented by RKHS and the regularization 1-Norm restrictions on the combination coefficients can be used to establish the objective function on RKHS:

$$\min_{D,Y} \sum_{i=1}^{N} \left\| \varphi(x_i) - \sum_{j=1}^{L} y_{ij} \varphi(d_j) \right\|^2 + \lambda \sum_{i=1}^{N} \|y_i\|_1 \quad (22)$$

Zhang proposed an SPD manifold online dictionary learning algorithm based on the Stein kernel function [38]. The Gaussian kernel function based on the Stein divergence is only positively definite for some bandwidth parameters, so the Stein divergence is only an approximation of the Riemannian metric. So Barachant proposed a sparse coding and dictionary learning method based on the Riemannian kernel function and applied it in the human-machine interface [39]. Li based on the Log-Euclidean framework and proposed to use the LE kernel function to map the SPD matrix manifold data to RKHS to solve the sparse coding and dictionary learning problems [16]. Similarly, the training data $X = \{x_1, \cdots, x_N\} \subseteq Sym_+^n$ and the initialized dictionary $D = \{d_1, \cdots, d_L\}$ are mapped to the RKHS through the non-linear mapping φ generated by the LE kernel function, and then the objective function is established on the RKHS:

$$\min_{D,Y} \sum_{i=1}^{N} \left\| \varphi(x_i) - \sum_{j=1}^{L} y_{ij} \varphi(d_j) \right\|^2 + \lambda \sum_{i=1}^{N} \|y_i\|_1 \quad (23)$$

The objective function can be derived as:

$$\min_{D,Y} \sum_{i=1}^{N} \left[-2 \sum_{j=1}^{L} y_{ij} k_g(x_i, d_j) + \sum_{j=1}^{L} \sum_{p=1}^{L} y_{ij} y_{ip} k_g(d_j, d_p) \right] + \lambda \sum_{i=1}^{N} ||y_i||_1 \quad (24)$$

where $k_g(\bullet, \bullet)$ is the LE Gaussian kernel function, which is similar to K-SVD [39] when the dictionary is updated. Although the method based on the LE kernel function has achieved good results in face recognition and picture classification, it does not reflect the geometric structure of the SPD manifold. The key to mapping the SPD matrix manifold data onto RKHS while preserving the geometry of the manifold is the construction of the kernel function. Therefore, in the SPD manifold dictionary learning and sparse coding methods based on kernel methods, the selection and construction of kernel functions is a key issue.

Although Sym_+^n does not form a linear space, we can configure a metric or divergence to measure the difference between the two SPD. The metric that is commonly used today is the Riemannian metric. However, from the perspective of computational complexity, divergence seems to be simpler than Riemannian metric [17]. Harandi uses the Stein divergence and Jeff divergence defined by the Bregman divergence to measure the difference between the two SPD [17]. Zhang et al argue that directly using the original eigenvalues may be problematic, and propose a discriminative Stein kernel, in which an extra parameter vector is defined to adjust the eigenvalues of input SPD matrices [40]. Asha Das et al applied RKHS-based sparse coding and dictionary learning to breast tumor classification methods [41]. Zhi Gao et al propose a novel SPD distance measure for the similarity-based algorithm [42]. That is a tailored set-to-set distance measure by making use of the family of $\alpha - \beta$ divergences and further propose to learn the point-to-set transformation and the set-to-set distance measure jointly, yielding a powerful similarity-based algorithm on SPD manifold.

Definition 7: Let $\zeta : Sym_+^n \to \mathbb{R}$ be a strictly convex differentiable function on the definition Sym_+^n, then the Bregman divergence $d_\zeta : Sym_+^n \times Sym_+^n \to [0, +\infty]$ on Sym_+^n is defined as: for any $X, Y \in Sym_+^n$,

$$d_\zeta(X, Y) = \zeta(X) - \zeta(Y) - \left\langle \frac{\partial \zeta(Y)}{\partial Y}, X - Y \right\rangle \quad (25)$$

where $\langle X, Y \rangle = tr(X^T Y)$, $\frac{\partial \zeta(Y)}{\partial Y} = \begin{bmatrix} \frac{\partial \zeta(Y)}{\partial y_{11}} & \cdots & \frac{\partial \zeta(Y)}{\partial y_{1n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial \zeta(Y)}{\partial y_{n1}} & \cdots & \frac{\partial \zeta(Y)}{\partial y_{nn}} \end{bmatrix}$.

Using the Bregman divergence, the Stein divergence and Jeff divergence can be further defined.

Definition 8: Let the strictly convex differentiable function in Bregman divergence be $\zeta(X) = -log|X|$, and the definition of Jeff divergence is as follows:

$$J(X, Y) = \frac{1}{2} d_\zeta(X, Y) + \frac{1}{2} d_\zeta(Y, X) \quad (26)$$

Definition 9: Let the strictly convex differentiable function in Bregman divergence be $\zeta(X) = -log|X|$, and the definition of Stein divergence is as follows:

$$S(X, Y) = \frac{1}{2} d_\zeta(X, \frac{X+Y}{2}) + \frac{1}{2} d_\zeta(Y, \frac{X+Y}{2}) \quad (27)$$

Bregman divergence is only an approximation to the geodesic distance of SPD manifold, so the kernel function based on Bregman divergence has some defects, such as it cannot fully reflect the true distance between two point of SPD manifold.

IV. SPD DATA DICTIONARY LEARNING FOR DATA-DEPENDENT KERNEL LEARNING BASED ON RIEMANNIAN METRIC

A. SPD DATA AND RIEMANNIAN METRIC

At present, using local covariance descriptor to extract feature data from image or video data is a very popular method. Such feature matrices are SPD matrices. Sym_{++}^n is not a linear space so that matrix addition and multiplication of real numbers does not ensure the result is still SPD matrix. The finite-dimensional linear space is isomorphic with the Euclidean space. Various machine learning algorithms developed on Euclidean space that cannot be directly applied to Sym_{++}^n.

Although Sym_{++}^n cannot constitute a linear space, we can define a measure on Sym_{++}^n. This measure measures the distance between two SPD matrices. Here we use the Riemannian metric, which is the most commonly used. The geometric structure of Sym_{++}^n is usually kept by Riemannian metric. Only the Riemannian metric is defined and the geodesic distance generated by the Riemannian metric. Because the distance between the tangent vectors on the tangent space can reflect the geodesic distance of the corresponding points on Sym_{++}^n. Log-Euclidean metric is a Riemannian metric that it is usually used.

B. SAMPLE-DEPENDENT AND LEARNABLE KERNEL

Because Sym_{++}^n does not constitute a linear space in the real field, we cannot do dictionary learning directly in Sym_{++}^n. This is because dictionary learning involves many linear operations. The algorithm in this paper is to map Sym_{++}^n to RKHS H and then do dictionary learning on H.

H can be uniquely determined by the kernel function. Now, the downside of the kernel approach in machine learning is that we do not have a lot of kernels to choose. For specific machine learning, existing kernel methods and samples are not closely related. Whatever the sample is, the original data maps to the same RKHS. This paper proposes a framework based on data-dependent and learnable kernel as follows:

$$k(x, y) = k_b(x, y) + \zeta \widetilde{\beta}^T(x) M \widetilde{\beta}(y), \zeta > 0 \quad (28)$$

where $k_b(x, y)$ is a kernel function, called the basic kernel function. It can be selected according to needs, but must satisfy the conditions of SPD.

Remark 5: M is a symmetric semi-positive definite matrix, which is the learnable part of the kernel function. For example, in this article, M will be optimized according to the requirements of dictionary learning.

Remark 6: $\widetilde{\beta}(x) = \begin{bmatrix} \beta(x, x_1) \\ \vdots \\ \beta(x, x_N) \end{bmatrix} \in \mathbb{R}^N$, where $\{x_1, \cdots, x_N\}$ is the dataset, N is the number of given learning samples. $\beta(x, y)$ is any binary function, which can be selected according to the specific application.

