Riemannian Sparse Coding for Positive Definite Matrices

Anoop Cherian[1] and Suvrit Sra[2]

[1] LEAR team, Inria Grenoble Rhône-Alpes, France
[2] Max Planck Institute for Intelligent Systems, Tübingen, Germany

Abstract. Inspired by the great success of sparse coding for vector valued data, our goal is to represent symmetric positive definite (SPD) data matrices as sparse linear combinations of atoms from a dictionary, where each atom itself is an SPD matrix. Since SPD matrices follow a non-Euclidean (in fact a Riemannian) geometry, existing sparse coding techniques for Euclidean data cannot be directly extended. Prior works have approached this problem by defining a sparse coding loss function using either extrinsic similarity measures (such as the log-Euclidean distance) or kernelized variants of statistical measures (such as the Stein divergence, Jeffrey's divergence, etc.). In contrast, we propose to use the intrinsic Riemannian distance on the manifold of SPD matrices. Our main contribution is a novel mathematical model for sparse coding of SPD matrices; we also present a computationally simple algorithm for optimizing our model. Experiments on several computer vision datasets showcase superior classification and retrieval performance compared with state-of-the-art approaches.

Keywords: Sparse coding, Riemannian distance, Region covariances.

1 Introduction

Symmetric positive definite matrices—in the form of region covariances [1]—play an important role as data descriptors in several computer vision applications. Notable examples where they are used include object recognition [2], face recognition [3], human detection and tracking [4,5], visual surveillance [6], 3D object recognition [7], among others. Compared with popular vectorial descriptors, such as bag-of-words, Fischer vectors, etc., the second-order structure that covariance matrices offer makes them particularly appealing. For instance, covariance descriptors offer a convenient platform for fusing multiple features into a compact form independent of the number of data points. By choosing appropriate features, this fusion can be made invariant to image affine distortions [8], or robust to static image noise and illumination variations, while generating these matrices remains efficient using integral image transforms [4].

In this paper, we study SPD matrices in the context of sparse coding. The latter is now an important, established tool in signal processing and computer vision: it helps understand the inherent structure of the data [9,10], leading to

D. Fleet et al. (Eds.): ECCV 2014, Part III, LNCS 8691, pp. 299–314, 2014.
© Springer International Publishing Switzerland 2014

state-of-the-art results for a variety of vision applications [11,12,13]. Given an input data point and an overcomplete dictionary of basis atoms, Euclidean sparse coding seeks a representation of this point as sparse linear combination of atoms so that a squared Euclidean loss is minimized. Formally, if $\mathcal{B}$ is the dictionary and z the input data point, generic sparse coding may be formulated as

$$\min_\theta \quad \mathcal{L}(z, \mathcal{B}, \theta) + \lambda \operatorname{Sp}(\theta), \tag{1}$$

where the loss function $\mathcal{L}$ measures reconstruction accuracy obtained by using the "code" θ, while λ regulates the impact of the sparsity penalty $\operatorname{Sp}(\theta)$.

Sparse coding has found great success for vector valued data, so it is natural to hope for similar benefits when applying it to the more complex setting of data represented via SPD matrices. However, applying sparse coding to SPD matrices is not straightforward, a difficulty that arises primarily because SPD matrices form a curved Riemannian manifold of negative sectional curvature (so that distances along the manifold are lower than corresponding Euclidean distances). As a result, this manifold cannot be isometrically embedded into Euclidean space through operations such as mere vectorization, without introducing embedding errors. Such errors can affect the application performance [14,4]. On the other hand, computing distances and solving optimization problems on the SPD manifold is computationally demanding (see Section 3). Thus care must be taken to select an appropriate loss function.

The main goal of this paper is to study sparse coding of SPD matrices in their native Riemannian geometric context by using a dictionary comprised of SPD matrices as atoms. Towards this end, we make the following contributions.

- *Formulation:* We propose a novel model that finds nonnegative sparse linear combinations of SPD atoms from a given dictionary to well-approximate an input SPD matrix. The approximation quality is measured by the squared intrinsic Riemannian distance. As a theoretical refinement to our model, we describe a surprising but intuitive geometric constraint under which the nonconvex Riemannian sparse coding task actually becomes convex.
- *Optimization:* The main challenge in using our formulation is its higher computational cost relative to Euclidean sparse coding. However, we describe a simple and effective approach for optimizing our objective function.
- *Experiments:* We present results on a few computer vision tasks on several state-of-the-art datasets to demonstrate superior performance obtained by using our new sparse coding model.

To set the stage for presenting our contributions, we first survey some recent methods suggested for sparse coding. After that we review key tools from Riemannian geometry that we will use to develop our ideas. Throughout we work with real matrices. The space of $d \times d$ SPD matrices is denoted as $\mathcal{S}_+^d$, symmetric matrices by $\mathcal{S}^d$, and the space of (real) invertible matrices by $\mathrm{GL}(d)$. By $\operatorname{Log}(X)$, for $X \in \mathcal{S}_+$, we mean the principal matrix logarithm.

2 Related Work

Sparse coding of SPD matrices has recently received a significant attention in the vision community due to the performance gains that it brings to the respective applications. As alluded to earlier, the manifold geometry hinders a straightforward extension of classical sparse coding techniques to these objects. Prior methods typically use one of the following proxies: (i) rather than Riemannian geometry, use an information geometric perspective using an appropriate statistical measure; (ii) map the matrices into a flat Riemannian symmetric space; or (iii) use a kernelizable similarity measure to embed the matrices into an RKHS. We briefly review each of these schemes below.

