Learning Log-Determinant Divergences for Positive Definite Matrices

Anoop Cherian* Panagiotis Stanitsas* Jue Wang Mehrtash Harandi
Vassilios Morellas Nikolaos Papanikolopoulos

Abstract—Representations in the form of Symmetric Positive Definite (SPD) matrices have been popularized in a variety of visual learning applications due to their demonstrated ability to capture rich second-order statistics of visual data. There exist several similarity measures for comparing SPD matrices with documented benefits. However, selecting an appropriate measure for a given problem remains a challenge and in most cases, is the result of a trial-and-error process. In this paper, we propose to learn similarity measures in a data-driven manner. To this end, we capitalize on the $\alpha\beta$-log-det divergence, which is a meta-divergence parametrized by scalars α and β, subsuming a wide family of popular information divergences on SPD matrices for distinct and discrete values of these parameters. Our key idea is to cast these parameters in a continuum and learn them from data. We systematically extend this idea to learn vector-valued parameters, thereby increasing the expressiveness of the underlying non-linear measure. We conjoin the divergence learning problem with several standard tasks in machine learning, including supervised discriminative dictionary learning and unsupervised SPD matrix clustering. We present Riemannian gradient descent schemes for optimizing our formulations efficiently, and show the usefulness of our method on eight standard computer vision tasks.

Index Terms—region covariance matrices, positive definite matrices, log-det divergence, action recognition, texture recognition.

1 Introduction

The trail of Symmetric Positive Definite (SPD) matrices in computer vision applications is becoming more prominent in the recent years. Examples include, flexible image representations derived in the form of Region CoVariance Descriptors (RCoVDs) [1] capable of fusing various modalities while compactly capturing their second order statistics. Popular deep learning architectures have also benefited by the use of SPD matrices employing them as second-order pooling operators [2], [3], [4], [5]. In an effort to harness the representational power of SPD matrices, the newly devised domain of geometric deep learning has produced powerful machinery for addressing inference problems in a Riemannian deep learning setup [5]. More classical approaches promote the use of SPD matrices as kernel matrices of high-dimensional data [6], points in diffusion MRI [7], and diffusion tensors [8].

- *Anoop Cherian* is with Mitsubishi Electric Research Labs (MERL), Cambridge, MA. Work done while at the Australian Centre for Robotic Vision, Australian National University, Canberra. The * indicates equal contribution. E-mail: cherian@merl.com*
- *Panagiotis Stanitsas* is with the University of Minnesota, Minneapolis, MN. E-mail: stani078@umn.edu*
- *Jue Wang is with the Research School of Engineering, The Australian National University, ACT 2601, Australia. E-mail: jue.wang@anu.edu.au*
- *Mehrtash Harandi is with the department of Electrical and Computer Systems Engineering, Monash University, and Data61-CSIRO, Melbourne, Australia. E-mail: mehrtash.harandi@monash.edu*
- *Vassilios Morellas is with the University of Minnesota, Minneapolis, MN. E-mail: morellas@cs.umn.edu*
- *Nikolaos Papanikolopoulos is with the University of Minnesota, Minneapolis, MN. E-mail: npapas@cs.umn.edu*

SPD matrices belong to an open cone in the Euclidean space as a result of their positive definiteness property, and thus a natural way to compare two SPD matrices is the standard Euclidean distance. However, it is often found practically that using a Euclidean geometry leads to sub-optimal performance. A motivating application in this regard is perhaps the problem of computing the barycenter of a set of SPD matrices [9], [10]; a common task arising in ensemble learning of Gaussian models [11] or when interpolating DT-MRI points (which are SPD matrices) [7], [12]. It is often found that using a Euclidean geometry in these applications lead to unrealistic barycenters (often producing bloated SPD matrices, which may be interpreted as an overestimated covariance in Gaussian models). However, enforcing a non-linear (often Riemannian) geometry on the SPD cone using a suitable non-linear measure avoids such pitfalls, leading to practical benefits, as demonstrated in several successful applications [2], [3], [13], [14], [15], [16].

Interestingly, SPD matrices constitute a very rich class of mathematical objects that appear in several disciplines, and for which a wide variety of similarity measures and geometries have been associated with. Notable representatives of such measures include: (i) the Affine Invariant Riemannian Metric (AIRM) which is a geodesic distance induced by their natural Riemannian geometry [7], (ii) the Jensen-Bregman [17], [18], [19] and Burg [20] divergences which result from information geometry perspective and, (iii) the Jeffreys KL divergence (KLDM) using relative entropy [21], among several others such as the Bures distance used in quantum information theory [22], Bures-Wasserstein distance [23] recently proposed in optimal transport theory, the popular log-Euclidean Riemannian metric used in diffusion MRI [12] and its kernel analogues such as the Hilbert-Schmidt metric [24]. Given such an elaborate choice of potential measures, it is overwhelming to consider choosing an appropriate similarity for a given problem.

In this paper, we attempt to tackle the challenge of choosing

the similarity measure using data-driven approach by directly learning the divergence from a training set of SPD matrices. While, one may resort to standard metric learning [25] ideas or their SPD matrix variants [13], [26], [27], [28] in this regard, we propose to use the recently introduced $\alpha\beta$-Logdet Divergence (ABLD) [29] for this purpose. ABLD is a meta-divergence on SPD matrices, parametrized by scalars α and β, and which is shown to be exactly equal to some of the popular divergences and metrics listed above for specific values of these scalars. For example, ABLD converges to AIRM as $\alpha, \beta \rightarrow 0$ and is equal to Burg divergence when $\alpha = \beta = 1$, among others (see Sec. 3). This unifying result suggests that we could learn to choose an appropriate similarity measure in a data-driven way by learning α and β. In contrast to standard metric learning approaches, our proposed learning setup not only allows learning a suitable measure from a continuum of log-det divergences, but also guarantees that standard measures are included in the learning landscape.

While, learning ABLD is our primary motivation, we go far beyond this goal and generalizes our learning setup in two predominant ways, (i) a dictionary learning and divergence learning setting for supervised regression and classification, and (ii) divergence learning combined with unsupervised clustering in a K-Means setting. Specifically, instead of learning the parameters of a single ABLD, we propose to learn a family of ABLDs, each parametrized separately, via learning a vector of parameter pairs under the assumption that there might be distinct SPD matrix subspaces (clusters or classes) in the given data that may adhere to their own divergences. We further parameterize the origin in these subspaces via learning them; our full setup we call *Information Divergence and Dictionary Learning* (IDDL), where each origin forms an atom in an SPD matrix dictionary. Using this setup, we propose SPD data regression and classification models by embedding a given SPD matrix into a vector; each dimension of this vector capturing its similarity to a respective dictionary atom via its learned divergence. We combine IDDL with i) a ridge-regression objective, and ii) using a structured-SVM objective. Our full model (including IDDL and classifier) are learned end-to-end in a Riemannian alternating minimization setup. For clustering, we derive a scheme for K-Means on SPD matrices while addressing the problem of finding optimal values of α and β. This effort yields a variant of K-Means, which we call $\alpha\beta$-KMeans.

To evaluate our frameworks, we present experiments on an extensive set of computer vision applications, including texture recognition, activity recognition, 3D object, and cancerous tissue recognition, on a multitude of datasets with different levels of data complexities. We supplement our experimental comparisons with an equally extensive ablation study of our schemes under various settings. Our results demonstrate the benefits of joint learning by achieving state-of-the-art performance against competing techniques, including the recent sparse coding, Riemannian metric learning, and kernel coding schemes.

This paper extends our previous works on this topic [30], [31], which consider the IDDL and AB-KMeans clustering setups separately. In this paper, we not only unifies these setups, but also explores more richer classification settings via a structured-SVM formulation, which is jointly learned with our IDDL framework. We also provide additional technical details of our optimization. Extensive discussion is provided in the form of a detailed ablation study discussing all the different elements of our scheme.

2 Related Work

The $\alpha\beta$-logdet divergence is a matrix generalization of the well-known $\alpha\beta$-divergence [32] that computes the (a)symmetric (dis)similarity between two finite positive measures (data densities). As the name implies, $\alpha\beta$-divergence is a unification of the so-called α-family of divergences [33] (that includes popular measures such as the KL-divergence, Jensen-Shannon divergence, and the chi-square divergence) and the β-family [34] (including the squared Euclidean distance and the Itakura Saito distance). Against several standard measures for computing similarities, both α and β divergences are known to lead to solutions that are robust to outliers and additive noise [35], thereby improving application performance. They have been used in several statistical learning applications including non-negative matrix factorization [36], [37], [38], nearest neighbor embedding [39], and blind-source separation [40].

A class of methods with similarities to our formulation are metric learning schemes on SPD matrices. One popular technique is the manifold-manifold embedding of large SPD matrices into a lower-dimensional SPD space in a discriminative setting [13]. Log-Euclidean metric learning has also been proposed for this embedding in [27], [41]. While, we also learn a metric in a discriminative setup, ours is different in that we learn an information divergence. In Thiyam et al. [42], ABLD is proposed replacing symmetric KL divergence in better characterizing the learning of a decision hyperplane for BCI applications. In contrast, we propose to embed the data matrices as vectors, each dimension of these vectors learning a different ABLD, thus leading to a richer representation of the input matrix. More recently, extensions of ABLD to an infinite dimensional Hilbert space setting is explored in [43]. However, our work is complementary to this effort, and explores a finite-dimensional setting.

Vectorial embedding of SPD matrices has been investigated using disparate formulations for computer vision applications. As alluded to earlier, the log-Euclidean projection [12] is a common way to achieve this, where an SPD matrix is isomorphically mapped to the Euclidean space of symmetric matrices using the matrix logarithm. Popular sparse coding schemes have been extended to SPD matrices in [44], [45], [46] using SPD dictionaries, where the resulting sparse vector is assumed Euclidean. Another popular way to handle the non-linear geometry of SPD matrices is to resort to kernel schemes by embedding the matrices in an infinite dimensional Hilbert space which is assumed to be linear [6], [47], [48]. In all these methods, the underlying similarity measure is fixed and is usually chosen to be one among the popular $\alpha\beta$-logdet divergences or the log-Euclidean metric.

Several unsupervised schemes for clustering SPD matrices have also been proposed in the relevant literature. Commonly used schemes capitalize on conventional clustering machinery after being modified towards abiding to the non-linear geometry of SPD matrices. In this direction, two extensions of the popular KMeans have been derived admitting the manifold of the SPD matrices. In the first variant of KMeans, centroids are computed using the Karcher means algorithm [10] and the affine-invariant Riemannian metric [7]. Substituting the similarity computation based on the AIRM by the log-Euclidean metric [12], yields a second variant of KMeans for SPD matrices termed LE-KMeans. Using the matrix logarithm operation, which entails a diffeomorphic mapping of an SPD matrix onto its tangent space, allows for distance computations in a Euclidean manner (as this tangent space is Euclidean). In that way, centroids are computed by averaging the

(α, β)	ABLD	Divergence
$(\alpha, \beta) \to 0$	$\left\|\operatorname{Log} X^{-\frac{1}{2}} Y X^{-\frac{1}{2}}\right\|_F^2$	Squared Affine Invariant Riemannian Metric (AIRM) [7]
$\alpha = \beta = \pm\frac{1}{2}$	$4\left(\log\det\frac{X+Y}{2} - \frac{1}{2}\log\det XY\right)$	Jensen-Bregman Logdet Divergence (JBLD) [17]
$\alpha = \pm 1, \beta \to 0$	$\frac{1}{2}\operatorname{Tr}\left(XY^{-1} + YX^{-1}\right) - d$	Jeffreys KL Divergence[1] [21]
$\alpha = 1, \beta = 1$	$\operatorname{Tr}\left(XY^{-1}\right) - \log\det XY^{-1} - d$	Burg Matrix Divergence [20]

TABLE 1: ABLD and its connections to popular divergences.

samples' vectorial representations in the tangent space. Additional variants of KMeans can be derived by capitalizing on the different similarity measures for SPD matrices.