For any finite number of data $\{x_1, \cdots, x_N\}$, there is

$$\begin{aligned} K &= \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_N) \\ \vdots & \ddots & \vdots \\ k(x_N, x_1) & \cdots & k(x_N, x_N) \end{bmatrix} \\ &= \begin{bmatrix} k_b(x_1, x_1) & \cdots & k_b(x_1, x_N) \\ \vdots & \ddots & \vdots \\ k_b(x_N, x_1) & \cdots & k_b(x_N, x_N) \end{bmatrix} \\ &\quad + \zeta \begin{bmatrix} \widetilde{\beta}^T(x_1) M \widetilde{\beta}(x_1) & \cdots & \widetilde{\beta}^T(x_1) M \widetilde{\beta}(x_N) \\ \vdots & \ddots & \vdots \\ \widetilde{\beta}^T(x_N) M \widetilde{\beta}(x_1) & \cdots & \widetilde{\beta}^T(x_N) M \widetilde{\beta}(x_N) \end{bmatrix} \\ &= K_b + \zeta \Phi^T M \Phi \end{aligned} \tag{29}$$

where $\Phi = \left[\widetilde{\beta}(x_1) \cdots \widetilde{\beta}(x_N)\right] \in \mathbb{R}^{N \times N}$. Because $k_b(\bullet, \bullet)$ is a kernel function, K_b is an SPD matrix, and because M is a symmetric semi-definite definite matrix, $\Phi^T M \Phi$ is also a symmetric semi-definite definite matrix. Therefore, K is an SPD matrix.

Note that belkin's kernel framework $\widetilde{k}(x_i, x_j) = k(x_i, x_j) - \mu k_{x_i}^T (I + MK)^{-1} M k_{x_j}$ starts from space [43]. It first redefines an inner product, and then proves that the inner product satisfies the sufficient and necessary conditions for reproducing kernel, while our proposed framework starts from the kernel function.

C. DICTIONARY LEARNING BASED ON KERNEL LEARNING AND RIEMANNIAN METRIC (KLRM-DL)

This paper proposes dictionary learning based on kernel learning and Riemannian metric. KLRM-DL first builds an RKHS H based on the data-dependent and learnable kernel function $k(\bullet, \bullet)$ proposed in this paper, then uses $k(\bullet, \bullet)$ to transform a given dictionary learning sample to H, and finally do dictionary learning in H.

1) BASIC KERNELS BASED ON LOG-EUCLIDEAN METRIC

KLRM-DL selects the radial basis kernel function [15] and the Log-Euclidean metric as the basis kernel function for data-dependent and learnable kernel functions:

$$k_b = e^{-\delta_{G1}(x,y)} \tag{30}$$

where $\delta_{G1}(x, y)$ is Log-Euclidean metric as defined in Eq. 14.

2) THE CONSTRUCTION OF DICTIONARIES IN RKHS

Since Sym_{++}^n does not constitute a linear space, it makes little sense to learn dictionaries on Sym_{++}^n. And our machine learning is only done in RKHS H, not in Sym_{+}^n. Therefore, we only need to learn dictionaries in H.

Let $X = \{x_1, \cdots, x_N\} \subseteq Sym_{++}^n$ be a set of data points used for dictionary learning, and use the reproducing kernel of H to transform X into a set of new data points $\varphi(X) = \{\varphi(x_1), \cdots, \varphi(x_N)\} \subseteq H$. KLRM-DL uses a linear combination of $\varphi(X)$ to construct the dictionaries $\{\varphi(d_1), \cdots, \varphi(d_L)\} \subseteq H$:

$$\varphi(d_i) = \sum_{j=1}^{N} b_{ij} \varphi(x_j) \in H, i = 1, \cdots, L \tag{31}$$

where $B = \begin{bmatrix} b_{11} & \cdots & b_{1N} \\ \vdots & \ddots & \vdots \\ b_{L1} & \cdots & b_{LN} \end{bmatrix} = \begin{bmatrix} B_{1Row} \\ \vdots \\ B_{LRow} \end{bmatrix} \in \mathbb{R}^{L \times N}$. And B is a dictionary generation matrix.

Obviously, learning a dictionary in H is learning B. And then we can derive:

$$\begin{aligned} \left\langle \varphi(d_p), \varphi(d_q) \right\rangle &= \left\langle \sum_{i=1}^{N} b_{pi} \varphi(x_i), \sum_{j=1}^{N} b_{qj} \varphi(x_j) \right\rangle \\ &= \sum_{i=1}^{N} \sum_{j=1}^{N} b_{pi} b_{qj} \left\langle \varphi(x_i), \varphi(x_j) \right\rangle \\ &= \sum_{i=1}^{N} \sum_{j=1}^{N} b_{pi} b_{qj} k(x_i, x_j) \\ &= B_{pRow} K B_{qRow}^T \end{aligned} \tag{32}$$

And finally we come out:

$$\begin{aligned} &\begin{bmatrix} \langle \varphi(d_1), \varphi(d_1) \rangle & \cdots & \langle \varphi(d_1), \varphi(d_L) \rangle \\ \vdots & \ddots & \vdots \\ \langle \varphi(d_L), \varphi(d_1) \rangle & \cdots & \langle \varphi(d_L), \varphi(d_L) \rangle \end{bmatrix} \\ &= \begin{bmatrix} B_{1Row} K B_{1Row}^T & \cdots & B_{1Row} K B_{LRow}^T \\ \vdots & \ddots & \vdots \\ B_{LRow} K B_{1Row}^T & \cdots & B_{LRow} K B_{LRow}^T \end{bmatrix} \\ &= \begin{bmatrix} B_{1Row} K [B_{1Row}^T \cdots B_{LRow}^T] \\ \vdots \\ B_{LRow} K [B_{1Row}^T \cdots B_{LRow}^T] \end{bmatrix} \\ &= \begin{bmatrix} B_{1Row} K B^T \\ \vdots \\ B_{LRow} K B^T \end{bmatrix} = B K B^T \end{aligned} \tag{33}$$

D. THE MODEL OF KLRM-DL

The objective function of dictionary learning is

$$\sum_{i=1}^{N} \left[\left\| \varphi(x_i) - \sum_{j=1}^{L} a_{ij} \varphi(d_j) \right\|^2 + \lambda \text{Sparse}(A_{iRow}) \right] \tag{34}$$

where $A = \begin{bmatrix} a_{11} & \cdots & a_{1L} \\ \vdots & \ddots & \vdots \\ a_{N1} & \cdots & a_{NL} \end{bmatrix} = \begin{bmatrix} A_{1Row} \\ \vdots \\ A_{NRow} \end{bmatrix}$ is a sparse coding matrix for dictionaries. Then

$$\left\| \varphi(x_i) - \sum_{j=1}^{L} a_{ij}\varphi(d_j) \right\|^2 = \|\varphi(x_i)\|^2 + A_{iRow}BKB^T A_{iRow}^T - 2A_{iRow}BK_{iRow}^T \quad (35)$$

So

$$\sum_{i=1}^{N} \left\| \varphi(x_i) - \sum_{j=1}^{L} a_{ij}\varphi(d_j) \right\|^2 = \sum_{i=1}^{N} \|\varphi(x_i)\|^2 + tr(ABKB^T A^T) - 2tr(ABK) \quad (36)$$

Since $\sum_{i=1}^{N} \|\varphi(x_i)\|^2$ has nothing to do with B and A, the objective function for dictionary learning is simplified as:

$$\min_{A,B} tr(ABKB^T A^T) - 2tr(ABK) + \lambda \sum_{i=1}^{N} \text{Sparse}(A_{iRow}) \quad (37)$$

In the above objective function, the sparse coding matrix A is not necessary. The dictionary generation matrix B is what we want to get.