Statistical measures. In [15], a sparse coding framework is proposed based on the log-determinant divergence (Burg loss) to model the loss function. Their formulation requires sophisticated interior point methods for the optimization, and as a result it is often slow even for moderately large covariances (more than 5×5). In [16], a data matrix is approximated by a sparse linear combination of rank-one matrices under a Frobenius norm based loss. Although this scheme is computationally efficient, it discards the manifold geometry.

Differential geometric schemes. Among the several computationally efficient variants of Riemannian distances, one of the most popular is the log-Euclidean distance d_{le} [17] defined for $X, Y \in \mathcal{S}_+^d$ as $d_{\mathrm{le}}(X, Y) := \|\mathrm{Log}(X) - \mathrm{Log}(Y)\|_{\mathrm{F}}$. The Log operator maps an SPD matrix isomorphically and diffeomorphically into the flat space of symmetric matrices; the distances in this space are Euclidean. Sparse coding with the squared log-Euclidean distance has been proposed in the past [18] with promising results. A similar framework was suggested recently [19] in which a local coordinate system is defined on the tangent space at the given data matrix. While, their formulation uses additional constraints that make their framework coordinate independent, their scheme restricts sparse coding to specific problem settings.

Kernelized Schemes. In [20], a kernelized sparse coding scheme is presented for SPD matrices using the Stein divergence [21] for generating the underlying kernel function. But this divergence does not induce a kernel for all bandwidths. To circumvent this issue [22,23] propose kernels based on the log-Euclidean distance. It is well-known (and also shown in our experiments) that a kernelized sparse coding scheme suffers significantly when the number of dictionary atoms is high.

In contrast to all these methods, our scheme directly uses the intrinsic Riemannian distance to design our sparse reconstruction loss, which is the natural distance for covariances. To circumvent the computational difficulty we propose an efficient algorithm based on spectral projected gradient. Our experiments demonstrate that our scheme is efficient and provides state of the art results on several computer vision problems that use covariance matrices.

3 Preliminaries

We provide below a brief overview of the Riemannian geometry of SPD matrices. An SPD matrix has the property that all its eigenvalues are positive. For an $n \times n$ SPD matrix, such a property restricts it to span only the convex half-cone of the n^2 dimensional Euclidean space of symmetric matrices. A manifold is a Hausdorff space that is locally Euclidean and second-countable. The former property means that there is a distinct neighborhood for every point belonging to this manifold. Second countability suggests that there exists a countable collection of open sets such that every open set is the union of these sets. These properties are often useful for analyzing stationary points for optimization problems on the manifold.

For a point X on the manifold, its tangent space is a vector space consisting of all the tangent vectors at that point. SPD matrices form a differentiable Riemannian manifold, which implies that every point on it has a well-defined continuous collection of scalar products defined on its tangent space and is endowed with an associated Riemannian metric [24, Ch. 6]. This metric provides a measure on the manifold for computing distances between points. As the manifold is curved, this distance specifies the length of the shortest curve that connects the points, i.e., *geodesics*. A manifold is Riemannian if it is locally Euclidean, that is its geodesics are parallel to the tangent vectors.

There are predominantly two operations that one needs for computations on the Riemannian manifold, namely (i) the exponential map $\exp_P : \mathcal{S}^d \to \mathcal{S}^d_+$ and (ii) the logarithmic map $\log_P = \exp_P^{-1} : \mathcal{S}^d_+ \to \mathcal{S}^d$, where $P \in \mathcal{S}^d_+$. While the former projects a symmetric point on the tangent space onto the manifold, the latter does the reverse. Note that these maps depend on the manifold point P at which the tangent spaces are computed. In our analysis, we will be measuring distances assuming P to be the identity matrix[1], I. A popular intrinsic (i.e., distances are computed along the curvature of the manifold) metric on the SPD manifold is the affine invariant *Riemannian distance* [14]:

$$d_{\mathcal{R}}(X, Y) = \left\| \operatorname{Log} X^{-1/2} Y X^{-1/2} \right\|_{\mathrm{F}}. \tag{2}$$

4 Problem Formulation

We are now ready to introduce our new model for sparse coding of SPD matrices. Figure 1 provides a schematic of our sparse coding model on the manifold.

Model. Let $\mathcal{B}$ be a dictionary with n atoms $B_1, B_2, \cdots, B_n$, where each $B_i \in \mathcal{S}^d_+$. Let $X \in \mathcal{S}^d_+$ be an input matrix that must be sparse coded. Our basic sparse coding objective is to solve

[1] As the metric that we use in this paper is affine invariant, such a choice will not distort the geometry of the manifold and is achieved by scaling the SPD matrices by $X^{-1/2}$ on the left and the right.

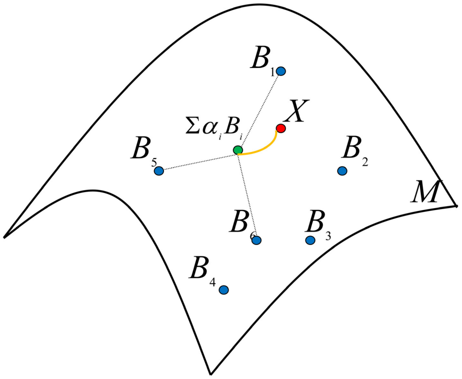


Fig. 1. A schematic illustration of our sparse coding objective formulation. For the SPD manifold M and given SPD basis matrices B_i on the manifold, our objective seeks a non-negative sparse linear combination $\sum_i \alpha_i B_i$ of the B_i's that is closest (in a geodesic sense) to the given input SPD matrix X.

$$\begin{aligned}\min_{\alpha\geq 0} \quad \phi(\alpha) &:= \frac{1}{2} d^2_{\mathcal{R}}\left(\sum\nolimits_{i=1}^n \alpha_i B_i, X\right) + \mathrm{Sp}(\alpha) \\ &= \frac{1}{2}\left\|\mathrm{Log} \sum\nolimits_{i=1}^n \alpha_i X^{-\frac{1}{2}} B_i X^{-\frac{1}{2}}\right\|_{\mathrm{F}}^2 + \mathrm{Sp}(\alpha),\end{aligned} \tag{3}$$

where α_i is the i-th component of α, and Sp is a sparsity inducing function.