A second family of clustering schemes for SPD matrices takes advantage of Euclidean embeddings in the form of similarity matrices computed using suitable measures. In that direction, Spectral clustering schemes have been developed for SPD matrices by computing suitable Mercer kernels on the data using appropriate distances (e.g., LE). Sparse subspace clustering schemes have also been derived for SPD matrices via their embedding into a Reproducing Kernel Hilbert Space [49], [50]. Such schemes come at the expense of additional memory requirements involved with computing the eigen spectrum of the computed kernel.

In addition, non-parametric schemes for clustering SPD matrices have been derived in the form of dimensionality reduction on Riemannian manifolds [51], using Locally Linear Embeddings [52], or capitalizing on the Laplacian eigenmaps [53]. In the family of non-parametric clustering algorithms, a Bayesian framework for SPD matrices is formulated using the Dirichlet Process [54]. Finally, variants of the Mean shift clustering algorithm and Kernel Density Estimation for SPD matrices have also been derived in [55] and [56] respectively.

In contrast to all these methods, to the best of our knowledge, this is the first unified attempt to bridge the learning of information divergences with dictionary learning and clustering objectives. Even though automatic selection of the parameters of $\alpha\beta$-divergence is investigated in [57], [58] they deal solely with scalar density functions in a maximum-likelihood setup and do not consider the optimization of α and β jointly.

Notation: Following standard practice, we use upper case for matrices (such as X), lower-bold case for vectors $\mathbf{x}$, and lower case for scalars x. Further, $\mathcal{S}_{++}^d$ is used to denote the cone of $d \times d$ SPD matrices. We use $\mathbf{B}$ to denote a 3D tensor each slice of which is an SPD matrix of size $d \times d$. Further, we use I_d to denote the $d \times d$ identity matrix, Log for the matrix logarithm, and diag for the diagonalization operator. In addition, we use $\mathbf{C}$ to denote a 3D tensor each slice of which corresponds to an SPD centroid of size $d \times d$ for the proposed clustering setup. Finally, we use $\mathbf{\Pi} = \{\pi_1, ..., \pi_k\}$ to denote a clustering of data into k partitions; π_i is the i-th partition and comprises a subset of the dataset assigned to this cluster.

3 Background

In this section, we setup the mathematical preliminaries necessary to elucidate our contributions.

3.1 Alpha-Beta-Log-Determinant Divergence (ABLD)

Introduced in Cichocki et al. [29], the $\alpha\beta$-log-det divergence is a result of an effort to extend the log-det divergences, existing in various forms [7], [17], [21], and finding connections between them. Below, we review this meta-divergence, its connections to other divergences, and some of its properties that will be useful in our formulations.

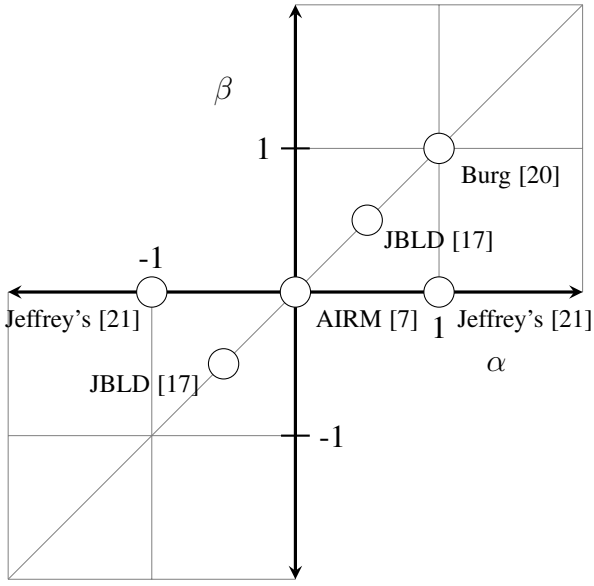


Fig. 1: An illustration of ABLD and its connections to popular divergences on SPD matrices.

Definition 1 (ABLD [29]). For $X, Y \in \mathcal{S}_{++}^d$, the $\alpha\beta$-log-det divergence is defined as:

$$D^{(\alpha,\beta)}(X\|Y) = \frac{1}{\alpha\beta}\log\det\left(\frac{\alpha(XY^{-1})^{\beta} + \beta(XY^{-1})^{-\alpha}}{\alpha+\beta}\right), \tag{1}$$

$$\alpha \neq 0,\ \beta \neq 0 \ \text{ and } \alpha + \beta \neq 0. \tag{2}$$

It can be shown that ABLD depends only on the generalized eigenvalues of X and Y [29]. Suppose λ_i denotes the i-th eigenvalue of XY^{-1}. Then under constraints defined in (2), we can rewrite (1) as:

$$D^{(\alpha,\beta)}(X\|Y) = \frac{1}{\alpha\beta}\sum_{i=1}^{d}\log\left(\alpha\lambda_i^{\beta} + \beta\lambda_i^{-\alpha}\right) - d\log(\alpha+\beta). \tag{3}$$

This formulation will come handy when deriving the gradient updates for α and β in the sequel. As alluded to earlier, a hallmark of ABLD is that it unifies several popular distance measures on SPD matrices that one commonly encounters in machine learning and vision applications. In Table 1, we provide connections of ABLD with popular similarity measures on SPD matrices. A graphical illustration of the landscape of these measures is provided in Figure 1 for various values of α and β.

3.2 ABLD Properties

Avoiding Degeneracy: An important observation regarding the design of optimization algorithms on ABLD is that the quantity

inside the $\log\det$ term has to be positive definite. are simultaneously positive or negative, this qua definite. However, when α and β have different s term may become degenerate, the necessary cor which are stipulated by the following theorem.

Theorem 1. For $X, Y \in \mathcal{S}_{++}^d$, if λ_i is the i-t XY^{-1}, then $D^{(\alpha,\beta)}(X \parallel Y) \geq 0$ only if

$$\lambda_i > \left|\frac{\alpha}{\beta}\right|^{\frac{1}{\alpha+\beta}}, \text{ for } \alpha > 0 \text{ and } \beta < 0, \text{ or}$$

$$\lambda_i < \left|\frac{\beta}{\alpha}\right|^{\frac{1}{\alpha+\beta}}, \text{ for } \alpha < 0 \text{ and } \beta > 0, \forall i =$$

Proof See [29].

As the algorithms we introduce in the sequel u SPD matrices, it may be computationally challenging to impose the restrictions in Theorem 1 on the learned α and β for every matrix pair. Thus, we restrict our scope in this paper to the form of ABLD when both α and β have the same sign, thereby avoiding the quantity inside $\log\det$ to become indefinite. We make this assumption in our formulations in Section 4.

Affine Invariance: For a non-singular matrix $\mathbf{A} \in \mathbb{R}^{d\times d}$, ABLD is invariant under congruent transformations:

$$D^{(\alpha,\beta)}(X \parallel Y) = D^{(\alpha,\beta)}(AXA^\top \parallel AYA^\top). \tag{6}$$

This is an important property that makes this divergence useful in a variety of applications, e.g., diffusion MRI [59], where affine-invariance allows the similarity to be robust to anatomical changes in the subjects or to configurations of the image acquisition hardware. Assuming affine invariance also helps derive a metric in a Riemmanian symmetric space; this metric is often found to be better in restoration, interpolation, and filtering of DTMRI images [7].

Identity of Indiscernibles: For any α, β, it holds that

$$D^{(\alpha,\beta)}(X \parallel Y) = 0 \text{ if and only if } X = Y. \tag{7}$$

Scaling Invariance: For any $c > 0$, it is easy to show that

$$D^{(\alpha,\beta)}(cX \parallel cY) = D^{(\alpha,\beta)}(X \parallel Y). \tag{8}$$

Smoothness of α, β: Assuming α, β have the same sign, except at the origin ($\alpha = \beta = 0$), ABLD is smooth everywhere with respect to α and β, thus allowing us to develop Newton-type algorithms on them. Due to the discontinuity at the origin, we ought to design algorithms specifically addressing this particular case.

Dual Symmetry: This property allows us to extend results derived for the case of α to the one on β later.

$$D^{(\alpha,\beta)}(X \parallel Y) = D^{(\beta,\alpha)}(Y \parallel X). \tag{9}$$

4 Proposed Method

We start by introducing our *Information Divergence and Dictionary Learning* (IDDL) problem setup in which a discriminative divergence learning framework is combined with a dictionary learning setting. Specifically, instead of using a single divergence measure on the SPD matrices, we present a very general setting in which we assume there are SPD subspaces in the data, which may be characterized via learning separate ABLD measures, each parametrized distinctly. We further generalize this model into a bag-of-words setting, by combining our multi-divergence learning problem with an SPD dictionary learning problem, where we also assume that each divergence also has its own SPD subspace origin with respect to which the divergence is measured. We include our IDDL framework within a joint classification setup. We explore two variants of the classification model: i) using a ridge-regression loss, and ii) using a structured SVM loss. Next, we present divergence learning within a K-Means clustering setup, where alongside learning the ABLD parameters, we also learn the SPD cluster centroids.

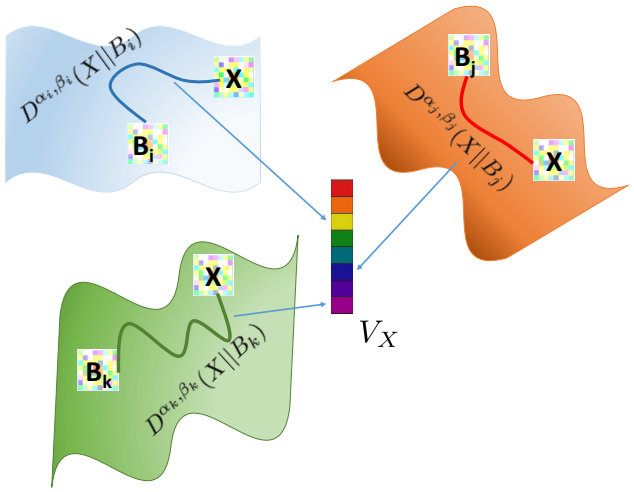


Fig. 2: A schematic illustration of our IDDL scheme. From an infinite set of potential geometries, our goal is to learn multiple geometries (parameterized by (α, β)) and representative dictionary atoms for each geometry (represented by B's), such that a given SPD data matrix X can be embedded into a similarity vector V_X, each dimension of which captures the divergence of X to the Bs using the respective measure. We use V_X for classification.

