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 similarities 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 $\alpha$ and $\beta$, 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 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 $\alpha$ and $\beta$, 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 $\alpha, \beta \to 0$ 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 $\alpha$ and $\beta$. 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 $\alpha$ and $\beta$. 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 $\alpha$-family of divergences [33] (that includes popular measures such as the KL-divergence, Jensen-Shannon divergence, and the chi-square divergence) and the $\beta$-family [34] (including the squared Euclidean distance and the Itakura Saito distance). Against several standard measures for computing similarities, both $\alpha$ and $\beta$ 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], 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 [14], [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
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 \([59, 60]\). 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 \([55, 56]\) 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 \(\alpha\) and \(\beta\) jointly.

**Notation:** Following standard practice, we use upper case for matrices (such as \(X\)), lower-bold case for vectors \(x\), and lower case for scalars \(x\). Further, \(S^n_{++}\) is used to denote the cone of \(d \times d\) SPD matrices. We use \(B\) 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 \(\text{diag}\) for the diagonalization operator. In addition, we use \(C\) 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 \(\Pi = \{\pi_1, \ldots, \pi_k\}\) to denote a clustering of data into \(k\) partitions; \(\pi_i\) is the \(i\)-th partition and comprises a subset of the dataset assigned to this cluster.

### 3 Background

In this section, we set up 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.

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

\[
D^{(\alpha, \beta)}(X \parallel Y) = \frac{1}{\alpha \beta} \log \det \left( \frac{\alpha (XY^{-1})^\beta + \beta (XY^{-1})^{-\alpha}}{\alpha + \beta} \right),
\]

\(\alpha \neq 0, \beta \neq 0\) and \(\alpha + \beta \neq 0\). \(\quad (1)\)

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

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

\(\quad (3)\)

This formulation will come handy when deriving the gradient updates for \(\alpha\) and \(\beta\) 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 \(\alpha\) and \(\beta\).

#### 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. When both $\alpha$, $\beta$ are simultaneously positive or negative, this quantity is positive-definite. However, when $\alpha$ and $\beta$ have different signs, the log det term may become degenerate, the necessary conditions to avoid which are stipulated by the following theorem.

**Theorem 1.** For $X, Y \in S^d_{++}$, if $\lambda_i$ is the $i$-th eigenvalue of $XY^{-1}$, then $D^{(\alpha,\beta)}(X \parallel Y) \geq 0$ only if

\[
\lambda_i > \left| \frac{\alpha}{\beta} \right|, \text{ for } \alpha > 0 \text{ and } \beta < 0, \text{ or }
\]

\[
\lambda_i < \left| \frac{\beta}{\alpha} \right|, \text{ for } \alpha < 0 \text{ and } \beta > 0, \forall i = 1, 2, \cdots, d. \tag{5}
\]

**Proof** See [29].

As the algorithms we introduce in the sequel use sets of input SPD matrices, it may be computationally challenging to impose the restrictions in Theorem 1 on the learned $\alpha$ and $\beta$ for every matrix pair. Thus, we restrict our scope in this paper to the form of ABLD when both $\alpha$ and $\beta$ 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 $A \in \mathbb{R}^{d \times d}$, 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 Riemannian symmetric space; this metric is often found to be better in restoration, interpolation, and filtering of DTMRI images [7].

**Identity of Indiscernibles:** For any $\alpha, \beta$, 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 $\alpha, \beta$:** Assuming $\alpha, \beta$ have the same sign, except at the origin ($\alpha = \beta = 0$), ABLD is smooth everywhere with respect to $\alpha$ and $\beta$, 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 $\alpha$ to the one on $\beta$ 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.

### 4.1 Information Divergence & Dictionary Learning

Suppose we are given a dataset of SPD matrices $X = \{X_1, X_2, \cdots, X_N\}$, each $X_i \in S^d_{++}$ and their associated class labels $y_i \in \mathcal{L} = \{1, 2, \cdots, \ell\}$. Our goal is to amalgamate three learning pursuits, namely, (i) learn a dictionary $B \in S^d_{++ \times n}$, 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 $B$ 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.