Problem (3) measures reconstruction quality offered by a sparse non-negative linear combination of the atoms to a given input point X. It will turn out (see experiments in Section 6) that the reconstructions obtained via this model actually lead to significant improvements in performance over sparse coding models that ignore the rich geometry of SPD matrices. However, this gain comes at a price: model (3) is a difficult nonconvex problem, which remains nonconvex even if we take into account the geodesic convexity of $d_{\mathcal{R}}$.

While in practice this nonconvexity does not seem to impede the use of our model, we show below a surprising but highly intuitive constraint under which Problem 3 actually becomes convex!

Theorem 1. *The function $\phi(\alpha) := d^2_{\mathcal{R}}(\sum_i \alpha_i B_i, X)$ is convex on the set*

$$\mathcal{A} := \{\alpha \mid \sum\nolimits_i \alpha_i B_i \preceq X, \text{ and } \alpha \geq 0\}. \tag{4}$$

Before we prove this theorem, let us intuitively describe what it is saying. While sparsely encoding data we are trying to find sparse coefficients $\alpha_1, \ldots, \alpha_n$, such that in the ideal case we have $\sum_i \alpha_i B_i = X$. But in general this equality cannot be satisfied, and one only has $\sum_i \alpha_i B_i \approx X$, and the quality of this approximation is measured using $\phi(\alpha)$ or some other desirable loss-function. The loss $\phi(\alpha)$ from (3) is nonconvex while convexity is a "unilateral" property—it lives in the world of inequalities rather than equalities [25]. And it is known that SPD matrices in addition to forming a manifold also enjoy a rich conic geometry that is endowed with the Löwner partial order. Thus, instead of seeking arbitrary approximations $\sum_i \alpha_i B_i \approx X$, if we limit our attention to those that underestimate X as in (4), we might benefit from the conic partial order. It is this intuition that Theorem 1 makes precise.

Lemma 1. *Let $Z \in \mathrm{GL}(d)$ and let $X \in \mathcal{S}_+^d$. Then, $Z^T X Z \in \mathcal{S}_+^d$.*

Lemma 2. *The Fréchet derivative [26, see e.g., Ch. 1] of the map $X \mapsto \log X$ at a point Z in the direction E is given by*

$$D\log(Z)(E) = \int_0^1 (\beta Z + (1-\beta)I)^{-1} E (\beta Z + (1-\beta)I)^{-1} d\beta. \tag{5}$$

Proof. This is a classical result, for a proof see e.g., [26, Ch. 11].

Corollary 1. *Consider the map $\ell(\alpha) := \alpha \in \mathbf{R}_+^n \mapsto \mathrm{Tr}(\log(SM(\alpha)S)H)$, where M is a map from $\mathbf{R}_+^n \to \mathcal{S}_+^d$ and $H \in \mathcal{S}^d$, $S \in \mathcal{S}_+^d$. Then, for $1 \le p \le n$, we have*

$$\tfrac{\partial \ell(\alpha)}{\partial \alpha_p} = \int_0^1 \mathrm{Tr}[K_\beta S \tfrac{\partial M(\alpha)}{\partial \alpha_p} S K_\beta H] d\beta,$$

where $K_\beta := (\beta S M(\alpha) S + (1-\beta)I)^{-1}$.

Proof. Simply apply the chain-rule of calculus and use linearity of $\mathrm{Tr}(\cdot)$.

Lemma 3. *The Fréchet derivative of the map $X \mapsto X^{-1}$ at a point Z in direction E is given by*

$$D(Z^{-1})(E) = -Z^{-1} E Z^{-1}. \tag{6}$$

We are now ready to prove Theorem 1.

Proof (Thm. 1). We show that the Hessian $\nabla^2 \phi(\alpha) \succeq 0$ on $\mathcal{A}$. To ease presentation, we write $S = X^{-1/2}$, $M \equiv M(\alpha) = \sum_i \alpha_i B_i$, and let D_q denote the differential operator D_{α_q}. Applying this operator to the first-derivative given by Lemma 4 (in Section 5), we obtain (using the product rule) the sum

$$\mathrm{Tr}\big([D_q \log(SMS)](SMS)^{-1} S B_p S\big) + \mathrm{Tr}\big(\log(SMS) D_q[(SMS)^{-1} S B_p S]\big).$$