Further, KLRM-DL uses the data-dependent and learnable kernel function k proposed in this paper, where the learning samples use the learning samples X to do dictionary learning. k in this paper is

$$K = K_b + \zeta \Phi^T M \Phi \quad (38)$$

where $K = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_N) \\ \vdots & \ddots & \vdots \\ k(x_N, x_1) & \cdots & k(x_N, x_N) \end{bmatrix}$,

$$\Phi = \left[\widetilde{\beta}(x_1), \cdots, \widetilde{\beta}(x_N)\right] = \begin{bmatrix} \beta(x_1, x_1) & \cdots & \beta(x_1, x_N) \\ \vdots & \ddots & \vdots \\ \beta(x_N, x_1) & \cdots & \beta(x_N, x_N) \end{bmatrix} \in \mathbb{R}^{N\times N}. \quad (39)$$

Therefore, the objective function of KLRM-DL is:

$$\min_{A,B,M} \sum_{i=1}^{N} \|\varphi(x_i)\|^2 + tr(AB(K_b + \zeta \Phi^T M \Phi)B^T A^T) - 2tr(AB(K_b + \zeta \Phi^T M \Phi)) + \lambda \sum_{i=1}^{N} \text{Sparse}(A_{iRow}) \quad (40)$$

Remark 7: If you take $\beta(x, y) = k_b(x, y)$, then

$$\Phi = \begin{bmatrix} k_b(x_1, x_1) & \cdots & k_b(x_1, x_N) \\ \vdots & \ddots & \vdots \\ k_b(x_N, x_1) & \cdots & k_b(x_N, x_N) \end{bmatrix} \in \mathbb{R}^{N\times N}.$$

And then

$$K = K_b + \zeta K_b M K_b = K_b(I_N + \zeta M K_b) \quad (41)$$

E. SPARSE CODING

After KLRM-DL, B and kernel function feature matrix M will be determined.

For any $x \in Sym_{++}^n$, the objective function of its sparse coding matrix is:

$$\left\| \varphi(x) - \sum_{j=1}^{L} \alpha_j \varphi(d_j) \right\|^2 = \|\varphi(x)\|^2 + a^T BKB^T a - 2a^T Bk(x) \quad (42)$$

where $a = [\alpha_1, \cdots, \alpha_L]^T$ and

$$k(x) = \begin{bmatrix} k(x, x_1) \\ \vdots \\ k(x, x_N) \end{bmatrix} = \begin{bmatrix} k_b(x, x_1) \\ \vdots \\ k_b(x, x_N) \end{bmatrix} + \zeta \begin{bmatrix} \widetilde{\beta}^T(x_1) M \widetilde{\beta}(x) \\ \vdots \\ \widetilde{\beta}^T(x_N) M \widetilde{\beta}(x) \end{bmatrix} = k_b(x) + \zeta \Phi^T M \widetilde{\beta}(x)$$

So the objective function of sparse coding of x is:

$$\min_{a} a^T BKB^T a - 2a^T Bk(x) + \lambda \text{Sparse}(a) \quad (43)$$

V. SOLUTION TO KLRM-DL

The final objective function can be expressed as:

$$\min_{A,B,M} \sum_{i=1}^{N} \left\| \varphi(x_i) - A_{iRow} \sum_{j=1}^{L} b_{ij}\varphi(x_j) \right\|^2 + \lambda \text{Sparse}(A_{iRow}) = \min_{A,B,M} \sum_{i=1}^{N} (k(x_i, x_i) - 2A_{iRow}Bk(x_i) + A_{iRow}BK(X, X)B^T A_{iRow}^T) + \lambda \text{Sparse}(A_{iRow}) \quad (44)$$

where

$$K(X, X) = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_N) \\ \vdots & \ddots & \vdots \\ k(x_N, x_1) & \cdots & k(x_N, x_N) \end{bmatrix}$$

The key to dictionary learning and sparse coding on SPD manifold is to choose a suitable kernel function. Kernel functions use the data-dependent kernel functions $k(x, y) = k_b(x, y) + \zeta \widetilde{\beta}^T(x) M \widetilde{\beta}(y), \zeta > 0$ proposed in this paper. This kernel function learning and dictionary learning can be combined.

KLRM-DL needs to learn the dictionaries and kernel function parameters. The dictionaries and kernel parameters can be learned together or separately. This article adopts the strategy is learning together.

Given training data $X = \{x_1, \cdots, x_N\} \subseteq Sym_{++}^n$, learning dictionaries and kernel parameters use an alternating optimization strategy, which is divided into three steps in each iteration: update the sparse coding matrix A, update dictionary coding matrix B and update kernel parameters M.

The other two variables are fixed when each variable is updated. The steps are explained below.

1) Update the sparse coding matrix A. In the process of A, fixed M and B,the objective function is:

$$\min_{A} \sum_{i=1}^{N} (-2A_{iRow}Bk(x_i) + A_{iRow}BK(X,X)B^T A_{iRow}^T) + \lambda \text{Sparse}(\boldsymbol{A}_{iRow}) \tag{45}$$

The solution of the above equation is a typical Lasso problem which can be quickly solved by the toolbox SPAMS [44] or CVX [45].

2) Update dictionary coding matrix B. During this phase, the coding results A and kernel parameters M are fixed. The objective function is:

$$\begin{aligned}\min_{B} &\sum_{i=1}^{N} (-2A_{iRow}Bk(x_i) + A_{iRow}BK(X,X)B^T A_{iRow}^T) \\ &= \min_{B} (-2tr(ABK(X,X)) + tr\left(ABK(X,X)B^T A^T)\right)\end{aligned} \tag{46}$$

The function about B is:

$$f(B) = -2tr(ABK(X,X)) + tr(ABK(X,X)B^T A^T) \tag{47}$$

The derivative of $f(B)$ with respect to B, we get:

$$\begin{aligned}\frac{\partial f}{\partial B} &= -2\frac{\partial tr(ABK(X,X))}{\partial B} + \frac{\partial tr(ABK(X,X)B^T A^T)}{\partial B} \\ &= -2A^T K(X,X) + 2A^T ABK(X,X)\end{aligned} \tag{48}$$

Let the derivative be 0, we can get the updated formula:

$$B = (A^T A)^{-1} A^T \tag{49}$$

3) Update kernel parameter. When updating the kernel parameters M, A and B are fixed, and the objective function is:

$$\begin{aligned}\min_{M} &\sum_{i=1}^{N} (k(x_i,x_i) - 2A_{iRow}Bk(x_i) + A_{iRow}BK(X,X)B^T A_{iRow}^T) + \lambda tr(M^T M) \\ &= \min_{M} tr(\Phi(Z,X)^T M \Phi(Z,X)) - 2tr(AB\Phi(Z,X)^T M \Phi(Z,X)) \\ &\quad + tr(AB\Phi(Z,X)^T M \Phi(Z,X)B^T A^T) + tr(M^T M)\end{aligned} \tag{50}$$

where $tr(M^T M)$ is the regular term for M, $\Phi(Z,X) = \begin{bmatrix} \beta(z_1,x_1) & \cdots & \beta(z_1,x_N) \\ \vdots & \ddots & \vdots \\ \beta(z_Q,x_1) & \cdots & \beta(z_Q,x_N) \end{bmatrix} \in \mathbb{R}^{Q\times N}$ and $Z = \{z_1,\cdots,z_Q\}$ can be a subset of $X = \{x_1,\cdots,x_N\}$, not necessarily $Z = X$.

Let

$$\begin{aligned}f(M) = &tr(\Phi(Z,X)^T M \Phi(Z,X)) \\ &- 2tr(AB\Phi(Z,X)^T M \Phi(Z,X)) \\ &+ tr(AB\Phi(Z,X)^T M \Phi(Z,X)B^T A^T) \\ &+ tr(M^T M)\end{aligned} \tag{51}$$

There are two main ways to solve the above equation, one is Trust-region method [46], the other is Riem-CG method [47]. Riem-CG can get results faster and the results are stable, so we use Riem-CG to update M in this paper.