\[
\text{IDDL} := \min_{B_{k>0}, W_{\alpha>0,\beta>0}} \sum_{i=1}^{N} f(v_i, y_i; W) \tag{10}
\]

subject to $v_k^i = D^{(\alpha_k,\beta_k)}(X_i \parallel B_k)$, where $k$-th dictionary atom in $B$ is denoted by $B_k$, the vector $v_i \in \mathbb{R}^n$ denotes the encoding of $X_i$ in terms of the dictionary, and $v_k^i$ is the $k$-th dimension of this encoding. Specifically, the $k$-th dimension of $v_i$ captures the distance of $X_i$ to the dictionary atom $B_k$ via the respective divergence $D^{(\alpha_k,\beta_k)}(X_i \parallel B_k)$. Note that, in our formulations, we assume $\alpha, \beta$ are non-negative.
to avoid degenerate solutions during optimization caused by the constraint in (3). 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 \( v_i \) 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 Ridge Regression Loss

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

\[
f_1(v_i, y_i; W) = \frac{1}{2} \| h_{i} - Wv_i \|^2 + \gamma \| W \|^2_F ,
\]

where \( \gamma \) is a regularization parameter.

4.2 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 \( \Delta \) denote a control variable on the margins of the classifiers and let \( g(v_i, W) = Wv_i + b \), for a bias \( b \in \mathbb{R}^{L \times 1} \), we define \( f_2 \) as:

\[
f_2(v_i, y_i; W) = \sum_{l \neq y_i} \left( \max(0, g(v_i, W)_l - g(v_i, W)_{y_i} + \Delta) \right) + \gamma \| W \|^2_F ,
\]

where \( \gamma \) is a regularization parameter, and \( g(v_i, W)_{y_i} \) 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 \( v \) to have \( n+1 \) dimensions, where the last dimension is a constant with value one.

4.4 Information Divergence & Clustering

In the clustering setup, our objective is to partition a set of SPD matrices \( \mathcal{X} = \{X_1, X_2, \cdots, X_K\} \) into \( k \) partitions, for a given \( k \). Let \( \Pi = \{\pi_1, \cdots, \pi_k\} \) denote a partitioning of \( \mathcal{X} \) where \( \pi_i \) is the set of samples assigned to the \( i \)-th cluster and let \( C_i \) be the respective cluster centroid. We cast the joint information divergence learning and clustering (IDC) problem as:

\[
\text{IDC} := \min_{\mathbf{c}, \Pi, \alpha, \beta > 0} f_3(\Pi, X; \alpha, \beta) + \Omega(\alpha, \beta),
\]

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

\[
f_3(\Pi, X; \alpha, \beta) = \sum_{\pi \in \Pi} \sum_{i \in \pi} \left( D(\alpha, \beta)(X_i \parallel C_\pi) \right).
\]

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:

\[
\text{minimize } \mathcal{L}(B) \quad \text{s.t. } B \in S_+^{d}.
\]

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 \([4, 44]\).

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 \( S_+^d \).

\[
S_+^d \triangleq \{X \in \mathbb{R}^{d \times d} : v^T X v > 0, \forall v \in \mathbb{R}^d - \{0_d\} \}.
\]

To minimize (15), RCG starts from an initial solution \( B(0) \) and improves its solution using the update rule:

\[
B(t+1) = \tau_B(t) (P(t) ),
\]

where \( P(t) \) identifies a search direction and \( \tau_B(\cdot) : T_B S_+^d \rightarrow S_+^d \) 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 \( S_+^d \) and has a lower objective. The search direction \( P(t) \) in \( T_B^{(0)} S_+^d \) is obtained by:

\[
P(t) = -\nabla \mathcal{L}(P(t)) + \eta(t) \pi (P(t-1), B(t-1), B(t) ).
\]

Here, \( \eta(t) \) can be thought of as a variable learning rate, obtained via techniques such as Fletcher-Reeves \([60]\). Furthermore, \( \nabla \mathcal{L}(B) \) is the Riemannian gradient of the objective function at \( B = \pi(P, X, Y) \) 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 \( S_+^d \).

| Riemannian gradient | Retraction. | Parallel Transport. |
|---------------------|-------------|---------------------|
| \( \text{grad } \mathcal{L}(B) = B \text{sym}(\nabla_B(B))B \) | \( \tau_B(\xi) = B^\frac{3}{2} \text{Exp}(B^{-\frac{1}{2}} \xi B^{-\frac{1}{2}}) B^\frac{3}{2} \) | \( \pi(P, X, Y) = ZPZ^T \) |

\[
\text{TABLE 2: Riemannian tools to perform RCG on } S_+^d. \text{ Here, } \text{sym}(X) = \frac{1}{2}(X + X^T), \text{Exp}(\cdot) \text{ denotes the matrix exponential and } Z = (XY^{-1})^{\frac{3}{2}}.
\]
5.2 Learning ABLDs via Block Coordinate Descent