4.1 Information Divergence & Dictionary Learning

Suppose we are given a dataset of SPD matrices $\mathcal{X} = \{X_1, X_2, \cdots, X_N\}$, each $X_i \in \mathcal{S}_{++}^d$ and their associated class labels $y_i \in \mathcal{L} = \{1, 2, \cdots, L\}$. Our goal is to amalgamate three learning pursuits, namely, (i) learn a dictionary $\mathbf{B} \in \mathcal{S}_{++}^d \times_n$, a product of n SPD manifolds, (ii) learn an ABLD on each dictionary atom to best represent the given data for the task of classification, and (iii) learn a discriminative objective function on the encoded SPD matrices (in terms of $\mathbf{B}$ and the respective ABLDs) for the purpose of classification. This is the most general form of our classification model. For example, if we assume the dictionary atoms are all identity matrices, then our setting is basically a multi-divergence learning setting, while if we assume $n = 1$, we have a single divergence learning setup. We formalize the three aforementioned objectives in the formulation presented below.

$$\text{IDDL} := \min_{\substack{\mathbf{B}>0, W, \\ \boldsymbol{\alpha}>0, \boldsymbol{\beta}>0}} \sum_{i=1}^{N} f(\mathbf{v}_i, y_i; W)$$
$$\text{subject to } \mathbf{v}_i^k = D^{(\boldsymbol{\alpha}_k, \boldsymbol{\beta}_k)}(X_i \parallel B_k), \tag{10}$$

where the k-th dictionary atom in $\mathbf{B}$ is denoted by B_k, the vector $\mathbf{v}_i \in \mathbb{R}^n$ denotes the encoding of X_i in terms of the dictionary, and $\mathbf{v}_i^k$ is the k-th dimension of this encoding. Specifically, the k-th dimension of $\mathbf{v}_i$ captures the distance of X_i to the dictionary atom B_k via the respective divergence $D^{(\boldsymbol{\alpha}_k, \boldsymbol{\beta}_k)}(X_i \parallel B_k)$. Note that, in our formulations, we assume α, β are non-negative

to avoid degenerate solutions during optimization caused by the constraint in (2). Instead, we consider the positive and negative orthants separately, and design specialized descent for the case when $\alpha = \beta = 0$, and select the parameter configuration that leads to the smallest loss. The function f in (10), parametrized by $W \in \mathbb{R}^{L\times n}$, learns a classifier on $\mathbf{v}_i$ according to the provided class labels y_i. Figure 2 illustrates our classification model. For the classifier, we consider two popular loss functions, (i) a ridge regression loss, which has a closed form update as we show below, and (ii) a structured-SVM loss which we optimize in an iterative manner.

4.1.1 *Ridge Regression Loss*

Our first choice for f is a simple ridge regression objective. Let $\mathbf{h}_i \in \{0,1\}^L$ be a one-off encoding of class labels (i.e., $\mathbf{h}_i^{y_i} = 1$, zero everywhere else). Then, we define f_1 as follows,

$$f_1(\mathbf{v}_i, y_i; W) = \frac{1}{2}\left\|\mathbf{h}_i - W\mathbf{v}_i\right\|^2 + \gamma\left\|W\right\|_F^2, \qquad (11)$$

where γ is a regularization parameter.

4.1.2 *Structured-SVM Loss*

A more richer choice for f in (10) is perhaps to consider a max-margin hinge loss (similar to structured-SVM). Letting Δ denote a control variable on the margins of the classifiers and let $g(\mathbf{v}_i, W) = W\mathbf{v}_i + b$, for a bias $b \in \mathbb{R}^{L\times 1}$, we define f_2 as:

$$f_2(\mathbf{v}_i, y_i; W) = \sum_{l\neq y_i}\left(\max\left(0, g(\mathbf{v}_i, W)_l - g(\mathbf{v}_i, W)_{y_i} + \Delta\right)\right) + \gamma||W||_F^2, \qquad (12)$$

where γ is a regularization parameter, and $g(\mathbf{v}_i, W)_{y_i}$ represent the y_i element of the vector g. For simplifying our notation, in the sequel, for the case of structured-SVM loss, we assume $W \in \mathbb{R}^{L+1\times n}$, where the last row of W captures the bias b. Similarly, we augment $\mathbf{v}$ to have $n+1$ dimensions, where the last dimension is a constant with value one.

4.2 Information Divergence & Clustering

In the clustering setup, our objective is to partition a set of SPD matrices $\mathcal{X} = \{\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_N\}$ into k partitions, for a given k. Let $\mathbf{\Pi} = \{\pi_1, \cdots, \pi_k\}$ denote a partitioning of $\mathcal{X}$ where π_i is the set of samples assigned to the i-th cluster and let $\mathbf{C}_i$ be the respective cluster centroid. We cast the joint *information divergence learning and clustering* (IDC) problem as:

$$\text{IDC} := \min_{\mathbf{C},\mathbf{\Pi},\alpha,\beta>0} f_3\left(\mathbf{\Pi}, \mathcal{X}; \alpha, \beta\right) + \mathbf{\Omega}(\alpha, \beta), \qquad (13)$$

where our objective jointly learns a partition of data points $\mathbf{\Pi}$, the cluster centroids $\mathbf{C}$, and the divergence scalar parameters α, β. The function $\mathbf{\Omega}(\alpha, \beta) = \mu\left(\alpha^2 + \beta^2\right)$ is a regularization term on the parameters α and β, and μ is a regularization constant. Substituting the standard KMeans formulation in (13) and using ABLD as the similarity measure, we have the following definition for f_3:

$$f_3\left(\mathbf{\Pi}, \mathcal{X}; \alpha, \beta\right) = \sum_{\pi\in\mathbf{\Pi}}\sum_{i\in\pi}\left(D^{(\alpha,\beta)}(X_i \parallel \mathbf{C}_\pi)\right). \qquad (14)$$

5 Efficient Optimization

In this section, we briefly review the necessary Riemannian optimization machinery that we resort to for solving our objectives formulated in the last section.

5.1 Optimization on Riemannian Manifolds

As was shown in Section 4, we need to solve a non-convex constrained optimization problem in the form:

$$\begin{aligned} &\text{minimize } \mathcal{L}(B) \\ &\text{s.t. } \; B \in \mathcal{S}_{++}^d \,. \end{aligned} \qquad (15)$$

Classical optimization methods generally turn a constrained problem into a sequence of unconstrained problems for which unconstrained techniques can be applied. In contrast, in this paper we make use of the optimization on Riemannian manifolds to minimize (15). This is motivated by recent advances in Riemannian optimization techniques where benefits of exploiting geometry over standard constrained optimization are shown [60]. As a consequence, these techniques have become increasingly popular in diverse application domains [6], [44].

A detailed discussion of Riemannian optimization goes beyond the scope of this paper, and we refer the interested reader to [60]. However, the knowledge of some basic concepts will be useful in the remainder of this paper. As such, here, we briefly consider the case of Riemannian Conjugate Gradient method (RCG), our choice when the empirical study of this work is considered. First, we formally define the SPD manifold.

Definition 2 (The SPD Manifold). The set of $(d \times d)$ dimensional real, SPD matrices endowed with the Affine Invariant Riemannian Metric (AIRM) [7] forms the SPD manifold $\mathcal{S}_{++}^d$.

$$\mathcal{S}_{++}^p \triangleq \{X \in \mathbb{R}^{d\times d} : v^T X v > 0,\ \forall v \in \mathbb{R}^d - \{0_d\}\}\,. \qquad (16)$$

To minimize (15), RCG starts from an initial solution $B^{(0)}$ and improves its solution using the update rule

$$B^{(t+1)} = \tau_{B^{(t)}}\left(P^{(t)}\right), \qquad (17)$$

where $P^{(t)}$ identifies a search direction and $\tau_B(\cdot) : T_B\mathcal{S}_{++}^d \to \mathcal{S}_{++}^d$ is a *retraction*. The retraction serves to identify the new solution along the geodesic defined by the search direction $P^{(t)}$. In RCG, it is guaranteed that the new solution obtained by Eq. (17) is on $\mathcal{S}_{++}^d$ and has a lower objective. The search direction $P^{(t)} \in T_{B^{(t)}}\mathcal{S}_{++}^d$ is obtained by

$$P^{(t)} = -\text{grad}\,\mathcal{L}(B^{(t)}) + \eta^{(t)}\pi(P^{(t-1)}, B^{(t-1)}, B^{(t)})\,. \qquad (18)$$

Here, $\eta^{(t)}$ can be thought of as a variable learning rate, obtained via techniques such as Fletcher-Reeves [60]. Furthermore, $\text{grad}\,\mathcal{L}(B)$ is the Riemannian gradient of the objective function at B and $\pi(P, X, Y)$ denotes the parallel transport of P from T_X to T_Y. In Table 2, we define the mathematical entities required to perform RCG on the SPD manifold. Note that computing the standard Euclidean gradient of the function $\mathcal{L}$, denoted by $\nabla_*(\mathcal{L})$, is the only requirement to perform RCG on $\mathcal{S}_{++}^d$.

	$\mathcal{S}_{++}^d$
Riemannian gradient	$\text{grad}\,\mathcal{L}(B) = B\text{sym}\big(\nabla_B(\mathcal{L})\big)B$
Retraction.	$\tau_B(\xi) = B^{\frac{1}{2}}\,\text{Exp}(B^{-\frac{1}{2}}\xi B^{-\frac{1}{2}})B^{\frac{1}{2}}$
Parallel Transport.	$\pi(P, X, Y) = ZPZ^T$

TABLE 2: Riemannian tools to perform RCG on $\mathcal{S}_{++}^d$. Here, $\text{sym}(X) = \frac{1}{2}(X + X^T)$, $\text{Exp}(\cdot)$ denotes the matrix exponential and $Z = (YX^{-1})^{\frac{1}{2}}$.

5.2 Learning ABLDs via Block Coordinate Descent

We propose to use a block-coordinate descent (BCD) scheme to optimize our objectives in (10) and (14). In this scheme, each variable is updated alternately, while fixing others. We use the Riemannian conjugate gradient (RCG) algorithm [60] for optimizing over the dictionary atoms in IDDL (and cluster centroids in IDC). As our objective is non-convex in its variables (except for W), convergence of BCD iterations to a global minima is not guaranteed. In Alg. 1 and Alg. 2, we detail out the meta-steps in our optimization scheme for solving our IDDL objective, while Alg. 3 details the steps for optimizing our IDC objective. We initialize the dictionary atoms and the divergence parameters as described in Section 6.3.

Recall from Section 5.1 that an essential ingredient in RCG is efficient computations of the Euclidean gradients of the objective with respect to the variables. In the following, we derive expressions for these gradients. Note that we assume that the dictionary atoms (i.e., B_i) to be on an SPD manifold. As the optimization setup when both $\boldsymbol{\alpha}, \boldsymbol{\beta}$ are either positive or negative is essentially the same, in the derivations to follow, we assume $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ belong to the non-negative orthant of the Euclidean space. We use the spectral projected gradient (SPG) descent algorithm [61] for learning $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. This algorithm uses Barzillai-Borwein step-size selection in the gradient descent, and projects the iterates into the non-negative orthant (or to the non-positive orthant if $\boldsymbol{\alpha}, \boldsymbol{\beta} < 0$) to enforce the constraints. Empirically, using SPG is seen to converge very quickly, as is also observed in [44].

Input: $\mathcal{X}$, H, n
$\mathbf{B} \leftarrow$ log-euc-kmeans$(\mathcal{X}, n)$, $(\boldsymbol{\alpha}, \boldsymbol{\beta}) \leftarrow$ GridSearch;
repeat
  for $k = 1$ **to** n **do**
    $B_k \leftarrow$ update_B$(\mathcal{X}, W, \boldsymbol{\alpha}, \boldsymbol{\beta}, B_k)$; // use (27)
  end
  $(\boldsymbol{\alpha}, \boldsymbol{\beta}) \leftarrow$ update_$\alpha\beta(\mathcal{X}, W, \mathbf{B}, \boldsymbol{\alpha}, \boldsymbol{\beta})$; // use (28)
  $W \leftarrow$ update_W; // use (25)
until *until convergence*;
return $\mathbf{B}, \boldsymbol{\alpha}, \boldsymbol{\beta}$

Algorithm 1: Block-Coordinate Descent for IDDL and Ridge Regression Loss.