We now treat these two terms individually. To the first we apply Corr. 1. So

$$\begin{aligned}
\mathrm{Tr}\big([D_q \log(SMS)](SMS)^{-1} S B_p S\big) &= \int_0^1 \mathrm{Tr}(K_\beta S B_q S K_\beta (SMS)^{-1} S B_p S) d\beta \\
&= \int_0^1 \mathrm{Tr}(S B_q S K_\beta (SMS)^{-1} S B_p S K_\beta \cdot) d\beta \\
&= \int_0^1 \langle \Psi_\beta(p), \Psi_\beta(q) \rangle_M \, d\beta,
\end{aligned}$$

where the inner-product $\langle \cdot, \cdot \rangle_M$ is weighted by $(SMS)^{-1}$ and the map $\Psi_\beta(p) := S B_p S K_\beta$. We find a similar inner-product representation for the second term too. Starting with Lemma 3 and simplifying, we obtain

$$\begin{aligned}
\mathrm{Tr}\big(\log(SMS) D_q[(SMS)^{-1} S B_p S]\big) &= -\mathrm{Tr}\big(\log(SMS)(SMS)^{-1} S B_q M^{-1} B_p S\big) \\
&= \mathrm{Tr}\big(-S \log(SMS) S^{-1} M^{-1} B_q M^{-1} B_p\big) \\
&= \mathrm{Tr}\big(M^{-1} B_p [-S \log(SMS) S^{-1}] M^{-1} B_q\big).
\end{aligned}$$

By assumption $\sum_i \alpha_i B_i = M \preceq X$, which implies $SMS \preceq I$. Since $\log(\cdot)$ is operator monotone [24], it follows that $\log(SMS) \preceq 0$; an application of Lemma 1 then yields $S\log(SMS)S^{-1} \preceq 0$. Thus, we obtain the weighted inner-product

$$\mathrm{Tr}\big(M^{-1}B_p[-S\log(SMS)S^{-1}]M^{-1}B_q\big) = \big\langle M^{-1}B_p,\, M^{-1}B_q\big\rangle_L\,,$$

where $L = [-S\log(SMS)S^{-1}] \succeq 0$, whereby $\langle\cdot,\,\cdot\rangle_L$ is a valid inner-product.

Thus, the second partial derivatives of ϕ may be ultimately written as

$$\frac{\partial^2\phi(\alpha)}{\partial\alpha_p\partial\alpha_q} = \langle \varGamma(B_q),\, \varGamma(B_p)\rangle\,,$$

for some map $\varGamma$ and some corresponding inner-product (the map and the inner-product are defined by our analysis above). Thus, we have established that the Hessian is a Gram matrix, which shows it is semidefinite. Moreover, if the B_i are different ($1 \le i \le n$), then the Hessian is strictly positive definite. □

5 Optimization

We briefly describe below our optimization approach for solving our main problem (3). In particular, we propose to use a first-order method, i.e., a method based on the gradient $\nabla\phi(\alpha)$. The following lemma proves convenient towards this gradient computation.

Lemma 4. *Let B, C, and X be fixed SPD matrices. Consider the function $f(x) = d^2_{\mathcal{R}}(xB + C, X)$. The derivative $f'(x)$ is given by*

$$f'(x) = 2\,\mathrm{Tr}(\log(X^{-1/2}(xB + C)X^{-1/2})X^{1/2}(xB + C)^{-1}BX^{-1/2}). \qquad (7)$$

Proof. Introduce the shorthand $S \equiv X^{-1/2}$ and $M(x) \equiv xB + C$. From definition (2) and using $\|Z\|_{\mathrm{F}}^2 = \mathrm{Tr}(Z^TZ)$ we have

$$f(x) = \mathrm{Tr}(\log(SM(x)S)^T\log(SM(x)S)).$$

Differentiating this the chain-rule of calculus immediately yields

$$f'(x) = 2\,\mathrm{Tr}(\log(SM(x)S)(SM(x)S)^{-1}SM'(x)S),$$

which is nothing but (7). □

Writing $M(\alpha_p) = \alpha_pB_p + \sum_{i\neq p}\alpha_iB_i$ and using Lemma 4 we obtain

$$\frac{\partial\phi(\alpha)}{\partial\alpha_p} = \mathrm{Tr}\Big(\log\big(SM(\alpha_p)S\big)\big(SM(\alpha_p)S\big)^{-1}SB_pS\Big) + \frac{\partial\,\mathrm{Sp}(\alpha)}{\partial\alpha}. \qquad (8)$$

Computing (8) for all α is the dominant cost in a gradient-based method for solving (3). We present pseudocode (Alg. 1) that efficiently implements the gradient for the first part of (8). The total cost of Alg. 1 is $O(nd^2) + O(d^3)$—a naïve implementation of (8) costs $O(nd^3)$, which is substantially more expensive.

```
Input: $B_1, \dots, B_n, X \in \mathcal{S}_+^d$, $\alpha \geq 0$
$S \leftarrow X^{-1/2}$; $M \leftarrow \sum_{i=1}^n \alpha_i B_i$;
$T \leftarrow \log(SMS)(MS)^{-1}$;
for $i = 1$ to $n$ do
    $g_i \leftarrow \mathrm{Tr}(TB_p)$;
end
return $g$
```

Algorithm 1. Subroutine for efficiently computing gradients

For simplicity, in (8) that defines $\phi(\alpha)$ we use the sparsity penalty $\mathrm{Sp}(\alpha) = \lambda\|\alpha\|_1$, where $\lambda > 0$ is a regularization parameter. Since we are working with $\alpha \geq 0$, we replace this penalty by $\lambda \sum_i \alpha_i$, which is differentiable. This allows us to use Alg. 1 in conjunction with a gradient-projection scheme that essentially runs the iteration

$$\alpha^{k+1} \leftarrow \mathscr{P}[\alpha^k - \eta_k \nabla\phi(\alpha^k)], \qquad k = 0, 1, \dots, \tag{9}$$

where $\mathscr{P}[\cdot]$ denotes the projection operator defined as

$$\mathscr{P}[\alpha] \equiv \alpha \mapsto \mathrm{argmin}_{\alpha'} \tfrac{1}{2}\|\alpha' - \alpha\|_2^2, \quad \text{s.t. } \alpha' \geq 0,\ \alpha' \in \mathcal{A}. \tag{10}$$

Iteration (9) has three major computational costs: (i) computing the stepsize η_k; (ii) obtaining the gradient $\nabla\phi(\alpha^k)$; and (iii) computing the projection (10). Alg. 1 shows how to efficiently obtain the gradient. The projection task (10) is a special least-squares (dual) semidefinite program (SDP), which can be solved using any SDP solver or by designing a specialized routine. However, for the sake of speed, we could drop the constraint $\alpha' \in \mathcal{A}$ in practice, in which case $\mathscr{P}[\alpha]$ reduces to the truncation $\max(0, \alpha)$, which is trivial. We stress at this point that developing efficient subroutines for the full projection (10) is rather nontrivial, and a task worthy of a separate research project, so we defer it to the future. It only remains to specify how to obtain the stepsize η_k.