For a smooth nonlinear function $f(x), x \in \mathbb{R}^n$, the updated formula of the Riem-CG method in the $t+1$ iteration is:

$$x_{t+1} = x_t + \gamma_t \xi_t \tag{52}$$

where ξ_t is the gradient descent direction:

$$\xi_t = -\text{grad}f(x_k) + \mu_k \xi_{t-1} \tag{53}$$

where $\text{grad}f(x_k)$ is the gradient of the function f at x_k and μ_k that is the conversion between ξ_t and ξ_{t-1}:

$$\mu_t = \frac{(\text{grad}f(x_t))^T(\text{grad}f(x_t) - \text{grad}f(x_{t-1}))}{\text{grad}f(x_{t-1})^T \text{grad}f(x_{t-1})} \tag{54}$$

The step size γ_t can be calculated by line search [48]. For Riem-CG, the difference is that the gradient uses the Riemannian gradient $\text{grad}_{\text{Riem}}f(M_k)$, and the relationship between the Riemannian gradient and the Euclidean space gradient [49] is:

$$\text{grad}_{\text{Riem}}f(M_k) = \frac{1}{2}M(\frac{\partial f(M)}{\partial M} + (\frac{\partial f(M)}{\partial M})^T)M \tag{55}$$

where $\frac{\partial f(M)}{\partial M}$ represents the Euclidean space gradient of $f(M)$ to M. Since $\text{grad}_{\text{Riem}}(M_k)$ and $\text{grad}_{\text{Riem}}f(M_{k-1})$ belong to different tangent spaces $T_{M_k}Sym_+^n$ and $T_{M_{k-1}}Sym_+^n$, so the equation above cannot be applied directly. The solution is to use vector transfer, which transfers a tangent vector to a Riemannian manifold corresponding to the tangent vector on the point. The updated formula of the Riem-CG method $t+1$ iteration is:

$$M_{t+1} = M_t + \gamma_t \xi_t \tag{56}$$

where ξ_t is the gradient descent direction:

$$\xi_t = -\text{grad}_{\text{Riem}}f(M_t) + \mu\theta_{\gamma_t\xi_t}(\xi_{t-1}) \tag{57}$$

where $\theta_{\gamma_t\xi_t}(\xi_{t-1})$ represents vector transfer:

$$\theta_{\gamma_t\xi_t}(\xi_{t-1}) = \frac{d}{dx}exp_{M_{t-1}}(\gamma_t\xi_{t-1} + x\xi_{t-1})|_{x=0} \tag{58}$$

μ_t of Riem-CG method:

$$\mu_t = \frac{\langle \text{grad}_{\text{Riem}}f(M_t), \text{grad}_{\text{Riem}}f(M_t)\rangle}{\langle \text{grad}_{\text{Riem}}f(M_t), \text{grad}_{\text{Riem}}f(M_t)\rangle} - \frac{\theta_{\gamma_t\xi_{t-1}}(\text{grad}_{\text{Riem}}f(M_t))}{\langle \text{grad}_{\text{Riem}}f(M_t), \text{grad}_{\text{Riem}}f(M_t)\rangle} \tag{59}$$

The Riem-CG method requires the gradient of Euclidean space of $f(M)$ when calculating the Riemannian gradient of $f(M)$, and the European-style spatial gradient of $f(M)$ to M is:

$$\begin{aligned}\frac{\partial f(M)}{\partial M} &= \frac{\partial tr(\Phi(Z,X)^T M \Phi(Z,X))}{\partial M} \\ &\quad - 2\frac{tr(AB\Phi(Z,X)^T M \Phi(Z,X))}{\partial M} \\ &\quad + \frac{\partial tr(AB\Phi(Z,X)^T M \Phi(Z,X) B^T A^T)}{\partial M} \\ &\quad + \frac{\partial tr(M^T M)}{\partial M} \\ &= \Phi(Z,X)\Phi(Z,X)^T \\ &\quad - 2\Phi(Z,X)B^T A^T \Phi(Z,X)^T \\ &\quad + \Phi(Z,X)B^T A^T AB\Phi(Z,X)^T + 2M \end{aligned} \quad (60)$$

The matlab version of the Riem-CG method is integrated in the manopt toolbox [50]. The algorithm flow of dictionary learning and kernel learning can be saw in Algorithm1.

Algorithm 1 Dictionary Learning and Kernel Parameter Learning

1) Input: training data $X = \{x_1, \cdots, x_N\} \subseteq Sym^n_+$, data sample $Z = \{z_1, \cdots, z_Q\} \subseteq Sym^n_+$, numbers of dictionaries L, maximum number of iterations T_{iter}.
2) Output: dictionary B, kernel function k.
3) Initialization, initialization dictionary D generated by SPD manifold clustering algorithm, $t = 1$.
4) Repeat T_{iter}
5) Sparse coding: update A by equation (45)
6) Dictionary update: update B by equation (49)
7) Kernel parameter update: update M by equation (56)
8) Until $t++ = T_{iter}$

VI. EXPERIMENTS

In order to understand the effectiveness of the algorithm proposed in this paper and the influence of related parameters on model performance, this section compares the algorithm with other algorithms based on three real data. We have three datasets, the first is Queen Mary University of London (QMUL), the second is texture data (Brodatz), and the third is face data (FERET). The first step of the experiment is to test the effect of the parameters of the algorithm model on the experimental results, so as to determine better parameters. The second step of the experiment is to compare with other algorithms on three commonly used public data sets. The basic kernel function and data-dependent binary function used in this paper are Gaussian kernels based on Log-Euclidean metric. Our accuracy was measured by linear classifier based on ridge regression and classification method based on minimum reconstruction error, represented by CRR and 1-NN respectively.

TABLE 1. Distribution of the QMUL dataset.

label	b	bg	f	l	r
train	2256	2256	2256	2256	2256
test	2096	1107	1772	1502	2248

A. COMPARISON ALGORITHMS

The comparison algorithms are RSR [17], TSC [33] and K-LE-DLSC [13]. In RSR, RKHS is generated by using a kernel function and SPD data are transformed to RKHS by using the kernel function. Two kernel functions are used in RSR: $k_J(x,y) = e^{-\alpha J(x,y)}$ and $k_S(x,y) = e^{-\alpha S(x,y)}$, where $J(x,y)$ and $S(x,y)$ are Jeff and Stein divergence respectively, both of them are developed from Bregman divergence. The main difference between RSR and our algorithm is that the kernel functions in RSR are not learnable. By the way, our algorithm uses Riemannian metric, not divergence.

In TSC, dictionary learning and sparse coding are performed directly in the SPD space, instead of transferring to RKHS. Since the linear combination of SPD matrices is not necessarily an SPD matrix, TSC stipulates that the combination coefficient must be non-negative. The main difference between TSC and our algorithm is that TSC is not based on RKHS. In addition, the kernel functions in TSC are not learnable either.

As well known, SPD dataset does not form a linear space under the usual matrix addition and scalar multiplication. K-LE-DLSC defines a special matrix addition and scalar multiplication to make SPD dataset become a linear space, and then defines inner product to make SPD dataset become inner product space. Then, K-LE-DLSC defines kernel functions by using these newly-defined inner products. The main difference between K-LE-DLSC and our algorithm is that the kernel functions defined in K-LE-DLSC are not learnable.

B. EXPERIMENTS ON QMUL DATASET

In this section, we do two experiments on the QMUL dataset. The first experiment is to select appropriate parameters to make the algorithm better. The accuracy of the classification experiment was calculated to determine our parameter selection. The second experiment is to compare the algorithm proposed in this paper with other algorithms on the QMUL dataset, so as to verify the reliability of the algorithm proposed in this paper.