Input: $\mathcal{X}$, H, n
$\mathbf{B} \leftarrow$ log-euc-kmeans$(\mathcal{X}, n)$, $(\boldsymbol{\alpha}, \boldsymbol{\beta}) \leftarrow$ GridSearch;
repeat
  for $k = 1$ **to** n **do**
    $B_k \leftarrow$ update_B$(\mathcal{X}, W, \boldsymbol{\alpha}, \boldsymbol{\beta}, B_k)$; // use (32)
  end
  $(\boldsymbol{\alpha}, \boldsymbol{\beta}) \leftarrow$ update_$\alpha\beta(\mathcal{X}, W, \mathbf{B}, \boldsymbol{\alpha}, \boldsymbol{\beta})$; // use (33)
  repeat
    $W \leftarrow$ update_W; // use (29) and (30)
  until *until convergence*;
until *until convergence*;
return $\mathbf{B}, \boldsymbol{\alpha}, \boldsymbol{\beta}$

Algorithm 2: Block-Coordinate Descent for IDDL Structured SVM Loss.

5.3 Gradients on ABLD

To complete the optimization schemes presented above, we need the gradients of our objectives with respect to the model parameters, which we present now. First, towards optimizing for the parameters of the divergence, quantities of particular interest are the derivatives of the ABLD with respect to its parameters α and β. Second, an equally important quantity for our learning schemes is the derivative of the ABLD with respect to the dictionary atoms. Furthermore, we explicitly handle the limiting case for $\alpha = \beta \to 0$ via the explicit derivation of the derivative of the ABLD with respect to its input matrices.

Input: $\mathcal{X}$, k
$\mathbf{C} \leftarrow$ log-euc-kmeans$(\mathcal{X}, n)$, $(\alpha, \beta) \leftarrow$ init(lb, up);
repeat
  for $z = 1$ **to** k **do**
    $C_z \leftarrow$ update_C$(\mathcal{X}, \mathbf{\Pi}, \alpha, \beta, C_z)$; // use (22)
  end
  $(\alpha, \beta) \leftarrow$ update_$\alpha\beta(\mathcal{X}, \mathbf{\Pi}, \alpha, \beta, C_z)$; // use (19)
  $\mathbf{\Pi} \leftarrow$ update_$\Pi(\mathcal{X}, \alpha, \beta, C)$; // use (34)
until *until convergence*;
return $C, \mathbf{\Pi}, \alpha, \beta$

Algorithm 3: Overview of Block-Coordinate Descent for IDC.

5.3.1 *Derivative of ABLD wrt* α

Towards computing the derivative of ABLD with respect to its parameter α, we use the form of the ABLD given in (3) that involves the generalized eigenvalues of XY^{-1}. Letting $\theta = \alpha + \beta$ and $\nu = \alpha\beta$, for $X, Y \in \mathcal{S}_{++}^d$ the derivative has the form:

$$\nabla_\alpha D^{(\alpha,\beta)}(X \parallel Y) = \sum_{i=1}^{d} \nabla_\alpha \left[\frac{1}{\nu} \log \frac{\alpha\lambda_i^\beta + \beta\lambda_i^{-\alpha}}{\theta} \right]$$
$$= \frac{1}{\alpha\nu} \sum_{i=1}^{d} \left\{ \frac{\alpha\lambda_i^\beta - \nu\lambda_i^{-\alpha}\log\lambda_i}{\alpha\lambda_i^\beta + \beta\lambda_i^{-\alpha}} - \frac{\alpha}{\theta} - \log \frac{\alpha\lambda_i^\beta + \beta\lambda_i^{-\alpha}}{\theta} \right\}. \tag{19}$$

Using the dual symmetry property of ABLD reviewed in (9), derivative of ABLD with respect to β is straightforward.

5.3.2 *Derivative of ABLD wrt* Y

Towards deriving the derivative of the ABLD with respect to its input matrix Y, we use the form of the divergence given in (1). In that way, letting $\rho = \frac{\alpha}{\beta}$ and $Z = X^{-1}$, the derivative of (1) with respect to matrix Y has the form:

$$\nabla_Y D^{(\alpha,\beta)}(X \parallel Y) = \frac{1}{\nu} \nabla_Y \log\det\left[\rho\left(ZY\right)^\theta + I_d\right] - \frac{1}{\beta} Y^{-1}. \tag{20}$$

The following theorem will come handy when we design gradients in our dictionary learning setup.

Theorem 2. For $A, B \in \mathcal{S}_{++}^d$ and $p, q \geq 0$,

$$\nabla_B \log\det\left[p\left(AB\right)^q + I_d\right] =$$
$$pqB^{-1}A^{-\frac{1}{2}}\left(A^{\frac{1}{2}}BA^{\frac{1}{2}}\right)^q \times \left(I_d + p\left(A^{\frac{1}{2}}BA^{\frac{1}{2}}\right)^q\right)^{-1} A^{\frac{1}{2}}.$$

Making use of Theorem 2, the derivative described in (20) becomes:

$$\nabla_Y D^{(\alpha,\beta)}(X \parallel Y) = \rho\theta Y^{-1} Z^{-\frac{1}{2}} \left(Z^{\frac{1}{2}} Y Z^{\frac{1}{2}}\right)^\theta$$
$$\times \left(I_d + \rho\left(Z^{\frac{1}{2}} Y Z^{\frac{1}{2}}\right)^\theta\right)^{-1} Z^{\frac{1}{2}} - \frac{1}{\beta} Y^{-1}. \tag{21}$$

Remark 1. Computing (21) for large datasets can become overwhelming. A more efficient implementation could be the following. Say, (U, Δ) be the Schur decomposition of $Z^{\frac{1}{2}} Y Z^{\frac{1}{2}}$, which is faster to compute than the eigenvalue decomposition [62] (required for computing $Z^{\frac{1}{2}}$). With $\boldsymbol{\delta} = \operatorname{diag}(\Delta)$, (21) can be rewritten as:

$$\nabla_Y D^{(\alpha,\beta)}(X \parallel Y) = \rho\theta Y^{-1}\left(Z^{-\frac{1}{2}}U\right)\left[\operatorname{diag}\left(\frac{\boldsymbol{\delta}^\theta}{1+\rho\boldsymbol{\delta}^\theta}\right)\right] \times \left(Z^{-\frac{1}{2}}U\right)^{-1} - \frac{1}{\beta}Y^{-1}. \tag{22}$$

Compared to (21), this reduces the number of matrix multiplications from 5 to 3 and matrix inversions from 2 to 1.

5.3.3 *Derivative of the ABLD for $\alpha = \beta \to 0$ wrt Y*

As alluded to earlier, ABLD is non-smooth at the origin and we need to resort to the limit of the divergence, which happens to be the natural Riemannian metric (AIRM). That is,

$$D^{(0,0)}(X \parallel Y) = \left\|\operatorname{Log}\left(X^{-\frac{1}{2}} Y X^{-\frac{1}{2}}\right)\right\|_F^2. \tag{23}$$

Letting $P = X^{-\frac{1}{2}} Y X^{-\frac{1}{2}}$, the derivative of (23) with respect to matrix Y is given by:

$$\nabla_Y D^{(0,0)}(X \parallel Y) = 2X^{-\frac{1}{2}} \left(\operatorname{Log} P\right) P^{-1} X^{-\frac{1}{2}}. \tag{24}$$

Note that a simplification similar to (22) is also possible for (24).

With this general gradients for learning α, β, dictionary and the cluster centroids, now we consider learning parameters specific to each of our objectives.

5.4 Optimizing IDDL – Ridge Regression Objective

Reconsidering our ridge regression loss f_1 defined in (11), suppose V and H are matrices obtained by stacking $\mathbf{v}_i$ and $\mathbf{h}_i$ along their i-th column, for $i = 1, 2, \cdots, N$. When fixing $\mathbf{B}, \boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, the objective can be solved in closed form as:

$$W^* = HV^T(VV^T + \gamma I_d)^{-1}, \tag{25}$$

Towards deriving the gradient of the IDDL objective for f_1 w.r.t. the k^{th} dictionary atom B_k we capitalize on the observation that only the k-th dimension of $\mathbf{v}_i$ involves B_k. To simplify the notation, let us assume:

$$\zeta_i = -(\mathbf{h}_i - W\mathbf{v}_i)^T W, \tag{26}$$

and let ζ_i^k be its k-th dimension. Then we have:

$$\nabla_{B_k} f_1 = \zeta_i^k \nabla_{B_k}\left(D^{(\boldsymbol{\alpha}^k,\boldsymbol{\beta}^k)}(X_i \parallel B_k)\right), \tag{27}$$

where the gradient of the ABLD w.r.t. the k-th atom is derived in (22). Similarly, for gradients w.r.t. $\boldsymbol{\alpha}_k$, using the derivative of the ABLD derived in (19), we get:

$$\nabla_{\boldsymbol{\alpha}_k} f_1 = \zeta_i^k \nabla_{\boldsymbol{\alpha}_k}\left(D^{(\boldsymbol{\alpha}^k,\boldsymbol{\beta}^k)}(X_i \parallel B_k)\right). \tag{28}$$

It should be noted that the gradients w.r.t. $\boldsymbol{\beta}_k$ can be easily computed using the dual symmetry property described in (9).

5.5 Optimizing IDDL – Structured SVM Objective

Looking back at our SVM loss f_2 in (12); differentiating f_2 w.r.t. the rows of W while fixing $\mathbf{B}, \boldsymbol{\alpha}$, and $\boldsymbol{\beta}$ we get:

$$\nabla_{\mathbf{w}_{y_i}} f_2 = -\left(\sum_{j \neq y_i} \mathbb{1}_{\left(\mathbf{w}_j^T\mathbf{v}_i - \mathbf{w}_{y_i}^T\mathbf{v}_i + \Delta > 0\right)}\right)\mathbf{v}_i + 2\gamma\mathbf{w}_{y_i} \tag{29}$$

$$\nabla_{w_{j\neq y_i}} f_2 = \mathbb{1}_{\left(\mathbf{w}_j^T\mathbf{v}_i - \mathbf{w}_{y_i}^T\mathbf{v}_i + \Delta > 0\right)}\mathbf{v}_i + 2\gamma\mathbf{w}_j \tag{30}$$

where $\mathbb{1}_{(.)}$ is the indicator function. Similarly to our derivations for f_1, to simplify our notations we let:

$$\xi_i = -\sum_{j \neq y_i} \mathbb{1}_{\left(\mathbf{w}_j^T\mathbf{v}_i - \mathbf{w}_{y_i}^T\mathbf{v}_i + \Delta > 0\right)}\left(\mathbf{w}_j - \mathbf{w}_{y_i}\right). \tag{31}$$

and let ξ_i^k be its k-th dimension. That way, making use of (22) and (19), we define the gradients of f_2 w.r.t to B_k and $\boldsymbol{\alpha}_k$, respectively as:

$$\nabla_{B_k} f_2 = \xi_i^k \nabla_{B_k}\left(D^{(\boldsymbol{\alpha}^k,\boldsymbol{\beta}^k)}(X_i \parallel B_k)\right), \tag{32}$$

$$\nabla_{\boldsymbol{\alpha}_k} f_2 = \xi_i^k \nabla_{\boldsymbol{\alpha}_k}\left(D^{(\boldsymbol{\alpha}^k,\boldsymbol{\beta}^k)}(X_i \parallel B_k)\right). \tag{33}$$

5.6 Optimizing IDC – Clustering Objective

The gradients of f_3 with respect to both the divergence parameters, as well as the clustering centroids, are obtained directly from (19) and (22), respectively. Lastly, to update $\boldsymbol{\Pi}$ in (14), we need to find the cluster centroid C_π nearest to a given data point X_i, for which we solve the following argmin problem, by assuming the ABLD parameters are fixed at the current iterate. Formally, the data points in the cluster π_z are updated as $\pi_{z^*} \to \pi_{z^*} \cup \{X_i\}$, where:

$$z^* = \underset{\forall z \in \{1,2,\cdots,k\}}{\arg\min} D^{(\alpha,\beta)}(X_i \parallel \mathbf{C}_z). \tag{34}$$

5.7 Computational Complexity

It should be noted that terms such as ${X_i}^{-1}$, which can be computed offline are omitted from this analysis. Using the simplifications depicted in (22) and Schur decomposition, gradient computation for each dictionary atom or centroid takes $\mathcal{O}(Nd^3)$ flops. Using the gradient formulation in (19) for $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, we need $\mathcal{O}(Ndn + Nd^3)$ flops. Computations of the closed form for W using the ridge regression loss in (25) takes $\mathcal{O}(n^2(L+N) + n^3 + nLN)$. For the discriminative setup, at test time, given that we have learned the dictionary and the parameters of the divergence, encoding a data matrix requires $\mathcal{O}(nd^3)$ flops, which is similar in complexity to the recent sparse coding schemes such as [44]. As for the gradient computation for each C in IDC takes $\mathcal{O}(Nd^3)$ flops and the overall clustering setup takes $\mathcal{O}(Ndk + Nd^3)$ flops, similar in complexity to a Karcher mean algorithm [10] using AIRM as the similarity measure.