There are several choices available in the nonlinear programming literature [27] for choosing η_k, but most of them can be quite expensive. In our quest for an efficient sparse coding algorithm, we choose to avoid expensive line-search algorithms for selecting η_k and prefer to use the Barzilai-Borwein stepsizes [28], which can be computed in closed form and lead to remarkable gains in performance [28,29]. In particular, we use the Spectral Projected Gradient (SPG) method [30] by adapting a simplified implementation of [29].

SPG runs iteration (9) using Barzilai-Borwein stepsizes with an occasional call to a nonmontone line-search strategy to ensure convergence of $\{\alpha^k\}$. Without the constraint $\alpha' \in \mathcal{A}$, we cannot guarantee anything more than a stationary point of (3), while if we were to use the additional constraint then we can even obtain global optimality for iterates generated by (9).

6 Experiments and Results

In this section, we provide experimental results on simulated and real-world data demonstrating the effectiveness of our algorithm compared to the state-of-the-art methods on covariance valued data. For all the datasets, we will be using the classification accuracy as the performance metric. Our implementations are in MATLAB and the timing comparisons used a single core Intel 3.6GHz CPU.

6.1 Comparison Methods

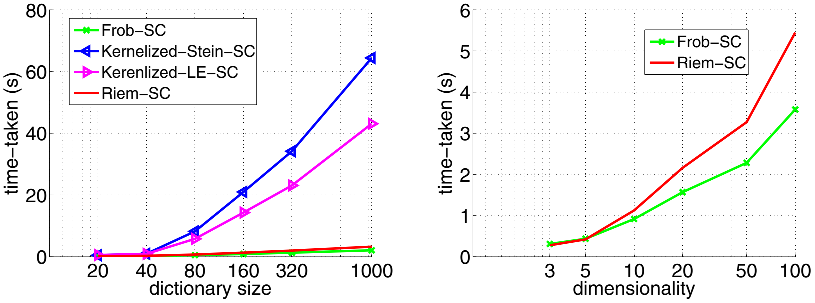
We denote our Riemannian Sparse coding setup as Riem-SC. We will compare against six other methods, namely (i) log-Euclidean sparse coding (LE-SC) [18] that projects the data into the Log-Euclidean symmetric space, followed by sparse coding the matrices as Euclidean objects, (ii) Frob-SC, in which the manifold structure is discarded, (iii) Stein-Kernel-SC [20] using a kernel defined by the symmetric Stein divergence [21], (iv) Log-Euclidean Kernel-SC which is similar to (iii) but uses the log-Euclidean kernel [23], (v) tensor sparse coding (TSC) [15] which uses the log-determinant divergence, and generalized dictionary learning (GDL) [16].

6.2 Simulated Experiments

Simulation Setup: In this subsection, we evaluate in a controlled setting, some of the properties of our scheme. For all our simulations, we used covariances generated from data vectors sampled from a zero-mean unit covariance normal distribution. For each covariance sample, the number of data vectors is chosen to be ten times its dimensionality. For fairness of the comparisons, we adjusted the regularization parameters of the sparse coding algorithms so that the codes generated are approximately 10% sparse. The plots to follow show the performance averaged over 50 trials. Further, all the algorithms in this experiment used the SPG method to solve their respective formulations so that their performances are comparable. The intention of these timing comparisons is to empirically point out the relative computational complexity of our Riemannian scheme against the baselines rather than to show exact computational times. For example, for the comparisons against the method Frob-SC, one can vectorize the matrices and then use a vectorial sparse coding scheme. In that case, Frob-SC will be substantially faster, and incomparable to our scheme as it solves a different problem.

Increasing Dictionary Size: In this experiment, we fixed the matrix dimensionality to 10, while increased the number of dictionary atoms from 20 to 1000. Figure 2(a) shows the result. As is expected, the sparse coding performance of all the kernelized schemes drops significantly for larger dictionary sizes, while our scheme performs fairly.

Increasing Matrix Dimensionality: In this experiment, we fixed the number of dictionary atoms to be 200, while increased the matrix dimensionality from 3 to 100. Figure 2(b) shows the result of this experiment. The plot shows that the extra computations required by Riem-SC is not substantial compared to Frob-SC.



(a) Dictionary Size (b) Matrix Dimensionality

Fig. 2. Sparse coding time against (a) increasing number of dictionary atoms and (b) increasing matrix dimensionality. We used a maximum of 100 iterations for all the algorithms.

6.3 Experiments with Public Datasets

Now let us evaluate the performance of our framework on real-world computer vision datasets. We experimented on four datasets, namely (i) texture recognition, (ii) person re-identification, (iii) view-invariant object recognition, and (iv) 3D object recognition. We describe these datasets below.