The QMUL dataset [51] is a dataset composed of pictures of the human head, and the pictures are collected from the terminal camera of the airport. This dataset has 20005 images. Image is divided into five types of 'back', 'background', 'front', 'left' and 'right', each category represents a direction of the head picture. 'Back' represents back image, 'backgound' represents background image, 'front' represents positive image, 'left' represents left image, 'right' represents right image. Some sample pictures are shown in Fig. 1. The dataset has been divided into training set and test set in advance. The number of pictures in each type of QMUL dataset and the specific division of training set and test set are shown in Table 1. In this experiment, each picture is represented by

FIGURE 1. QMUL sample picture.

a 13×13 SPD matrix, which means each position of the picture can be expressed by the 13-dimensional feature vector. Meanwhile, the 13-dimensional feature vector is calculated by the following formula:

$$f(x, y) = [I_L(x, y), I_a(x, y), I_b(x, y), \sqrt{I_x^2 + I_y^2}, arctan(\frac{|I_x^2|}{|I_y^2|}), G_1(x, y), \cdots, G_8(x, y)] \quad (61)$$

where $I_L(x, y)$, $I_a(x, y)$ and $I_b(x, y)$ are the three channel values of the CIElab color space. I_x and I_y is one-step-degree value respectively in the directions x and y of $I_L(x, y)$, and $G_i(x, y), i = 1, \cdots, 8$ is the response value of 8 DOOG (Difference of Offset Gaussian) fifilters at the position (x, y). Then by calculating the covariance matrix of all eigenvectors, we can get a 13×13 SPD matrix. During the experiment, 200 pictures from each class were randomly selected from the training set as training data to learn the dictionary and kernel parameters. Later, 300 pictures from each class were randomly selected from the test set, where 200 pictures were used as training data for the classififier and the remaining 100 pictures were used as the testing data to calculate the classification accuracy rate.

For ease of reading, the number of dictionaries is denoted as K_D, and the number of data dependencies selected is denoted as K_Z. Because the length of the encoding vector is consistent with the number of dictionaries K_D, K_D will have a great impact on the encoding result. When K_D is too small, the encoding will be so short that the information may be missing. When K_D is too large, the computational complexity will increase first, and then the encoding vector may contain redundant information, so that the number K_D is not necessarily as large as possible. Hence, in order to find out the K_D impact on the encoding result, we can change the value K_D but keep other settings of the algorithm model fixed. We can change K_D to find the impact according to the classification accuracy of the encoding result. In terms of experimental parameter settings, the number of data samples is 200 per class. The basic kernel function used is the kernel function based on Riemannian metric. The parameter of this kernel function *alpha* is set to 0.1. The kernel function of the data dependencies is also the kernel function based on

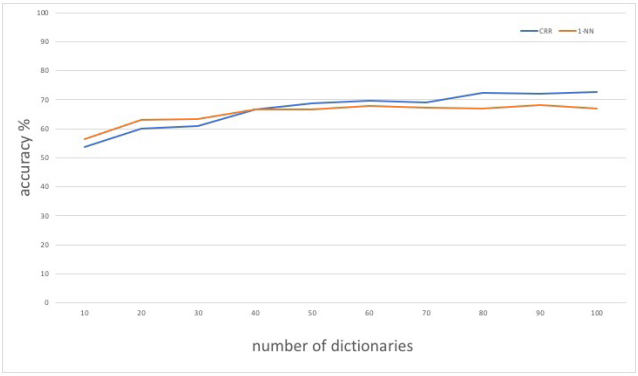

FIGURE 2. Relationship between classification accuracy and number of dictionaries.

Riemannian metric. The parameter *alpha* is also set to 0.1, and K_D is in steps of 10, from 10 to 100. The changes in the encoding result are shown in Fig. 2. It can be seen that the number of dictionaries is not as large as possible. When the number of dictionaries reaches a certain number, the effect of classification has no longer increase significantly.

In the data-dependent kernel function, data dependency is affected by two aspects, where one is the parameter, and the other is the data sample. Both the distribution and the number of data samples will affect the results of data dependency and functions. In order to explore the effect of the number of data samples K_Z on the encoding results, we can keep other settings of the algorithm model fixed, and only change the values K_Z. And then we can find the impact of K_Z based on the classification accuracy of encoding results. In terms of experimental parameter settings, the number of dictionaries is 50. The basic kernel function used is the Stein kernel function. The parameter *alpha* of the kernel function based on Riemannian metric is set to 0.1. The kernel function of the data dependency is also the kernel function based on Riemannian metric. K_Z is set in steps of 10, from 10 to 100. The variation of coding results with K_Z is shown in Fig. 3.

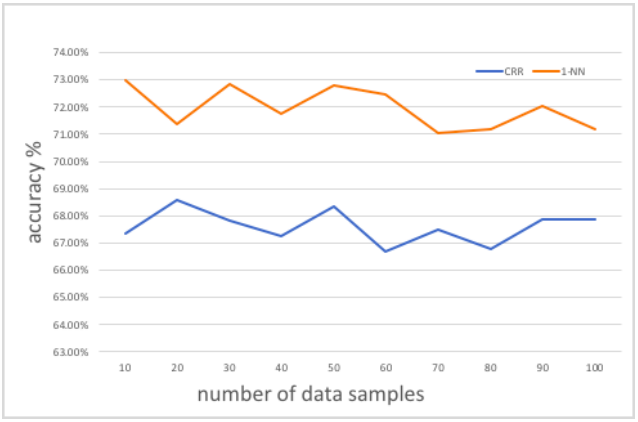

FIGURE 3. Relationship between classification accuracy and K_Z.

Finally, on the QMUL dataset, the proposed method is compared with dictionary learning and sparse coding

TABLE 2. QMUL dataset classification accuracy rate (%).

method	1-NN	CRR
Riem-DLSC	38.6	36.6
RSR-S	57.3	73.2
K-LE-DLSC-Poly	54.3	66.7
K-LE-DLSC-Exp	56.4	65.6
K-LE-DLSC-Gaussian	54.5	72.4
KLRM-DL	70.34	70.74

algorithms on SPD manifold in recent years, including K-LE-DLSC [13], RSR-S [17], and Riem-DLSC [1], which are all introduced in related section above. The classification accuracy results of each algorithm are shown in Table 1.

KLRM-DL have higher classification accuracy. Hence, Rime-DLSC has a much lower classification accuracy than the method based on the kernel method directly in SPD. However, in the kernel method, the RSR-S using the Stein kernel function. KLRM-DL has a higher classification accuracy rate than K-LE-DLSC-Gaussian using LE Gaussian kernel function. Meanwhile, the proposed method KLRM-DL showing a better effect than using the Stein kernel function only. Thus, it can be considered that KLRM-DL are effective.

C. EXPERIMENTS ON THE BRODATZ DATASET

In this section, we perform a classification experiment on the Brodatz dataset [52]. There are 112 texture photos in this dataset, each photo representing one type of texture. In this paper, two experiments will be performed on the Brodatz dataset, one is a grouping experiment with selected textures, and the other is a classification experiment for all texture images.

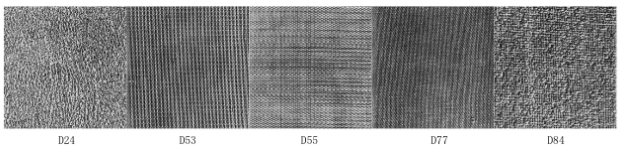

FIGURE 4. Brodatz dataset sample texture picture.

In the experiment of partial texture classification, we set up the same experiment as [31]. A total of several groups of classification experiments are set, and each group of experiments contained multiple sets of data. The pictures of some participating experiments are shown in the Fig. 4 above. The categories and numbers selected for each group of experiments are different, as shown in the Table 3 below.

For each set of data in each set of experiments, each picture is first down-sampled to size 256×256, and then divided into 64 non-overlapping local regions of size 32×32. For each position $I(x, y)$ of each local region, the feature vectors are calculated on $I(x, y)$. The vector uses the 5-dimensional feature vector, and then calculates the covariance matrix of all feature vectors in each local area, and obtains a matrix of size as the 5×5 SPD matrix representing this area.