6 EXPERIMENTS

In this section, we present a thorough evaluation of the proposed inference schemes on a diverse set of computer vision datasets. We use the following eight datasets, namely (i) JHMDB [63], (ii) HMDB [64] (iii) KTH-TIPS2 [65], (iv) Brodatz textures [66], (v) Virus [67], (vi) SHREC 3D [68], (vii) Myometrium cancer [69], and (viii) Breast cancer [69]. Below, we provide details of all these datasets, and how SPD descriptors are obtained on them.

We use standard evaluation schemes reported previously on these datasets. In some cases, we use our own implementations of popular methods but strictly following the recommended evaluation settings. For those datasets that do not have prescribed cross-validation splits, we repeat the experiments at least 5 times and average the performance scores. For our SVM-based experiments, we use a linear SVM on the log-Euclidean mapped SPD matrices.

6.1 Datasets

HMDB [64] and JHMDB [63] datasets: These are two popular action recognition benchmarks. The HMDB dataset consists of 51 action classes associated with 6766 video sequences, each sequence with 30–400 video frames. JHMDB is a subset of HMDB with 955 sequences in 21 action classes with 15–40 frames in each video, where the video subset in JHMDB has human pose better visible. Following the official train/test split, we use 70% of the videos for training and rest for testing on both datasets. To generate SPD matrices on these datasets, we use the scheme proposed in [70], where we compute RBF kernel descriptors on the output of per-frame CNN class predictions (fc8) for each stream (RBF and optical flow) separately, and fusing these two SPD matrices into a single block-diagonal matrix per sequence. For the two-stream model, we use a VGG16 model trained on optical flow and RGB frames separately as described in [71]. Thus, our descriptors are of size 102×102 for HMDB and 42×42 for JHMDB.

SHREC 3D Object Recognition Dataset [68]: It consists of 15000 RGBD covariance descriptors generated from the SHREC dataset [68] by following [72]. SHREC consists of 51 3D object classes. The descriptors are of size 18×18. Similar to [44], we randomly picked 80% of the dataset for training and used the remaining for testing.

KTH-TIPS2 dataset [65] and Brodatz Textures [66]: These are popular texture recognition datasets. The KTH-TIPS dataset consists of 4752 images from 11 material classes under varying conditions of illumination, pose, and scale. Covariance descriptors of size 23×23 are generated from this dataset following the procedure in [47]. We use the standard 4-split cross-validation for our evaluations on this dataset. As for the Brodatz dataset, we used 100 texture images for our experiments, each image is 640×640 resoultion. To produce the covariance descriptors, we follow the procedure outlined in [44]. Specifically, we extracted 32×32 non-overlapping patches from each image. Next, to produce the covariance descriptor for a given patch, we used the relative pixel coordinates for all pixels in the patch, its image intensity, and respective image gradients, which form 5-dimensional features, from which 5×5 region covariance descriptors are produced for the respective patch. Our dataset consists of 31000 SPD matrices. As proposed in [44], for our evaluation use an 80:20 rule as in the RGBD dataset above.

Virus Dataset [67]: It consists of 1500 images of 15 different virus types. Similar to the KTH-TIPS, we use the procedure in [47] to generate 29×29 covariance descriptors from this dataset and follow their evaluation scheme using three-splits.

Cancer Datasets [69]. Apart from these standard SPD datasets, we also report performances on two cancer recognition datasets from [69]. We use images from two types of cancers, namely (i) Breast cancer, consisting of binary classes (tissue is either cancerous or not) consisting of about 3500 samples, and (ii) Myometrium cancer, consisting of 3320 samples; we use covariance-kernel descriptors as described in [69] which are of size 8×8. We follow the 80:20 rule for evaluation on this dataset as well.

6.2 Experimental Setup

Since we present experiments on a variety of datasets and under various configurations, we summarize our main experiments first. There are four sets of experiments we conduct, namely (i) an ablative study of various parameters, learning configurations, convergence, and run-time analysis of our problem setup, (ii) comparison of IDDL against other popular measures on SPD matrices, (iii) comparisons among various configurations of IDDL, and (iv) comparisons against state of the art approaches on the above datasets. We follow a similar trend in our experimental setup for our clustering setup.

6.3 Parameter Initialization

In all the experiments, we initialized the parameters of IDDL (e.g., the initial dictionary) in a principle-way. We initialized the dictionary atoms by applying log-Euclidean K-Means; i.e., we compute the log-Euclidean map of the SPD data, compute Euclidean K-Means on these mapped points, and remap the K-Means centroids to the SPD manifold via an exponential map. To initialize α and β, we recommend grid-search by fixing the dictionary atoms as above. As an alternative to the grid-search, we empirically observed that a good choice is to start with the Burg divergence (i.e., $\alpha = \beta = 1$). The regularization parameter γ in IDDL was chosen using cross-validation, while the regularization parameter μ for IDC was set to 1.

6.4 Ablative Study

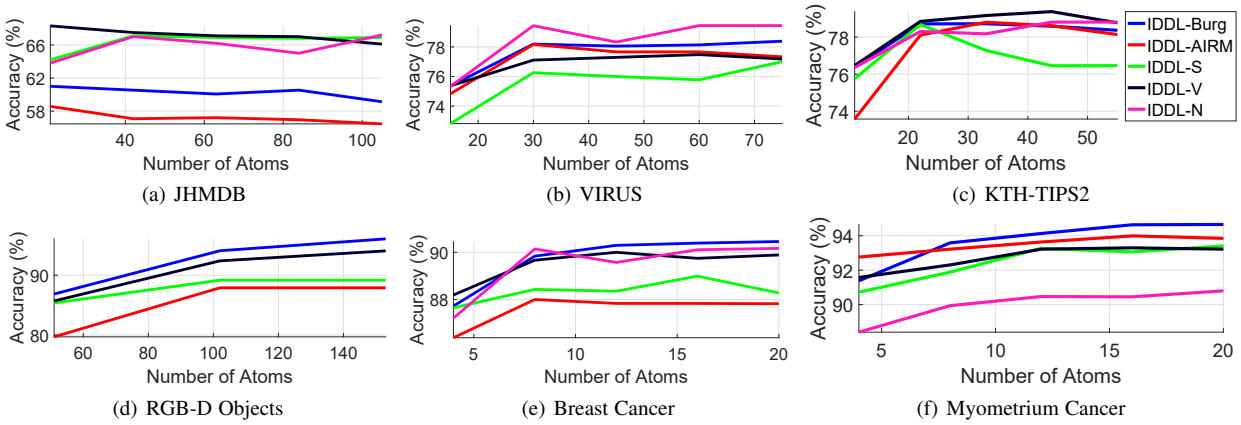
In this section, we study the influence of each of the components in our algorithm. First, we demonstrate how the performance on a dataset varies when changing the parameters α and β in ABLD. This forms the basis of all our further experiments.

6.4.1 Performance for Varying α, β

In Figure 4(a) and 4(b), we plot a heatmap of the classification accuracy against changing α and β on the KTH-TIPS2 and Virus datasets. We fixed the size of dictionaries to 22 for the KTH TIPS and 30 for the Virus datasets. The plots reveal that the performance varies for different parameter settings, thus (along with the results in Table 4) substantiates that learning these parameters is a way to improve performance.

6.4.2 Comparisons to Variants of IDDL

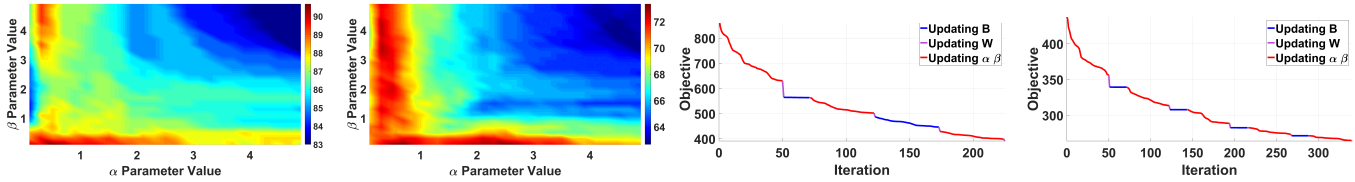
In this section, we analyze various aspects of the performance of IDDL, using the ridge regression objective. Generally speaking, IDDL formulation is generic and customizable. For example, even though we formulated the problem as using a separate ABLD on each dictionary atom, it does not hurt to learn the same divergence over all atoms in some applications. To this end, we test the performance of three scenarios, namely (i) using a scalar α and



(a) JHMDB (b) VIRUS (c) KTH-TIPS2

(d) RGB-D Objects (e) Breast Cancer (f) Myometrium Cancer

Fig. 3: Comparisons between different variants of IDDL for increasing number of dictionary atoms.



(a) (b) (c) (d)

Fig. 4: 4(a) and 4(b) show accuracy on KTH-TIPS2 and Virus datasets as we change α and β, fixing the number of dictionary atoms. See the algorithmic settings in Sec. 6.4.1). 4(c) and 4(d) show convergence of our optimization scheme for IDDL (ridge regression) on KTH-TIPS2 and Virus datasets. See the text for more details.

β that is shared across all the dictionary atoms (which we call IDDL-S), (ii) a vector α and β, where we assume $\alpha = \beta$, but each dictionary atom can potentially have a distinct parameter pair (we call this case IDDL-V), and (iii) the most generic case where we could have α, β as vectors and they may not be equal, which we refer as IDDL-N. In Figure 3, we compare all these configurations on six of the datasets. We also include specific cases such as the Burg divergence ($\alpha = \beta = 1$) and the AIRM case ($\alpha = \beta = 0$) for comparisons (using the dictionary learning scheme proposed in Section 5.3.2). Our experiments show that IDDL-N and IDDL-V consistently perform well on almost all datasets. This is unsurprising given the generality of IDDL. While IDDL-S shows similar performance to other methods when the number of atoms is small, it drops for more number of atoms.

6.4.3 *Convergence Study*

In Figures 4(c) and 4(d), we plot the convergence of our objective against iterations. We also depict the BCD objective as contributed by the dictionary learning updates and the parameter learning; we use the IDDL-V for this experiment. As is clear, most part of the decrement in objective happens when the dictionary is learned, which is not surprising given that it has the most number of variables to learn. For most datasets, we observe that the RCG converges in about 200-300 iterations. In Figure 5, we plot the running time for one iteration of RCG against the number of dimensions of the matrices and the number of dictionary atoms. While our dictionary updates seem quadratic in the number of dimensions, it scales linearly with the dictionary size.