Brodatz Texture: Covariances have shown promising results for texture recognition [1,31] problems. Following the work of [15], we use the Brodatz texture dataset[2] for this experiment, which consists of 110 gray scale texture images. Each image is of dimension 512 × 512. We sampled approximately 300 patches, each of dimension 25 × 25, from random locations of each image, from which we removed patches without any useful textures (low entropy). This resulted in approximately 10K patches. We used a five dimensional feature descriptor to compute the covariances, with features given by: $F_{textures} = \left[x, y, I, |I_x|, |I_y|\right]^T$. The first two dimensions are the coordinates of a pixel from the top-left corner of a patch, the last three dimensions capture the image intensity, and gradients in the x and y directions respectively. Covariances of size 5 × 5 are generated from all features in a patch.

[2] `http://www.ux.uis.no/~tranden/brodatz.html`

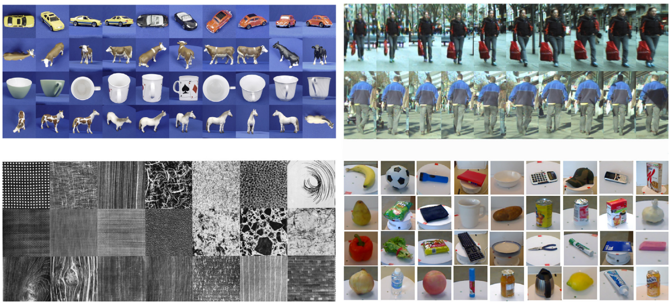


Fig. 3. Montage of sample images from the four datasets used in our experiments. Top-left are samples from the ETH80 object dataset, bottom-left are the Brodatz textures, top-right are the samples from ETHZ people dataset, and images from the RGB-D object recognition dataset are shown on bottom right.

ETH80 Object Recognition: Region covariances are studied for object recognition applications in [2] demonstrating significant performance gains. In this experiment, we loosely follow their experimental setup and evaluate our scheme on the object recognition problem using the ETH80 dataset. This dataset consists of eight ground truth object categories, each consisting of ten different instances (see Figure 3) from 41 different views, for a total of 3280 images. These objects undergo severe view point change, posing significant challenges to recognition due to the high intra-class diversity.

To generate covariances, we use a combination of texture and color features. First, we segment out the objects from the images using the given ground truth masks. Next, we generate texture features from these segmented objects using a bank of Laws texture filters [32] defined as: let $H_1 = [1\ 2\ 1]^T$, $H_2 = [-1\ 0\ 1]^T$, and $H_3 = [-1\ 2\ -1]^T$ denote the filter templates, then the filter bank is:

$$L_{bank} = \left[H_1H_1^T\ H_1H_2^T\ H_1H_3^T\ H_2H_1^T\ H_2H_2^T\ H_2H_3^T\ H_3H_1^T\ H_3H_2^T\ H_3H_3^T\right]^T. \tag{11}$$

Let F_{Laws} be a 9D feature vector obtained from every pixel after applying L_{bank}. Appending other texture and color features as provided by the pixel color, and gradients, our complete feature vector for every pixel on the object is:

$$F_{ETH80} = \left[F_{Laws}\ x\ y\ I_r\ I_g\ I_b\ |I_x|\ |I_y|\ |I_{LoG}|\ \sqrt{I_x^2 + I_y^2}\right]^T, \tag{12}$$

where I_{LoG} stands for the Laplacian of Gaussian filter, useful for edge detection. With this feature set, we generate covariances of size 19 × 19 for each image.

ETHZ Person Re-identification Dataset: Recognition and tracking of people are essential components of a visual surveillance system. Typically, the visual data from such systems are more challenging compared to other sub-areas of computer vision. These challenges arise mainly because the images used are generally shot in low-resolution or low-lighting conditions and the appearances of the same person differ significantly from one camera to the next due to changes in poses, occlusions, etc. Covariances have shown to be robust to these challenges, making it an attractive option for person re-identification tasks [20,5].

In this experiment, we evaluate people appearances recognition on the benchmark ETHZ dataset [33]. This dataset consists of low-resolution surveillance images with sizes varying between 78 × 30 to 400 × 200 pixels. The images are from 146 different individuals and the number of images for a single person varies between 5 and 356. Sample images from this dataset are shown in Figure 3. There are a total of 8580 images in this dataset.

In literature, there exist several proposals for features on this task; examples include Gabor wavelet [5], color gradients [20], etc. Rather than demonstrating the performances of various feature combinations, we detail below the combination that worked best in our experiments (on a small validation set). Our feature vector for this task is obtained by combining nine features:

$$F_{ETHZ} = \left[x \; I_r \; I_g \; I_b \; Y_i \; |I_x| \; |I_y| \; |\sin(\theta) + \cos(\theta)| \; |H_y|\right]^T, \tag{13}$$

where x is the x-coordinate of a pixel location, I_r, I_g, I_b are the RGB color of a pixel, Y_i is the pixel intensity in the YCbCr color space, I_x, I_y are the gray scale pixel gradients, and H_y is the y-gradient of pixel hue. Further, we also use the gradient angle $\theta = \tan^{-1}(I_y/I_x)$. We resized each image to a fixed size of 300 × 100, dividing it into upper and lower parts. We compute a different covariance matrix for each part, which are then merged as two block diagonal matrices to form a single 18 × 18 covariance for each image.