We propose to use a block-coordinate descent (BCD) scheme to optimize our objectives in (19) and (23). In this scheme, each variable is updated alternately, while fixing others. We use the Riemannian conjugate gradient (RCG) algorithm [60] for optimizing the dictionary atoms in IIDL (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 IIDDL 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 \( \alpha \) and \( \beta \) are either positive or negative is essentially the same, in the derivations to follow, we assume \( \alpha \) and \( \beta \) belong to the non-negative orthant of the Euclidean space. We use the spectral projected gradient (SPG) descent algorithm [61] for learning \( \alpha \) and \( \beta \). This algorithm uses Barzilai-Borwein step-size selection in the gradient descent, and projects the iterates into the non-negative orthant (or to the non-positive orthant if \( \alpha, \beta < 0 \)) to enforce the constraints. Empirically, using SPG is seen to converge very quickly, as is also observed in [44].

![Algorithm 1: Block-Coordinate Descent for IIDDL and Ridge Regression Loss.](image)

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 \( \alpha \) and \( \beta \). 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.

5.3.1 Derivative of ABLD wrt \( \alpha \)

Towards computing the derivative of ABLD with respect to its parameter \( \alpha \), 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 \( X, Y \in S^d_+ \), the derivative has the form:

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

Using the dual symmetry property of ABLD reviewed in [9], derivative of ABLD with respect to \( \beta \) 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 ABLD with respect to matrix \( Y \) has the form:

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

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

**Theorem 2.** For \( A, B \in S^d_+ \) and \( p, q \geq 0 \),

\[
\nabla_{B} \log \det \left[ p (AB)^q + I_d \right] = \rho q B^{-\frac{1}{2}} (A^\frac{q}{2} BA^\frac{q}{2})^{-1} (I_d + p (A^\frac{q}{2} BA^\frac{q}{2}))^{-\frac{1}{2}} A^\frac{q}{2}.
\]

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

\[
\nabla_{Y} D^{(\alpha,\beta)}(X \parallel Y) = \rho \theta Y^{-\frac{1}{2}} (Z^\frac{1}{2} Y Z^\frac{1}{2})^\theta \times (I_d + \rho (Z^\frac{1}{2} Y Z^\frac{1}{2})^\theta)^{-1} Z^\frac{1}{2} + \frac{1}{\beta} Y^{-1}. \tag{21}
\]

Input: \( X, n \)
\( B \leftarrow \log-euc-kmeans(X, n), (\alpha, \beta) \leftarrow \text{GridSearch}; \)
repeat
\( \text{for } k = 1 \text{ to } n \) do
\( B_k \leftarrow \text{update}_B(X, W, \alpha, \beta, B_k); \) // use (27)
end
\( (\alpha, \beta) \leftarrow \text{update}_{\alpha\beta}(X, W, B, \alpha, \beta); \) // use (28)
\( W \leftarrow \text{update}_W; \) // use (25)
until until convergence;
return \( B, \alpha, \beta \)

Algorithm 2: Block-Coordinate Descent for IIDDL Structured SVM Loss.
**Remark 1.** Computing \((21)\) for large datasets can become overwhelming. A more efficient implementation could be the following. Say, \((U, \Delta)\) be the Schur decomposition of \(Z \frac{1}{2} Y Z \frac{1}{2}\), which is faster to compute than the eigenvalue decomposition \((22)\) (required for computing \(Z \frac{1}{2}\)). With \(\delta = \text{diag}(\Delta)\), \((21)\) can be rewritten as:

\[
\nabla_{Y} D^{(\alpha, \beta)}(X \parallel Y) = \rho \theta Y^{-1} \left(Z^{-\frac{1}{2}} U \right \text{diag} \left( \frac{\delta \theta}{1 + \rho \delta \theta} \right) \right) \times \left(Z^{-\frac{1}{2}} U \right)^{-1} + \frac{1}{\beta} Y^{-1}.
\]

(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,

\[
D^{(0,0)}(X \parallel Y) = \left\| \text{Log} \left(X^{-\frac{1}{2}} Y X^{-\frac{1}{2}} \right) \right\|_{F}^{2}.
\]