6.4.4 *Evaluation of Joint Learning*

In Table 3, we evaluate the usefulness of learning the information divergence against learning the dictionary on the Virus dataset. For this experiment, we evaluated three scenarios, (i) fixing the dictionary to the initialization (using KMeans), and learning the parameters α, β using the IDDL-S variant, (ii) fixing α, β to the initialization using GridSearch, while learning the dictionary, and (iii) learning both dictionary and the parameters jointly. As the results in Table 3 shows, jointly learning the parameters demonstrates better results, thus justifying our IDDL formulation.

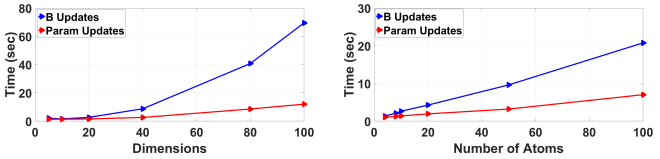


Fig. 5: Time taken for one gradient computation and two objective functions evaluations against an increasing number of matrix dimensions (left) and number of dictionary atoms (right).

Atoms — Method	IDDL-Fix(α, β)	IDDL-Fix$(\mathbf{B})$	IDDL-N
15	**78.33%**	61.67%	77.33%
45	80.33%	70.00%	**83.67%**
75	81.67%	76.00%	**82.33%**

TABLE 3: Performance evaluation of IDDL on a single split of the Virus dataset when fixing the dictionary atoms against fixing the parameters, and jointly learning the atoms and the parameters.

6.4.5 *Trajectories of α, β*

In this experiment, we demonstrate the BCD trajectories of α and β for the IDDL-S algorithm on the Virus dataset. Specifically, in Figure 6, we show how the value of α and β varies as the BCD iteration progresses. In this experiment, we used 15 dictionary atoms. All experiments used the same initializations for the invariants. We also plot the corresponding objective and

training accuracies. It appears that different initializations leads to disparate points of convergence. However, for all points of convergence, the objective convergence is very similar (and so is the training accuracy), suggesting that there are multiple local minima that leads to similar empirical results. We also find that initializing with $\alpha = 1.0$ demonstrates slightly better convergence than other possibilities, which we observed for other datasets too.

6.5 Comparisons to Standard Measures

In this experiment, we compare the IDDL (ridge regression) to the standard similarity measures on SPD matrices including log-Euclidean Metric [12], AIRM [7], and JBLD [17]. We report 1-NN classification performance on these baselines. In Table 4, we report the performance of these schemes. As a rule of thumb (and also supported empirically by cross-validation studies on our datasets), for a C-class problem, we chose $5C$ atoms in the dictionary. Increasing the size of the dictionary seems not helping in most cases. We also report a discriminative baseline by training a linear SVM on the log-Euclidean mapped SPD matrices. The results reported in Table 4 demonstrates the advantage of IDDL against the baselines, where the benefits can go over more than 10% in some cases (such as the JHMDB and virus).

6.6 Comparisons to the State of the Art

We compare IDDL to the following popular methods that share similarities to our scheme, namely (i) Log-Euclidean Metric learning (LEML) [27], (ii) kernelized Sparse Coding [47] that uses log-Euclidean metric for sparse coding SPD matrices (kSP_{LE}), (iii) kernelized sparse coding using JBLD (kSP_{JBLD}), (iv) kernelized locality constrained coding [6], and Riemannian dictionary learning and sparse coding (RSPDL) [44]. For IDDL, we chose the variant from Figure 3 that performed the best on the respective dataset (refer to the last column for the IDDL-variant). Our results are reported in Table 7. Again we observe that IDDL performs the best amongst all the competitive schemes, clearly demonstrating the advantage of learning the divergence and the dictionary. Note that comparisons are established by considering the same number of atoms for all schemes and fine-tuning the parameters of each algorithm (e.g., the bandwidth of the RBF kernel in kSP_{JBLD}) using a validation subset of the training set. As for LEML, we increased the number of pairwise constraints until the performance hit a plateau.

In Table 6, we further compare our best results on these datasets against our IDDL-V and N variants using the structured-SVM objective. We used the predicted label of the IDDL classifier (ridge regression or the structured SVM) for evaluation on all the datasets we use, except HMDB. On the HMDB dataset, we found that training an additional lib-linear SVM solver [73] on the embeddings produced by our trained ABLD model performed better ($\sim$ 1% better). This observation is perhaps unsurprising, given that we use 102-D covariance descriptors for this dataset (largest among our datasets), which are often (nearly) ill-conditioned, and thus may prevent the ABLD classifier from achieving the best performances in the non-convex learning setup, thereby producing embeddings that may be noisy. From Table 6, it is clear that our SSVM formulation is better on some datasets, for example, on Brodatz, and 3D object datasets, there is a substantial gain of nearly 5%, while on others the performance is similar to ridge regression objective. On the smaller datasets, such as Breast and Myometrium cancer, ridge regression is much better.

6.7 Evaluation of Clustering Objective

Now, we present experiments on the aforementioned benchmarks using our $\alpha\beta$-KMeans clustering framework. To evaluate the quality of clustering, we use the standard F1-Score. Before presenting comparisons to other popular clustering schemes on SPD matrices, we study the empirical properties of our formulation next.

We compare the quality of clustering against (i) dimensionality of the input matrices, and (ii) number of true clusters, and iii) time taken for the relevant updates. For this experiment, we use synthetic datasets generated using the code from [54], which produces Wishart SPD matrix clusters for k arbitrarily parameterized Wishart distributions; k being the number of true data clusters.

6.7.1 Increasing Matrix Dimensionality

For this experiment, we generate synthetic SPD datasets of dimensionality d, where $d \in \{5, 15, 30, 50, 75, 100\}$ corresponding to $k = 15$ clusters and using fifty samples per class. Figure 7(a) summarizes the computed F1-scores averaged across ten runs. We can clearly see that the accuracy of $\alpha\beta$-KMeans is not impacted much by the increasing dimensions of the input matrices, while both variants consistently outperform the baseline of LE-KMeans. Figure 7(b) present the time taken for a single iteration of each optimization component of $\alpha\beta$-KMeans, which looks approximately linear.

6.7.2 Increasing Number of Clusters

Next, we test the robustness of the $\alpha\beta$-KMeans with increasing number of true data clusters k, for $k \in \{2, 5, 10, 20, 50, 100\}$. For this experiment, we keep the dimension of the SPD matrices fixed to $d = 10$ and use twenty five samples per true data class. Figure 7(c) summarizes the F1-Score of $\alpha\beta$-KMeans averaged across ten runs for an increasing number of clusters. We can infer that both variants are negatively affected by large increases in the number of clusters, nevertheless, the our performance is consistently higher than that of the LE-KMeans baseline. In addition, as depicted in Figure 7(d), there is an increasing overall trend in the time required for all components of $\alpha\beta$-KMeans (compared to Figure 7(b), due to the time required to iterate through the different clusters.

6.7.3 Empirical Convergence Analysis

Now, we empirically study the convergence of $\alpha\beta$-KMeans. We select to present this analysis on the Myometrium cancer dataset nevertheless, the results remain consistent among the different datasets. Figure 8 illustrates the convergence of the BCD scheme discussed in Section 4 for $\alpha\beta$-KMeans with $\alpha = \beta$. Even though the objective is non-convex, it is apparent that the convergence is satisfactory. We run the scheme until more than 99.9% of the clustering assignments remain unchanged between two successive clustering steps.

6.7.4 Comparisons to Variants of KMeans

Comparisons are first established against two popular variants of KMeans for SPD matrices; LE-KMeans and Karcher Means. Table 7 summarizes the experiments evaluating the performance of the $\alpha\beta$-KMeans in a pure clustering setup. The first and second columns correspond to the F1-Score achieved by LE-KMeans and Karcher Means respectively. The two proposed variants of $\alpha\beta$-KMeans are depicted in columns three ($\alpha = \beta$) and four ($\alpha \neq \beta$). For each dataset, we average our results across ten different runs to alleviate the effect of initializations. We can clearly see

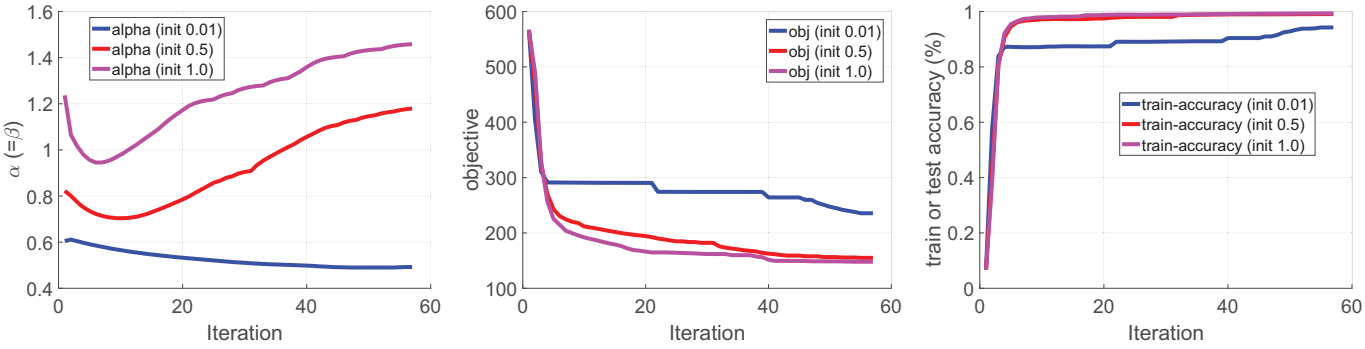


Fig. 6: (a) Trajectory of α and β over BCD iterations on the virus dataset, (b) shows the respective objective descent, and (c) shows the training set accuracy. We plot for various initializations of α and β.

Dataset — Classifier	LE 1-NN	AIRM 1-NN	JBLD 1-NN	SVM-LE	IDDL	Variant
JHMDB	52.99%	51.87%	52.24%	54.48%	**68.3%**	**V**
HMDB	29.30%	43.3%	46.3%	41.7%	**55.50%**	**N**
VIRUS	66.67%	67.89%	68.11%	68.00%	**78.39%**	**N**
BRODATZ	80.10%	80.50%	80.50%	**86.80%**	74.10%	**N**
KTH TIPS	72.05%	72.83%	72.87%	75.59%	**79.37%**	**V**
3D Object	97.4%	98.2%	95.6%	**98.9%**	96.08%	**Burg**
Breast Cancer	87.42%	80.00%	84.00%	87.71%	**90.46%**	**Burg**
Myometrium Cancer	80.87%	84.18%	93.20%	93.22%	**94.66%**	**Burg**

TABLE 4: Comparisons against 1-NN and SVM classification. Last column shows the variant of IDDL that worked best.