3D Object Recognition Dataset: The goal of this experiment is to recognize objects in 3D point clouds. To this end, we used the public RGB-D Object dataset [34], which consists of about 300 objects belonging to 51 categories and spread in about 250K frames. We used approximately 15K frames for our evaluation with approximately 250-350 frames devoted to every object seen from three different view points (30, 45, and 60 degrees above the horizon). Following the procedure suggested in [35][Chap. 5], for every frame, the object was segmented out and 18 dimensional feature vectors generated for every 3D point in the cloud (and thus 18 × 18 covariance descriptors); the features we used are as follows:

$$F_{RGBD} = \left[x, y, z, I_r, I_g, I_b, I_x, I_y, I_{xx}, I_{yy}, I_{xy}, I_m, \delta_x, \delta_y, \delta_m, \nu_x, \nu_y, \nu_z\right], \tag{14}$$

where the first three dimensions are the spatial coordinates, I_m is the magnitude of the intensity gradient, δ's represent gradients over the depth-maps, and ν represents the surface normal at the given 3D point. Sample images from this dataset are given in Figure 3.

Experimental Setup: We used 80% of the Brodatz texture and the ETH80 objects datasets to form the training set and the remaining as the test set. Further, 20% of the training set was used as a validation set. For both these datasets, we used a linear SVM for training and classification. For the ETHZ dataset and the RGB-D objects dataset, since there might not be enough images from a single class to train a classifier, we resort to a nearest neighbor classification scheme using our sparse coding framework. For this setup, we used 20% for learning the dictionary, while the remaining data points were used as the query database. The splitting was such that there is at least one data matrix from each class in the training and the test set respectively. We used a dictionary of fixed size, which is 10 times the matrix dimensionality, for all the experiments. This dictionary was learned from the training set using Log-Euclidean K-Means followed by projecting the cluster centroids onto the SPD manifold (exponential map). The regularizations in the sparse coding objective was adjusted for all the datasets (and all the experiments) to generate 10% sparse vectors. The slack parameter for the SVM was selected via cross-validation.

Results: In Tables 1, 2, 3, 4, we report results of our Riem-SC scheme against several state-of-the-art methods. For the texture and the object classification problems, we report the average SVM classification accuracy after 5-fold cross-validation. For the ETHZ people re-identification and RGB-D object recognition, our experiments were as follows: every data point in the test set was selected as a query point, and its nearest neighbor, in the Euclidean sense, is found (recall that the database points and this query point are sparse coded), and is deemed correct if their ground truth labels matched. The table shows that Riem-SC shows consistent and state-of-the-art performance against other schemes. Other methods, especially TSC and GDL are seen to perform poorly, while the kernelized schemes perform favorably. Along with the accuracies, we also report the respective standard deviations over the trials.

Discussion: Overall, our real-world and simulated experiments reveal that our sparse coding scheme demonstrates excellent application performance, while remaining computationally tractable. The kernelized schemes show a close match in accuracy to our scheme which is not unsurprising as they project the data points onto a linear feature space. However, these methods suffer when working with larger dictionary sizes, as our results in Figure 2 show. Other sparse coding schemes such as TSC are difficult to optimize while their performances are poor. In summary, our algorithm offers a practical trade off between accuracy and performance.

7 Conclusion and Future Work

In this paper, we proposed a novel scheme for representing symmetric positive definite matrices as sparse linear combinations of atoms from a given dictionary; these atoms themselves being SPD matrices. In contrast to other approaches that use proxy distances on the manifold to define the sparse reconstruction loss, we propose to use the most natural Riemannian metric on the manifold, namely the

Table 1. Brodatz texture dataset

Method	Accuracy (%)
LE-SC	47.4 (11.1)
Frob-SC	32.3 (4.4)
K-Stein-SC	39.2 (0.79)
K-LE-SC	47.9 (0.46)
TSC	35.6 (7.1)
GDL	43.7 (6.3)
Riem-SC(ours)	**53.9** (3.4)

Table 2. ETH80 object recognition

Method	Accuracy (%)
LE-SC	68.9 (3.3)
Frob-SC	67.3 (1.4)
K-Stein-SC	**81.6** (2.1)
K-LE-SC	76.6 (0.4)
TSC	37.1 (3.9)
GDL	65.8 (3.1)
Riem-SC(ours)	77.9 (1.9)

Table 3. ETHZ Person Re-identification

Method	Accuracy (%)
LE-SC	78.5 (2.5)
Frob-SC	83.7 (0.2)
K-Stein-SC	88.3 (0.4)
K-LE-SC	87.8 (0.8)
TSC	67.7 (1.2)
GDL	30.5 (1.7)
Riem-SC(ours)	**90.1** (0.9)

Table 4. RGB-D Object Recognition

Method	Accuracy (%)
LE-SC	**86.1** (1.0)
Frob-SC	80.3 (1.1)
K-Stein-SC	75.6 (1.1)
K-LE-SC	83.5 (0.2)
TSC	72.8 (2.1)
GDL	61.9 (0.4)
Riem-SC(ours)	84.0 (0.6)

Affine Invariant Riemannian distance. Further, we proposed a simple scheme to optimize the resultant sparse coding objective. Our experiments demonstrate that our scheme is computationally efficient and produces superior results compared to other schemes on several computer vision datasets. Going forward, an important future direction is the problem of efficient dictionary learning under these formulations.

Acknowledgements. AC would like to thank Dr. Duc Fehr for sharing the RGB-D objects covariance data.