TABLE 3. Brodatz group experiment data selection.

Experiment number	Picture selection
5-texture-1	D77, D84, D55, D53, D24
5-texture-2	Fabric.0000, Fabric.0017, Flowers.0002, Leaves.0006, Leaves.0013 [31]
5-texture-3	Fabric.0009, Fabric.0016, Fabric.0019, Flowers.0005, Food.0005 [31]
5-texture-4	Fabric.0007, Fabric.0009, Leaves.0003, Misc.0002, Sand.0000 [31]
10-texture-1	D4, D9, D19, D21, D24, D28, D29, D36, D37, D38
10-texture-2	Fabric.0009, Fabric.0016, Fabric.0019, Flowers.0005, Food.0005, Leaves.0003, Misc.0000, Misc.0002, Sand.0000, and Stone.0004 [31]
16-texture-1	D3, D4, D5, D6, D9, D21, D24, D29, D32, D33, D54, D55, D57, D68, D77, D84
16-texture-2	Fabric.0007, Fabric.0009, Fabric.0013, Fabric.0014, Fabric.0016, Flowers.0005, Food.0005, Grass.0001, Leaves.0003, Leaves.0008, Leaves.0012, Metal.0000, Metal.0002, Misc.0002, Sand.0000, Stone.0004 [31]

The generation of a 5-dimensional feature vector follows the following formula:

$$f(x, y) = [I(x, y), \partial I/\partial x, \partial I/\partial y, \partial^2 I/\partial x^2, \partial^2 I/\partial y^2] \quad (62)$$

where (x, y) is the position coordinate, $I(x, y)$ is the gray value of the position, $\partial I/\partial x$ and $\partial I/\partial y$ are a step in the x and y directions, $\partial^2 I/\partial x^2$ and $\partial^2 I/\partial y^2$ are two steps in x and y directions respectively. A 5×5 SPD matrix can be obtained for each local area, and each SPD matrix can generate 64 SPD matrices. That is, each type of texture has 64 SPD data.

In order to keep consistent with the experimental settings in [17], 14 SPD matrices are selected as data-dependent samples for each type of texture. The data-dependent samples remain unchanged in all experiments. During the classification experiment, 10 SPD matrices are randomly selected for each type of texture. Dictionaries are randomly selected 20 of the remaining 40 SPD matrices as the reference set of the nearest neighbor classifier, and the remaining 20 as the test set, sparsely encode the reference set and all SPD matrices in the test set with the learned dictionary. Finally the 1-NN classifier was used to calculate the classification accuracy rate. The classification results of each group of experiments are shown in Table 4.

TABLE 4. Grouping experiment of the dataset Brodatz classification accuracy rate (%).

Method	KLRM-DL	Riem-DLSC	RSR-S	K-LE-DLSC	TSC
5-texture-1	100.0	97.4	99.9	99.7	99.5
5-texture-2	88.7	73.8	86.6	87.9	93.2
5-texture-3	98.7	73.2	91.0	91.8	89.3
5-texture-4	96.7	94.1	96.9	97.5	85.6
10-texture-1	95.7	71.4	95.3	96.0	87.7
10-texture-2	90.0	71.3	88.5	88.0	81.1
16-texture-1	93.3	74.2	91.2	90.4	85.6
16-texture-2	81.3	52.6	81.1	79.2	79.2

In the packet texture classification experiment, the KLRM-DL proposed in this paper are compared with the

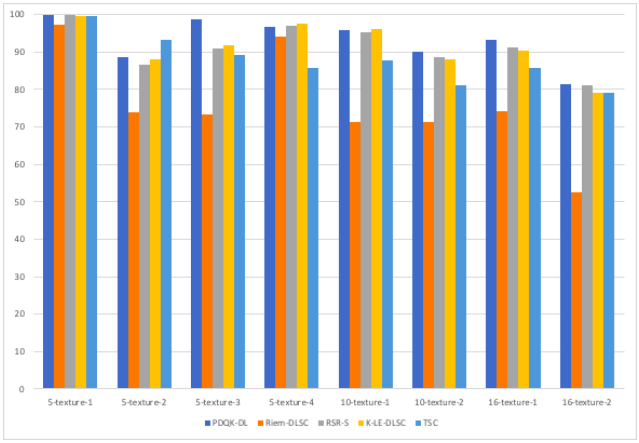

FIGURE 5. Comparison of algorithms for 1-NN classification accuracy in Brodatz dataset.

existing sparse coding methods on SPD manifold. The comparison methods are TSC [31], RSR-S [16], K-LE-DLSC [14] and Riem-DLSC [1].

It can be seen from Table 4 that the KLRM-DL proposed in this chapter have better performance on Brodatz dataset than existing SPD manifold sparse coding methods. The classification accuracy of KLRM-DL is the highest in most groups, and the classification accuracy of Riem-DLSC directly modeled on SPD manifold is lower than that of SPD manifold converted to RKHS. In grouping experiments with a large number of categories, the classification accuracy rate of methods such as RSR-S using the Stein kernel function and K-LE-DLSC-Gaussian using the LE kernel function is lower than that of grouping experiments with a small number of categories. However, the KLRM-DL method proposed in this chapter can still maintain a high classification accuracy rate when the number of categories is increased. On the 16-texture-1 group, KLRM-DL is 2.1% higher than the highest value of the other methods. KLRM-DL on the 16-texture-2 group2 is similar to the highest value of the existing method, and KLRM-DL can be considered to be effective.

In the classification experiments of all textures, the experimental setup is consistent with [1]. There are 112 texture images in the Brodat dataset. Each image represents a type of texture. In all texture classification experiments, all texture images will be used for experiments. The way to generate SPD data for each texture picture is consistent with the previous grouping experiments with selected textures.

In the experiment, 14 SPD matrices are selected as data samples for each type of texture, so the total number of data samples is 1568, the number of data samples actually used is 400, and the data samples are fixed, leaving 50 for each type of texture. We randomly select 20 SPD matrices as the training set learning dictionary and kernel function, randomly select 20 SPD matrices as the reference set for the classifier from the remaining 30 SPD matrices, and use the remaining 10 as the test set to calculate the classification accuracy. Using the learned dictionary to encode all SPD matrices in the reference set and test set, and then use the 1-NN classifier to calculate the classification accuracy rate.

TABLE 5. Accuracy of all texture classifications of Brodatz (%).

Method	1-NN
Riem-DLSC	74.9
Fro-DLSC	23.5
K-LE-DLSC	47.9
RSR-S	76.7
TSC	37.1
GDL	47.7
KLRM-DL	77.0

The method proposed in this chapter is compared with the existing sparse coding methods on SPD manifold in the experiments on the classification of all texture pictures in the Brodatz dataset. The comparison methods are TSC [31], RSR-S [15], K-LE -DLSC [14], Riem-DLSC [37], and GDL [33]. The classification accuracy of each method on the Brodatz dataset is shown in Table 5. As can be seen from the table above, the KLRM-DL method achieves the highest classification accuracy rate when all texture pictures are classified, which is 0.3% higher than the highest value RSR-S in the other methods, which is 2.1% better than building directly on SPD manifold Rime-DLSC method. In addition, the KLRM-DL method is much higher than the Fro-DLSC method directly modeled in symmetric matrix space and the K-LE-DLSC method using the LE kernel function. It can be seen that the KLRM-DL method is still effective in multi-class classification.

D. EXPERIMENTS ON THE FERET DATASET

In this section, a face recognition experiment is performed on the FERET dataset [53]. The experiment uses the 'b' subset of the FERET dataset. The 'b' subset contains a total of 2000 face pictures from 200 people. The direction deviation angle of the face is divided into 10 directions 'ba', 'bb', 'bc', 'bd', 'be', 'bf', 'bg', 'bh', 'bi', 'bk'. The picture is a grayscale image, the original size is 256 × 384. For each picture, the area is first extracted where the face is located, and then down-sampled to 64 × 64. Some sample pictures can be seen in Fig. 6.