Dataset/Method	LEML	kSP$_{LE}$	kSP$_{JBLD}$	kLLC	RSPDL	IDDL-S	IDDL-V	IDDL-N	IDDL-A	IDDL-B
JHMDB	58.85%	55.97%	44.40%	57.46%	57.5%	67.10%	**68.3%**	67.20%	61.19%	61.01%
HMDB	52.15%	44.9%	28.43%	40.20%	21.0%	52.30%	57.3%	**58.6%**	43.20%	46.94%
VIRUS	74.60%	68.00%	57.84%	70.91%	60.8%	76.48%	77.74%	**79.44%**	78.33%	78.37%
BRODATZ	47.15%	55.00%	65.19%	70.00%	74.9%	72.50%	73.2%	77.10%	62.63%	**79.44%**
KTH TIPS	79.25%	77.18%	69.92%	73.96%	64.5%	78.68%	79.37%	**79.67%**	78.80%	78.36%
3D Object	87.56%	59.26%	72.45%	87.40%	80.0%	89.17%	94.07%	92.57%	87.90%	**96.08%**
Breast Cancer	83.18%	76.34%	71.67%	82.32%	74.2%	89.99%	90.00%	90.02%	88.00%	**90.46%**
Myometrium Cancer	90.94%	88.69%	86.80%	88.74%	87.0%	93.41%	93.30%	90.24%	93.99%	**94.66%**

TABLE 5: Comparisons against state of the art. IDDL-A and IDDL-B refers to IDDL-AIRM and IDDL-Burg respectively. Refer to Section 6.4.2 for details of other abbreviations.

	Ridge Regression		Structured-SVM	
Datasets	IDDL_V	IDDV_N	IDDL_V	IDDL_N
JHMDB	68.3%	67.2%	**69.2%**	65.8%
HMDB	57.3%	**58.6%**	56.7%	53.8%
VIRUS	77.4%	79.4%	**79.6%**	78.6%
BRODATZ	73.2%	77.1%	81.5%	**82.1%**
KTH TIPS2	**79.4%**	79.7%	71.3%	73.7%
3D Object	94.1%	92.3%	**98.3%**	98.2%
Breast	**90.0%**	88.0%	78.1%	76.7%
Myometrium	**93.3%**	90.2%	89.8%	89.2%

TABLE 6: Comparisons between IDDL variants for ridge regression and structured SVM alternatives.

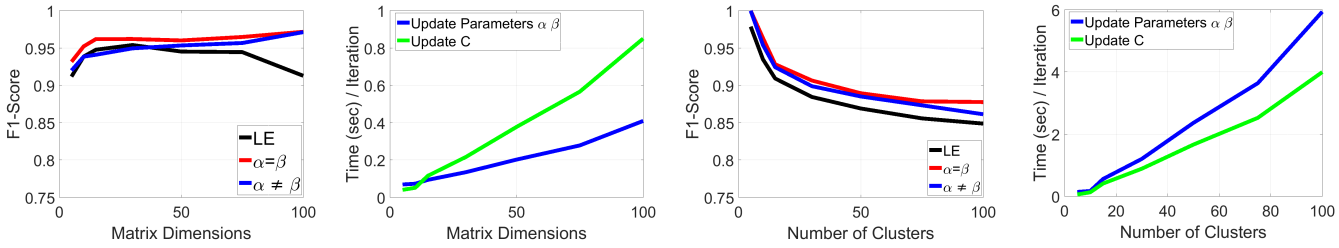
Dataset — Method	LE	Karcher	$\alpha\beta$-E	$\alpha\beta$-NE
VIRUS	0.248	0.254	0.252	**0.257**
BRODATZ	0.353	0.366	0.378	**0.381**
KTH TIPS	0.379	0.400	**0.429**	0.419
Prostate Cancer	0.578	0.594	**0.679**	0.660
Myometrium Cancer	0.737	0.661	0.778	**0.779**

TABLE 7: F1-Score based comparisons against different KMeans variants.

that the two variants of $\alpha\beta$-KMeans consistently outperform the competing schemes underlying the merits of learning the measure while clustering the data.

7 Discussions

Our experiments show that learning the divergence and the parameters of the respective task demonstrate superior performances on all the datasets we used. That said, there are also some challenges one may need to circumvent when using the setup. The main limitation of our approach is the non-convexity of our objective; that precludes a formal analysis of the convergence. A further limitation is that the gradient expressions involve matrix inversions and may need careful regularizations to avoid numerical instability. We also note that the AB divergence has a discontinuity at the origin, which needs to be accounted for when learning the parameters. Further, from our experimental analysis, it looks like there is no single variant of IDDL (amongst IDDL-S, IDDL-V, IDDL-N, IDDL-A, and IDDL-B) that consistently performs the best for all datasets. However, with the possibility of learning alpha-beta, we would think the most generalized variant IDDL-N with the structurd-SVM formulation might perhaps be the



(a) (b) (c) (d)

Fig. 7: Sensitivity of $\alpha\beta$-KMeans against 7(a) increasing dimensionality in the range $[5, 100]$. The blue and red lines correspond to $\alpha\beta$-KMeans with $\alpha = \beta$ and $\alpha \neq \beta$ respectively, while the black line corresponds to LE-KMeans. 7(b) Time required for each iteration of updating parameters $\alpha\beta$ (blue line) and centroids (green line). 7(c) and 7(d) show the same for increasing number of clusters.

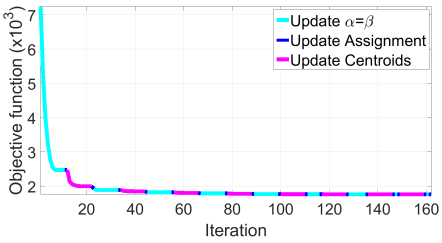


Fig. 8: Convergence plot of the objective function 13 for the myometrium cancer dataset and $\alpha = \beta$. Cyan line segments correspond to iterations of updating the divergence parameters, blue segments correspond to updating the clustering assignment and magenta segments correspond to iterations of updating the centroids.

best choice for any application as it can plausibly learn all the alternatives.

As for our clustering setup, we found that it is essential to use a regularizer on α and β; in the absence of which, the optimization was seen to diverge, the parameters taking very large values leading to irrecoverable numerical deficiencies. As noted earlier, we found quadratic regularizers on α, β yielded good results. Exploring other forms, such as polynomials on α and β, or robust priors such as the Huber loss, is left as future work. From our experiments on real data, we found beneficial small additive perturbations on the diagonal of the SPD matrices (to make them strongly positive definite). On all our datasets, we found each block of updates using RCG converged in about 5-10 steps. Surprisingly, the proposed BCD scheme is seen to converge much faster for the $\alpha \neq \beta$-case in comparison to $\alpha = \beta$, when centroids are initialized using the LE-KMeans rather than randomly selecting samples from each dataset. This faster convergence is perhaps because of the more degrees of parameter freedom and the conditioning of the matrices.

8 Conclusions

In this paper, we proposed a novel framework unifying the problem of information divergence learning and standard machine learning tasks, such as dictionary learning, ridge regression, classification, and clustering on SPD matrices. We leveraged the recent advances in information geometry for this purpose, namely using the $\alpha\beta$-logdet divergence. We formulated objectives for jointly learning the divergence and the respective task specific objectives, and showed that it can be solved efficiently using optimization methods on Riemannian manifolds in an end-to-end manner. Experiments on eight computer vision datasets demonstrate superior performance of our approach against alternatives.

9 Acknowledgements

This material is based upon work supported by the National Science Foundation through grants #SMA-1028076, #CNS-1338042, #CNS-1439728, #CNS-1514626, #CNS-1939033, and #CNS-1919631. Anoop Cherian was funded by the Australian Research Council Centre of Excellence for Robotic Vision (#CE140100016). Research reported in this publication was supported by the National Cancer Institute of the NIH under Award Number R01CA225435.

Appendix

Proof of Theorem 2. To simplify the notation, let $S = A^{\frac{1}{2}}$. Note that the eigenvalues (and hence the $\log\det$) of AB is the same as that of $A^{\frac{1}{2}}BA^{\frac{1}{2}}$, however, the latter is symmetric and thus keeps B symmetric, when using it in a gradient descent scheme. Thus, we will use this form. Then,

$$\log\det\left[p\left(SBS\right)^q + I_d\right] = \mathrm{Tr}\left(\mathrm{Log}\left(p\left(SBS\right)^q + I_d\right)\right). \quad (35)$$

Using Taylor series expansion of Log:

$$(35) = \mathrm{Tr}\left(p(SBS)^q - \frac{p^2(SBS)^{2q}}{2} - \cdots\right). \quad (36)$$

$$\begin{aligned}\nabla_B\,(36) &= pqS(SBS)^{q-1}S - qp^2S(SBS)^{2q-1}S + \cdots \\ &= pqS(SBS)^{-1}\left(SBS\right)^q\left(I_d - p\left(SBS\right)^q + \cdots\right)S\end{aligned}$$

Recall that, the middle term is the MacLaurin series expansion:

$$\left(I_d - p\left(SBS\right)^q + \cdots\right) = \left(I_d + p\left(SBS\right)^q\right)^{-1}$$

substituting for which we get our desired result. Note that, the series expansions we use in the proof are valid only when $p\left\|SBS\right\|_2 \leq 1$, which can be achieved via rescaling or normalizing our data. ■

References

[1] O. Tuzel, F. Porikli, and P. Meer, "Region covariance: A fast descriptor for detection and classification," in *ECCV*, 2006.

[2] A. Cherian and S. Gould, "Second-order temporal pooling for action recognition," *International Journal of Computer Vision*, vol. 127, no. 4, pp. 340–362, 2019.

[3] Y. Gao, O. Beijbom, N. Zhang, and T. Darrell, "Compact bilinear pooling," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[4] C. Ionescu, O. Vantzos, and C. Sminchisescu, "Matrix backpropagation for deep networks with structured layers," in *International Conference on Computer Vision*, 2015.

[5] Z. Huang and L. Van Gool, "A Riemannian network for SPD matrix learning," *AAAI Conference on Artificial Intelligence*, 2017.

[6] M. Harandi and M. Salzmann, "Riemannian coding and dictionary learning: Kernels to the rescue," in *CVPR*, 2015.

[7] X. Pennec, P. Fillard, and N. Ayache, "A Riemannian framework for tensor computing," *International Journal of Computer Vision*, vol. 66, no. 1, pp. 41–66, 2006.

[8] T. Brox, J. Weickert, B. Burgeth, and P. Mrázek, "Nonlinear structure tensors," *Image and Vision Computing*, vol. 24, no. 1, pp. 41–55, 2006.

[9] F. Yger, F. Lotte, and M. Sugiyama, "Averaging covariance matrices for eeg signal classification based on the CSP: An empirical study," in *European Signal Processing Conference*, 2015.

[10] D. A. Bini and B. Iannazzo, "Computing the Karcher mean of symmetric positive definite matrices," *Linear Algebra and its Applications*, vol. 438, no. 4, pp. 1700–1710, 2013.

[11] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Advances in Neural Information Processing Systems*, 2017.

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

[13] M. T. Harandi, M. Salzmann, and R. Hartley, "Dimensionality reduction on SPD manifolds: The emergence of geometry-aware methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 1, pp. 48–62, 2018.

[14] R. Wang, H. Guo, L. S. Davis, and Q. Dai, "Covariance discriminative learning: A natural and efficient approach to image set classification," in *CVPR*, 2012.

[15] K. Yu and M. Salzmann, "Statistically-motivated second-order pooling," in *European Conference on Computer Vision*, 2018.

[16] T.-Y. Lin, S. Maji, and P. Koniusz, "Second-order democratic aggregation," in *European Conference on Computer Vision*, 2018.

[17] A. Cherian, S. Sra, A. Banerjee, and N. Papanikolopoulos, "Jensen-Bregman Log-Det divergence with application to efficient similarity search for covariance matrices," *PAMI*, vol. 35, no. 9, pp. 2161–2174, 2013.

[18] S. Sra, "A new metric on the manifold of kernel matrices with application to matrix geometric means," in *Advances in Neural Information Processing Systems*, 2012.

[19] A. Bhattacharyya, "On a measure of divergence between two statistical populations defined by their probability distributions," *Bull. Calcutta Math. Soc.*, vol. 35, pp. 99–109, 1943.