(23)

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

\[
\nabla_{Y} D^{(0,0)}(X \parallel Y) = 2X^{-\frac{1}{2}} (\log P) P^{-1} X^{-\frac{1}{2}}.
\]

(24)

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

With this general gradients for learning \(\alpha, \beta, \text{dictionary and the cluster centroids, now we consider learning parameters specific to each of our objectives.}\)

### 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 \(v_{i}\) and \(h_{i}\) along their \(i\)-th column, for \(i = 1, 2, \ldots, N\). When fixing \(B, \alpha\) and \(\beta\), the objective can be solved in closed form as:

\[
W^{*} = HV^{T}(VV^{T} + \gamma I_{d})^{-1},
\]

(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 \(v_{i}\) involves \(B_{k}\). To simplify the notation, let us assume:

\[
\zeta_{i} = -(h_{i} - Wv_{i})^{T} W,
\]

(26)

and let \(\zeta_{i}^{k}\) be its \(k\)-th dimension. Then we have:

\[
\nabla_{B_{k}} f_{1} = \zeta_{i}^{k} \nabla_{B_{k}} \left( D^{(\alpha^{k}, \beta^{k})}(X_{i} \parallel B_{k}) \right),
\]

(27)

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

\[
\nabla_{\alpha_{k}} f_{1} = \zeta_{i}^{k} \nabla_{\alpha_{k}} \left( D^{(\alpha^{k}, \beta^{k})}(X_{i} \parallel B_{k}) \right).
\]

(28)

It should be noted that the gradients w.r.t. \(\beta_{k}\) 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 \(B, \alpha\), and \(\beta\) we get:

\[
\nabla_{w_{y_{i}}} f_{2} = -\left( \sum_{j \neq y_{i}} I(w_{j}^{T} v_{i} - w_{y_{i}}^{T} v_{i} + \Delta > 0) \right) v_{i} + 2\gamma w_{y_{i}}.
\]

(29)

\[
\nabla_{w_{j \neq y_{i}}} f_{2} = \left( w_{j}^{T} v_{i} - w_{y_{i}}^{T} v_{i} + \Delta > 0 \right) v_{i} + 2\gamma w_{j}.
\]

(30)

where \(I(\cdot)\) is the indicator function. Similarly to our derivations for \(f_{1}\), to simplify our notations we let:

\[
\xi_{i} = -\sum_{j \neq y_{i}} I(w_{j}^{T} v_{i} - w_{y_{i}}^{T} v_{i} + \Delta > 0) (w_{j} - w_{y_{i}}).
\]

(31)

and let \(\xi_{i}^{k}\) be its \(k\)-th dimension. That way, making use of \((22)\) and \((19)\), we define the gradients of \(f_{2}\) w.r.t. \(B_{k}\) and \(\alpha_{k}\), respectively as:

\[
\nabla_{B_{k}} f_{2} = \xi_{i}^{k} \nabla_{B_{k}} \left( D^{(\alpha^{k}, \beta^{k})}(X_{i} \parallel B_{k}) \right),
\]

(32)

\[
\nabla_{\alpha_{k}} f_{2} = \xi_{i}^{k} \nabla_{\alpha_{k}} \left( D^{(\alpha^{k}, \beta^{k})}(X_{i} \parallel B_{k}) \right).
\]

(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 \(\Pi\) in \((14)\), we need to find the cluster centroid \(C_{\pi}\) 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 \(\pi\) are updated as \(\pi_{z} \to \pi_{z} \cup \{X_{i}\}\), where:

\[
\pi_{z} = \arg \min_{z \in \{1, 2, \ldots, k\}} D^{(\alpha, \beta)}(X_{i} \parallel C_{\pi}).
\]

(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 \(O(Nd^{3})\) flops. Using the gradient formulation in \((19)\) for \(\alpha\) and \(\beta\), we need \(O(Ndn + N d^{3})\) flops. Computations of the closed form for \(W\) using the ridge regression loss in \((25)\) takes \(O(n^{2}(L+N)+n^{3}+nLN)\). 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 \(O(nd^{3})\) flops, which is similar in complexity to the recent sparse coding schemes such as \((14)\). As for the gradient computation for each \(C\) in IDC takes \(O(Nd^{3})\) flops and the overall clustering setup takes \(O(Ndk + N d^{3})\) 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 $102 \times 102$ for HMDB and $42 \times 42$ 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 $18 \times 18$. 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 $23 \times 23$ 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 $640 \times 640$ resolution. To produce the covariance descriptors, we follow the procedure outlined in [44]. Specifically, we extracted $32 \times 32$ 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 $5 \times 5$ 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 $29 \times 29$ 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 $8 \times 8$. 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 $\alpha$ and $\beta$, 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 $\gamma$ in IDDL was chosen using cross-validation, while the regularization parameter $\mu$ 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 $\alpha$ and $\beta$ in ABLD. This forms the basis of all our further experiments.