FIGURE 6. Partial picture display of the 'b' subset of the Feret dataset.

In the experiment, in order to maintain the experimental setting consistent with [17], the pictures in the direction of

'ba', 'bc', 'bh', 'bk' are used as the training set. We are not need dictionary learning is performed that is because the training set pictures have category labels and the dictionary is labeled. The pictures in the direction of 'bd', 'be', 'bf', 'bg' are the test set. The data depends on the sample points randomly selected from the training set. The experimental dictionary on FERET has been fixed and has class labels, and dictionary learning is not required. Therefore, the experiments proposed by the algorithm on the FERET dataset only use the kernel learning strategy. In the experiment, each picture can be represented by an SPD matrix of size 43×43. The generation of eigenvectors follows the following formula:

$$f(x, y) = [I(x, y), x, y, |G_{0,0}(x, y)|, \cdots, |G_{4,7}(x, y)|] \quad (63)$$

Among them, (x, y) is the position coordinate, $I(x, y)$ is the gray value of the position, and $G_{u,v}(x, y)$ is the response value of the two-dimensional wavelet filter (Gabor Filter) at the position (x, y). The direction u of the wavelet filter $G_{u,v}$ is from 0 to 4, and the scale v is from 0 to 7. There are 40 wavelet filters in total.

Calculate the encoding result of the test set, and then use the sparse encoding-based classification method described to calculate the recognition accuracy rate. The results are shown in Table 6.

TABLE 6. Correct rate of face recognition in Feret dataset (%).

method	bd	be	bf	bg	average
SRC	27.5	55.5	61.0	26.0	42.5
GSRC	77.0	93.5	97.0	79.0	86.6
LogEuc-SC	74.0	94.0	97.5	80.5	86.5
TSC	36.0	73.0	73.5	44.5	56.8
RSR-S	82.5	94.5	98.0	83.5	89.6
RSR-J	79.5	96.5	97.5	86.0	89.9
KLRM-DL	89.5	96.0	97.0	94.0	94.1

The algorithm proposed in this paper is compiled and run on the Matlab platform. The computer processor used is Intel's 8th-generation 8-core processor i5-8265U. In 20 experiments, the average sample training time of the code is 706.54s and the classification test time is 5.68s. It can be seen that the training and testing time of the algorithm is relatively short. And it is compared with the existing sparse coding methods on SPD manifold. The comparison methods are TSC [31], LogEuc-SC [20], SRC [2], GSRC [3] and RSR [17]. It can be seen from Table 6 that the KLRM-DL method proposed in this chapter has a higher face recognition accuracy rate in the 'bd' and 'bg' subsets of the Feret dataset than the comparison algorithm. The recognition accuracy rate on the 'be' subset differs from the highest value by 6.5%, and the recognition accuracy rate on the 'bf' subset differs from the highest value by 4.5%, both of which are not much different. It can be seen that the performance of KLRM-DL on each subset is more stable and is not affected by the direction of the face picture. The performance on the set is more stable and is not affected by the orientation of the face image. In addition, KLRM-DL's average recognition accuracy rate in Feret's face set is much higher than the existing sparse coding method on SPD manifold, which is 0.9% higher than the second one. It can be considered that KLRM-DL is effective.

VII. CONCLUSION

RKHS is a common mathematical platform for various kernel methods in machine learning. The learning mechanism of kernel approach is to embed the original data space into RKHS via kernel function, where the Euclidean computations apply. The so-called kernel learning is to choose the most suitable RKHS according to the specific application of machine learning (dictionary learning, transfer learning, etc.) and a given learning sample. Since RKHS can only be generated by kernel functions, kernel learning is also kernel function learning. The current dilemma of kernel learning is that there are not many types of kernel functions that can be learned, and there are not many parameters that can be learned in these kernel functions, such as exponential parameters in Gaussian kernel functions, or combination coefficients in multi-kernel learning, and so on. The first contribution of this paper is to propose a kernel framework that calls dictionary learning based on kernel learning and Riemannian metric, where the SPD matrix in the positive definite quadratic form is the learnable part of KLRM-DL and Riemannian metric can really represent the true distance between two points.

At present, the use of regional covariance descriptors to generate SPD matrix feature data from images or videos is increasingly common in machine learning. However, SPD data itself does not constitute a linear space, and most machine learning algorithms (such as dictionary learning algorithms) involve a large number of linear operations. At present, a common method is to transform the SPD data to RKHS and perform machine learning in RKHS. The second contribution of this paper is to apply the KLRM-DL proposed in this paper to dictionary learning of SPD data. This article first transforms the given SPD learning samples to the RKHS generated by KLRM-DL, and then learns KLRM-DL and dictionary simultaneously on RKHS according to the dictionary learning criteria. The RKHS learned in this way provides the most suitable working space for dictionary applications (such as dictionary coding and dictionary-based recognition, etc.).

REFERENCES

[1] A. Cherian and S. Sra, "Riemannian dictionary learning and sparse coding for positive definite matrices," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 12, pp. 2859–2871, Dec. 2017.
[2] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 2, pp. 210–227, Feb. 2009.
[3] M. Yang and L. Zhang, "Gabor feature based sparse representation for face recognition with Gabor occlusion dictionary," in *Proc. Eur. Conf. Comput. Vis.*, 2010, pp. 448–461.
[4] S. Gao, I. W.-H. Tsang, and L.-T. Chia, "Laplacian sparse coding, hypergraph Laplacian sparse coding, and applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 92–104, Jan. 2013.
[5] S. Jayasumana, R. Hartley, M. Salzmann, H. Li, and M. Harandi, "Kernel methods on the Riemannian manifold of symmetric positive definite matrices," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 73–80.