[20] B. Kulis, M. Sustik, and I. Dhillon, "Learning low-rank kernel matrices," in *ICML*, 2006.

[21] M. Moakher and P. G. Batchelor, "Symmetric positive-definite matrices: From geometry to applications and visualization," in *Visualization and Processing of Tensor Fields*. Springer, 2006, pp. 285–298.

[22] S. Luo and Q. Zhang, "Informational distance on quantum-state space," *Physical Review A*, vol. 69, no. 3, p. 032106, 2004.

[23] R. Bhatia, T. Jain, and Y. Lim, "On the Bures–Wasserstein distance between positive definite matrices," *Expositiones Mathematicae*, vol. 37, no. 2, pp. 165–191, 2019.

[24] M. H. Quang, M. San Biagio, and V. Murino, "Log-Hilbert-Schmidt metric between positive definite operators on Hilbert spaces," in *Advances in Neural Information Processing Systems*, 2014.

[25] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon, "Information-theoretic metric learning," in *International Conference on Machine Learning*, 2007.

[26] Z. Huang and L. Van Gool, "A Riemannian network for SPD matrix learning," in *AAAI Conference on Artificial Intelligence*, 2017.

[27] 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 *International Conference on Machine Learning*, 2015.

[28] P. Zadeh, R. Hosseini, and S. Sra, "Geometric mean metric learning," in *International Conference on Machine Learning*, 2016.

[29] A. Cichocki, S. Cruces, and S.-i. Amari, "Log-determinant divergences revisited: Alpha-Beta and Gamma Log-Det divergences," *Entropy*, vol. 17, no. 5, pp. 2988–3034, 2015.

[30] P. Stanitsas, A. Cherian, V. Morellas, and N. Papanikolopoulos, "Clustering positive definite matrices by learning information divergences," in *Manifold Learning: From Euclid to Riemannian, IEEE International Conference on Computer Vision Workshop*, 2017.

[31] A. Cherian, P. Stanitsas, M. Harandi, V. Morellas, and N. Papanikolopoulos, "Learning discriminative $\alpha\beta$-divergences for positive definite matrices," in *ICCV*, 2017.

[32] A. Cichocki and S.-i. Amari, "Families of Alpha-Beta- and Gamma-Divergences: Flexible and robust measures of similarities," *Entropy*, vol. 12, no. 6, pp. 1532–1568, 2010.

[33] S.-i. Amari and H. Nagaoka, *Methods of information geometry*. American Mathematical Soc., 2007, vol. 191.

[34] A. Basu, I. R. Harris, N. L. Hjort, and M. Jones, "Robust and efficient estimation by minimising a density power divergence," *Biometrika*, vol. 85, no. 3, pp. 549–559, 1998.

[35] J. Lafferty, "Additive models, boosting, and inference for generalized divergences," in *Computational Learning Theory*, 1999.

[36] A. Cichocki, R. Zdunek, A. H. Phan, and S.-i. Amari, *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*. John Wiley & Sons, 2009.

[37] R. Kompass, "A generalized divergence measure for nonnegative matrix factorization," *Neural Computation*, vol. 19, no. 3, pp. 780–791, 2007.

[38] I. S. Dhillon and S. Sra, "Generalized nonnegative matrix approximations with bregman divergences," in *NIPS*, 2005.

[39] G. Hinton and S. Roweis, "Stochastic neighbor embedding," in *NIPS*, 2002.

[40] M. Mihoko and S. Eguchi, "Robust blind source separation by beta divergence," *Neural Computation*, vol. 14, no. 8, pp. 1859–1886, 2002.

[41] R. Sivalingam, V. Morellas, D. Boley, and N. Papanikolopoulos, "Metric learning for semi-supervised clustering of region covariance descriptors," in *ICDSC*, 2009.

[42] D. B. Thiyam, S. Cruces, J. Olias, and A. Cichocki, "Optimization of Alpha-Beta Log-Det divergences and their application in the spatial filtering of two class motor imagery movements," *Entropy*, vol. 19, no. 3, p. 89, 2017.

[43] H. Q. Minh, "Alpha-Beta Log-Determinant divergences between positive definite trace class operators," *Information Geometry*, pp. 1–76, 2019.

[44] A. Cherian and S. Sra, "Riemannian dictionary learning and sparse coding for positive definite matrices," *IEEE Transactions on Neural Networks and Learning Systems*, 2016.

[45] R. Sivalingam, D. Boley, V. Morellas, and N. Papanikolopoulos, "Tensor sparse coding for region covariances," in *ECCV*, 2010.

[46] Y. Xie, J. Ho, and B. Vemuri, "On a nonlinear generalization of sparse coding and dictionary learning," in *ICML*, 2013, p. 1480.

[47] M. Harandi and M. S. F. Porikli, "Bregman divergences for infinite dimensional covariance matrices," in *CVPR*, 2014.

[48] P. Li, Q. Wang, W. Zuo, and L. Zhang, "Log-Euclidean kernels for sparse representation and dictionary learning," in *ICCV*, 2013.

[49] M. Yin, Y. Guo, J. Gao, Z. He, and S. Xie, "Kernel sparse subspace clustering on symmetric positive definite manifolds," in *CVPR*, 2016.

[50] V. M. Patel and R. Vidal, "Kernel sparse subspace clustering," in *ICIP*, 2014.

[51] A. Goh and R. Vidal, "Clustering and dimensionality reduction on Riemannian manifolds," in *CVPR*, 2008.

[52] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[53] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *NIPS*, 2002.

[54] A. Cherian, V. Morellas, and N. Papanikolopoulos, "Bayesian nonparametric clustering for positive definite matrices," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 5, pp. 862–874, 2016.

[55] R. Subbarao and P. Meer, "Nonlinear mean shift over Riemannian manifolds," *International Journal of Computer Vision*, vol. 84, no. 1, pp. 1–20, 2009.

[56] B. Pelletier, "Kernel density estimation on Riemannian manifolds," *Statistics & Probability Letters*, vol. 73, no. 3, pp. 297–304, 2005.

[57] U. Şimşekli, A. T. Cemgil, and B. Ermiş, "Learning mixed divergences in coupled matrix and tensor factorization models," in *ICASSP*, 2015.

[58] O. Dikmen, Z. Yang, and E. Oja, "Learning the information divergence," *PAMI*, vol. 37, no. 7, pp. 1442–1454, 2015.

[59] Z. Wang and B. C. Vemuri, "An affine invariant tensor dissimilarity measure and its applications to tensor-valued image segmentation," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1. IEEE, 2004.

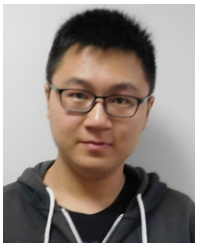
[60] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

[61] E. G. Birgin, J. M. Martínez, and M. Raydan, "Algorithm 813: SPG-software for convex-constrained optimization," *ACM Transactions on Mathematical Software (TOMS)*, vol. 27, no. 3, pp. 340–349, 2001.

[62] G. H. Golub and C. F. Van Loan, *Matrix computations*. JHU Press, 2012, vol. 3.
[63] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. J. Black, "Towards understanding action recognition," in *ICCV*, 2013.
[64] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "HMDB: a large video database for human motion recognition," in *ICCV*, 2011.
[65] P. Mallikarjuna, A. T. Targhi, M. Fritz, E. Hayman, B. Caputo, and J.-O. Eklundh, "The KTH-TIPS2 database," 2006.
[66] T. Ojala, M. Pietikäinen, and D. Harwood, "A comparative study of texture measures with classification based on featured distributions," *Pattern Recognition*, vol. 29, no. 1, pp. 51–59, 1996.
[67] G. Kylberg, M. Uppström, K. Hedlund, G. Borgefors, and I. Sintorn, "Segmentation of virus particle candidates in transmission electron microscopy images," *Journal of Microscopy*, vol. 245, no. 2, pp. 140–147, 2012.
[68] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view RGB-D object dataset," in *2011 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2011, pp. 1817–1824.
[69] P. Stanitsas, A. Cherian, X. Li, A. Truskinovsky, V. Morellas, and N. Papanikolopoulos, "Evaluation of feature descriptors for cancerous tissue recognition," in *ICPR*, 2016.
[70] A. Cherian, P. Koniusz, and S. Gould, "Higher-order pooling of CNN features via kernel linearization for action recognition," *CoRR*, vol. abs/1701.05432, 2017.
[71] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *NIPS*, 2014.
[72] D. Fehr, "Covariance based point cloud descriptors for object detection and classification," Ph.D. dissertation, University Of Minnesota, 2013.
[73] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *Journal of Machine Learning Research*, vol. 9, no. Aug, pp. 1871–1874, 2008.

Anoop Cherian is a Principal Research Scientist with Mitsubishi Electric Research Labs (MERL), Cambridge, MA and an Adjunct Researcher with the Australian National University. He received his M.S. and Ph.D. degrees in computer science from the University of Minnesota, Minneapolis in 2010 and 2013, respectively. He was a postdoctoral researcher in the LEAR group at Inria, Grenoble from 2012-2015, and a Research Fellow at the Australian National University from 2015-2017. Anoop has broad interests in machine learning, deep learning, and computer vision, and has authored more than 50 scientific articles. He is also the recipient of several awards, including the Best Student Paper Award at ICIP, 2012.

Panagiotis Stanitsas received his Diploma of Engineering in Civil Engineering from the University of Patras, Greece in 2010. He received his M.S. in Project Management and Transportation from the University of Patras, Greece in 2011 and his M.Sc in Civil Engineering from the University of Minnesota in 2013. He received his Ph.D. from the department of Computer Science and Engineering at the University of Minnesota in 2017. His research interests include machine learning and computer vision with emphasis on active schemes for learning. He was awarded with an award for excellence in Transportation from the University of Patras in 2010, while he has also received the interdisciplinary Matthew J. Huber award of excellence at the University of Minnesota in 2013.



Jue Wang is a PhD student with the Research School of Engineering at the Australian National University since 2016. He is also associated with CSIRO's Data61 in Australia. From 2010-2014, he received his double bachelor degree (honors) in Electronic Engineering from Australian National University and Beijing Institute of Technology. His research interest are in the area of computer vision and machine learning.

Mehrtash Harandi is a Senior Lecturer with the Department of Electrical and Computer Systems Engineering at Monash University. He is also a contributing research scientist in the Machine Learning Research Group (MLRG) at Data61/CSIRO and an associated investigator at the Australian Center for Robotic Vision (ACRV). His current research interests include theoretical and computational methods in machine learning, computer vision, signal processing, and Riemannian geometry.

Vassilios Morellas received his diploma degree in Mechanical Engineering from the National Technical University of Athens, Greece, his MSME degree from Columbia University, NY and his Ph.D. degree from the department of Mechanical Engineering at the University of Minnesota. He is Research Professor in the department of Electrical and Computer Engineering and Executive Director of the NSF Center for Robots and Sensors for the Human Well-Being. His research interests are in the area of geometric image processing, machine learning, robotics and sensor integration.

Nikos Papanikolopoulos (IEEE Fellow) received his Diploma of Engineering in Electrical and Computer Engineering, from the National Technical University of Athens in 1987. He received his M.S. in 1988 and Ph.D. in 1992 in Electrical and Computer Engineering from Carnegie Mellon University. His research interests include computer vision, robotics, sensors for transportation and precision agriculture applications, and control systems. He is the Director of the Minnesota Robotics Institute and the McKnight Presidential Endowed Professor at the University of Minnesota. He has also received numerous awards including the 2016 IEEE RAS George Saridis Leadership Award in Robotics and Automation.