6.4.1 Performance for Varying $\alpha$, $\beta$

In Figure 4(a) and 4(b), we plot a heatmap of the classification accuracy against changing $\alpha$ and $\beta$ 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 $\alpha$ and
β that is shared across all the dictionary atoms (which we call IDDL-S), (ii) a vector α and β, where we assume α = β, 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 (α = β = 1) and the AIRM case (α = β = 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.

| Atoms | Method | IDDL-Fix(α, β) | IDDL-Fix(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 6, 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 5$C$ 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 6 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 (JBLD) [47], (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 HMBD. On the HMDB dataset, we found that training an additional lib-linear SVM solver 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) presents 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 that the two variants of $\alpha \beta$-KMeans consistently outperform the competing schemes underlying the merits of learning the measure while clustering the data.

**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                | 58.8% | 55.97%     | 44.40%      | 57.46% | 57.5% | 67.10% | 68.3% | 67.20% | 61.19% | 61.01% |
| Viral DB             | 52.15%| 44.9%      | 28.43%      | 40.20% | 21.0% | 52.30% | 57.3% | 43.20% | 46.94% |
| BRODATZ             | 74.60%| 68.00%     | 57.84%      | 70.91% | 60.8% | 76.48% | 77.74% | 79.44% | 78.33% | 78.37% |
| KTH TIPS            | 47.15%| 55.00%     | 65.19%      | 70.00% | 74.9% | 72.50% | 73.2% | 77.10% | 62.63% | 79.44% |
| 3D Object           | 79.25%| 77.18%     | 69.92%      | 73.96% | 64.5% | 78.68% | 79.37% | 79.67% | 78.80% | 78.36% |
| 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.

| Dataset/Method       | Ridge Regression | Structured-SVM |
|----------------------|------------------|----------------|
| JHMDB                | IDDL-V | 68.3% | 67.2% | 69.2% | 65.8% |
| Viral DB             | IDDL-N | 57.3% | 58.6% | 56.7% | 55.8% |
| BRODATZ             | 57.4% | 55.00% | 65.19% | 70.00% | 74.9% | 72.50% | 73.2% | 77.10% | 62.63% | 79.44% |
| 3D Object           | 94.1% | 92.3% | 97.1% | 73.7% | 98.2% |
| Breast Cancer       | 90.8% | 88.0% | 78.1% | 76.7% |
| Myometrium Cancer   | 93.3% | 90.2% | 89.8% | 89.2% |

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

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

| Dataset/Method       | Karcher | $\alpha \beta$-E | $\alpha \beta$-NE |
|----------------------|---------|------------------|------------------|
| JHMDB                | 0.248   | 0.252            | 0.257            |
| BRODATZ             | 0.353   | 0.366            | 0.378            |
| KTH TIPS            | 0.379   | 0.400            | 0.429            |
| Prostate Cancer     | 0.578   | 0.594            | 0.679            |
| Myometrium Cancer   | 0.737   | 0.661            | 0.778            |

**TABLE 8:** 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.

**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
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.

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 $A^\frac{1}{2}B A^\frac{1}{2}$, 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,

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

Using Taylor series expansion of Log:

$$\log \left[ p(SBS)^q + I_d \right] = \text{Tr} \left( p(SBS)^q - \frac{p^2(SBS)^{2q}}{2} - \cdots \right).$$

$$\nabla_B \left[ p(SBS)^q \right] = pqS(SBS)^{q-1}S - q^2(SBS)^{2q-1}S - \cdots = pqS(SBS)^{-1}(SBS)^q(I_d - p(SBS)^q + \cdots) S$$

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

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

substituting for which we get our desired result. Note that, the series expansions we use in the proof are valid only when $p \norm{SBS}_2 \leq 1$, which can be achieved via rescaling or normalizing our data.
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 has also received numerous 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.