References

1. Tuzel, O., Porikli, F., Meer, P.: Region Covariance: A Fast Descriptor for Detection and Classification. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3952, pp. 589–600. Springer, Heidelberg (2006)
2. Sandeep, J., Richard, H., Mathieu, S., Li, H., Harandi, M.: Kernel methods on the Riemannian manifold of symmetric positive definite matrices. In: CVPR (2013)
3. Pang, Y., Yuan, Y., Li, X.: Gabor-based region covariance matrices for face recognition. IEEE Transactions on Circuits and Systems for Video Technology 18(7), 989–993 (2008)
4. Porikli, F., Tuzel, O.: Covariance tracker. In: CVPR (June 2006)
5. Ma, B., Su, Y., Jurie, F., et al.: Bicov: A novel image representation for person re-identification and face verification. In: BMVC (2012)

6. Cherian, A., Morellas, V., Papanikolopoulos, N., Bedros, S.J.: Dirichlet process mixture models on symmetric positive definite matrices for appearance clustering in video surveillance applications. In: IEEE CVPR, pp. 3417–3424 (2011)
7. Fehr, D., Cherian, A., Sivalingam, R., Nickolay, S., Morellas, V., Papanikolopoulos, N.: Compact covariance descriptors in 3d point clouds for object recognition. In: IEEE ICRA (2012)
8. Ma, B., Wu, Y., Sun, F.: Affine object tracking using kernel-based region covariance descriptors. In: Wang, Y., Li, T. (eds.) ISKE 2011. AISC, vol. 122, pp. 613–623. Springer, Heidelberg (2011)
9. Elad, M., Aharon, M.: Image denoising via learned dictionaries and sparse representation. In: CVPR (2006)
10. Olshausen, B., Field, D.: Sparse coding with an overcomplete basis set: A strategy employed by V1. Vision Research 37(23), 3311–3325 (1997)
11. Guha, T., Ward, R.K.: Learning sparse representations for human action recognition. PAMI 34(8), 1576–1588 (2012)
12. Wright, J., Yang, A.Y., Ganesh, A., Sastry, S.S., Ma, Y.: Robust face recognition via sparse representation. PAMI 31(2), 210–227 (2009)
13. Yang, J., Yu, K., Gong, Y., Huang, T.: Linear spatial pyramid matching using sparse coding for image classification. In: IEEE CVPR (2009)
14. Pennec, X., Fillard, P., Ayache, N.: A Riemannian framework for tensor computing. IJCV 66(1), 41–66 (2006)
15. Sivalingam, R., Boley, D., Morellas, V., Papanikolopoulos, N.: Tensor sparse coding for region covariances. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part IV. LNCS, vol. 6314, pp. 722–735. Springer, Heidelberg (2010)
16. Sra, S., Cherian, A.: Generalized dictionary learning for symmetric positive definite matrices with application to nearest neighbor retrieval. In: Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.) ECML PKDD 2011, Part III. LNCS (LNAI), vol. 6913, pp. 318–332. Springer, Heidelberg (2011)
17. Arsigny, V., Fillard, P., Pennec, X., Ayache, N.: Log-Euclidean metrics for fast and simple calculus on diffusion tensors. Magnetic Resonance in Medicine 56(2), 411–421 (2006)
18. Guo, K., Ishwar, P., Konrad, J.: Action recognition using sparse representation on covariance manifolds of optical flow. In: IEEE AVSS (2010)
19. Ho, J., Xie, Y., Vemuri, B.: On a nonlinear generalization of sparse coding and dictionary learning. In: ICML (2013)
20. Harandi, M.T., Sanderson, C., Hartley, R., Lovell, B.C.: Sparse coding and dictionary learning for symmetric positive definite matrices: A kernel approach. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part II. LNCS, vol. 7573, pp. 216–229. Springer, Heidelberg (2012)
21. Sra, S.: Positive definite matrices and the S-divergence. ArXiv preprint ArXiv:1110.1773 (2011)
22. Jayasumana, S., Hartley, R., Salzmann, M., Li, H., Harandi, M.: Kernel methods on the riemannian manifold of symmetric positive definite matrices. In: CVPR (2013)
23. Li, P., Wang, Q., Zuo, W., Zhang, L.: Log-euclidean kernels for sparse representation and dictionary learning. In: IEEE ICCV (2013)
24. Bhatia, R.: Positive Definite Matrices. Princeton University Press (2007)
25. Hiriart-Urruty, J.B., Lemaréchal, C.: Fundamentals of convex analysis. Springer (2001)
26. Higham, N.: Functions of Matrices: Theory and Computation. SIAM (2008)

27. Bertsekas, D.P.: Nonlinear Programming, 2nd edn. Athena Scientific (1999)
28. Barzilai, J., Borwein, J.M.: Two-Point Step Size Gradient Methods. IMA J. Num. Analy. 8(1) (1988)
29. Schmidt, M., van den Berg, E., Friedlander, M., Murphy, K.: Optimizing Costly Functions with Simple Constraints: A Limited-Memory Projected Quasi-Newton Algorithm. In: AISTATS (2009)
30. Birgin, E.G., Martínez, J.M., Raydan, M.: Algorithm 813: SPG - Software for Convex-constrained Optimization. ACM Transactions on Mathematical Software 27, 340–349 (2001)
31. Luis-García, R., Deriche, R., Alberola-López, C.: Texture and color segmentation based on the combined use of the structure tensor and the image components. Signal Processing 88(4), 776–795 (2008)
32. Laws, K.I.: Rapid texture identification. In: 24th Annual Technical Symposium, International Society for Optics and Photonics, pp. 376–381 (1980)
33. Schwartz, W., Davis, L.: Learning Discriminative Appearance-Based Models Using Partial Least Squares. In: Proceedings of the XXII Brazilian Symposium on Computer Graphics and Image Processing (2009)
34. Lai, K., Bo, L., Ren, X., Fox, D.: A large-scale hierarchical multi-view rgb-d object dataset. In: ICRA (2011)
35. Fehr, D.A.: Covariance based point cloud descriptors for object detection and classification. University of Minnesota (2013)