[6] A. Cherian, S. Sra, A. Banerjee, and N. Papanikolopoulos, "Jensen-Bregman LogDet divergence with application to efficient similarity search for covariance matrices," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 9, pp. 2161–2174, Sep. 2013.
[7] M. Harandi, C. Sanderson, C. Shen, and B. Lovell, "Dictionary learning and sparse coding on Grassmann manifolds: An extrinsic solution," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 3120–3127.
[8] H. E. Cetingul and R. Vidal, "Intrinsic mean shift for clustering on Stiefel and Grassmann manifolds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 1896–1902.
[9] X. Pennec, P. Fillard, and N. Ayache, "A Riemannian framework for tensor computing," *Int. J. Comput. Vis.*, vol. 66, no. 1, pp. 41–66, Jan. 2006.
[10] M. Harandi, M. Salzmann, and R. Hartley, "Dimensionality reduction on SPD manifolds: The emergence of geometry-aware methods," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 1, pp. 48–62, Jan. 2018.
[11] Y. Peng, N. Zhang, Y. Li, and S. Ying, "A Local-to-Global metric learning framework from the geometric insight," *IEEE Access*, vol. 8, pp. 16953–16964, 2020.
[12] 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. 32nd Int. Conf. Mach. Learn.*, vol. 37, 2015, pp. 1–10.
[13] M. Harandi, C. Sanderson, R. Hartley, and B. Lovell, "Sparse coding and dictionary learning for symmetric positive definite matrices: A kernel approach," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 216–229.
[14] R. Caseiro, J. Henriques, P. Martins, and J. Batista, "Semi-intrinsic mean shift on Riemannian manifolds," in *Proc. European Conf. Comput. Vis.*, 2012, pp. 342–355.
[15] J. Shawetaylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. New York, NY, USA: Cambridge Univ. Press, 2004.
[16] P. Li, Q. Wang, W. Zuo, and L. Zhang, "Log-Euclidean kernels for sparse representation and dictionary learning," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 1601–1608.
[17] M. T. Harandi, R. Hartley, B. Lovell, and C. Sanderson, "Sparse coding on symmetric positive definite manifolds using bregman divergences," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 6, pp. 1294–1306, Jun. 2016.
[18] M. Harandi, C. Sanderson, C. Shen, and B. Lovell, "Dictionary learning and sparse coding on Grassmann manifolds: An extrinsic solution," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 251–255.
[19] S. Said, L. Bombrun, Y. Berthoumieu, and J. H. Manton, "Riemannian Gaussian distributions on the space of symmetric positive definite matrices," *IEEE Trans. Inf. Theory*, vol. 63, no. 4, pp. 2153–2170, Apr. 2017.
[20] K. Guo, P. Ishwar, and J. Konrad, "Action recognition from video using feature covariance matrices," *IEEE Trans. Image Process.*, vol. 22, no. 6, pp. 2479–2494, Jun. 2013.
[21] A. Alavi, A. Wiliem, K. Zhao, B. C. Lovell, and C. Sanderson, "Random projections on manifolds of symmetric positive definite matrices for image classification," in *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, Mar. 2014, pp. 301–308.
[22] R. Kondor and J. Lafferty, "Diffusion kernels on graphs and other discrete input spaces," in *Proc. 19th Int. Conf. Mach. Learn.*, 2002, pp. 315–322.
[23] J. Lafferty and G. Lebanon, "Diffusion kernels on statistical manifolds," *J. Mach. Learn. Res.*, vol. 6, pp. 129–163, Jan. 2005.
[24] A. J. Smola and R. Kondor, "Kernels and regularization on graphs," in *Proc. 16th Annu. Conf. Comput. Learn. Theory Kernel Workshop*, 2003, pp. 144–158.
[25] J. B. Tenenbaum, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, Dec. 2000.
[26] S. T. Roweis, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, Dec. 2000.
[27] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Comput.*, vol. 15, no. 6, pp. 1373–1396, Jun. 2003.
[28] J. Ham, D. D. Lee, S. Mika, and B. Schölkopf, "A kernel view of the dimensionality reduction of manifolds," in *Proc. 21st Int. Conf. Mach. Learn. (ICML)*, Banff, AB, Canada, 2004, pp. 47–54.
[29] Y. Bengio and P. Vincent, "Out-of-sample extensions for LLE, Isomap, MDS, Eigenmaps, and spectral clustering," in *Proc. Int. Conf. Mach. Learn.*, 2005, pp. 824–831.
[30] V. Sindhwani, P. Niyogi, and M. Belkin, "Beyond the point cloud: From transductive to semi-supervised learning," in *Proc. 22nd Int. Conf. Neural Inf. Process. Syst.* Cambridge, MA, USA: MIT Press, 2004, pp. 177–184.
[31] R. Sivalingam, D. Boley, V. Morellas, and N. Papanikolopoulos, "Tensor sparse coding for positive definite matrices," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 3, pp. 592–605, Mar. 2014.
[32] R. Sivalingam, D. Boley, V. Morellas, and N. Papanikolopoulos, "Tensor dictionary learning for positive definite matrices," *IEEE Trans. Image Process.*, vol. 24, no. 11, pp. 4592–4601, Nov. 2015.
[33] S. Sra and A. Cherian, "Generalized dictionary learning for symmetric positive definite matrices with application to nearest neighbor retrieval," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*, 2011, pp. 318–332.
[34] A. Cherian and S. Sra, "Riemannian sparse coding for positive definite matrices," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 299–314.
[35] X. Zhang, W. Li, W. Hu, H. Ling, and S. Maybank, "Block covariance based l1 tracker with a subtle template dictionary," *Pattern Recognit.*, vol. 46, no. 7, pp. 1750–1761, Jul. 2013.
[36] K. Guo, P. Ishwar, and J. Konrad, "Action recognition using sparse representation on covariance manifolds of optical flow," in *Proc. 7th IEEE Int. Conf. Adv. Video Signal Based Surveill.*, Aug. 2010, pp. 188–195.
[37] C. Yuan, W. Hu, and L. X. and, "Human action recognition under log-euclidean Riemannian metric," in *Proc. Asian Conf. Comput. Vis.*, 2010, pp. 343–353.
[38] S. Zhang, S. Kasiviswanathan, P. Yuen, and H. M. and, "Online dictionary learning on symmetric positive definite manifolds with vision applications," in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 1–9.
[39] A. Barachant, S. Bonnet, M. Congedo, and C. Jutten, "Classification of covariance matrices using a riemannian-based kernel for BCI applications," *Neurocomputing*, vol. 112, pp. 172–178, Jul. 2013.
[40] J. Zhang, L. Wang, L. Zhou, and W. Li, "Learning discriminative stein kernel for SPD matrices and its applications," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 5, pp. 1020–1033, May 2016.
[41] A. Das, M. S. Nair, and S. D. Peter, "Sparse representation over learned dictionaries on the Riemannian manifold for automated grading of nuclear pleomorphism in breast cancer," *IEEE Trans. Image Process.*, vol. 28, no. 3, pp. 1248–1260, Mar. 2019.
[42] Z. Gao, Y. Wu, M. Harandi, and Y. Jia, "A robust distance measure for similarity-based classification on the SPD manifold," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Sep. 27, 2019, doi: 10.1109/TNNLS.2019.2939177.
[43] Y. Wu, Y. Jia, P. Li, J. Zhang, and J. Yuan, "Manifold kernel sparse representation of symmetric positive-definite matrices and its applications," *IEEE Trans. Image Process.*, vol. 24, no. 11, pp. 3729–3741, Nov. 2015.
[44] [Online]. Available: http://spams-devel.gforge.inria.fr/
[45] [Online]. Available: http://cvxr.com/cvx/
[46] P. A. Absil, C. G. Baker, and K. A. Gallivan, "Optimization algorithms on matrix manifolds," in *Trust-Region Methods on Riemannian Manifolds*, vol. 7, no. 3. 2007, pp. 303–330.
[47] P. A. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*. Princeton, NJ, USA: Princeton Univ. Press, 2008.
[48] D. P. Bertsekas, *Nonlinear Programming*. New York, NY, USA: Athena Scientific, 1999.
[49] S. Fiori, "Extended Hamiltonian learning on Riemannian manifolds: Theoretical aspects," *IEEE Trans. Neural Netw.*, vol. 22, no. 5, pp. 687–700, May 2011.
[50] [Online]. Available: https://www.manopt.org/
[51] D. Tosato, M. Spera, M. Cristani, and V. Murino, "Characterizing humans on Riemannian manifolds," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1972–1984, Aug. 2013.
[52] T. Randen and J. H. Husoy, "Filtering for texture classification: A comparative study," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 4, pp. 291–310, Apr. 1999.
[53] P. J. Phillips, H. Moon, S. A. Rizvi, and P. J. Rauss, "The FERET evaluation methodology for face-recognition algorithms," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 10, pp. 1090–1104, 2000.

RIXIN ZHUANG received the B.Sc. degree from the Guangdong University of Technology, Guangzhou, China, in 2017. He is currently pursuing the master's degree with the School of Electronics and Information Technology, Sun Yat-sen University, Guangzhou. His research interest is machine learning.

ZHENGMING MA received the B.Sc. degree in radio technology and the M.Sc. degree in electronic and communication system from the South China University of Technology, Guangzhou, China, in 1982 and 1985, respectively, and the Ph.D. degree in pattern recognition and intelligent control from Tsinghua University, Beijing, China, in 1989. He is currently a Professor with the School of Electronics and Information Technology, Sun Yat-sen University, Guangzhou. His current research interests include machine learning.

WEIJIA FENG received the B.Sc. degree from the Guangdong University of Technology, Guangzhou, China, in 2019. He is currently pursuing the master's degree with the School of Electronics and Information Technology, Sun Yat-sen University, Guangzhou. His research interest is machine learning.

YUANPING LIN received the B.S. degree in electronic information engineering from South China Agricultural University, Guangzhou, China, in 2017. He is currently pursuing the master's degree with the School of Electronics and Information Technology, Sun Yat-sen University, Guangzhou. His current research interests include machine learning and kernel learning.