Generalized Dictionary Learning for Symmetric Positive Definite Matrices with Application to Nearest Neighbor Retrieval

Suvrit Sra[1] and Anoop Cherian[2]

[1] MPI for Intelligent Systems,
72076 Tübingen, Germany
[2] University of Minnesota, Twin Cities,
Minneapolis, MN-55414, USA

Abstract. We introduce *Generalized Dictionary Learning* (GDL), a simple but practical framework for learning dictionaries over the manifold of positive definite matrices. We illustrate GDL by applying it to Nearest Neighbor (NN) retrieval, a task of fundamental importance in disciplines such as machine learning and computer vision. GDL distinguishes itself from traditional dictionary learning approaches by explicitly taking into account the manifold structure of the data. In particular, GDL allows performing "sparse coding" of positive definite matrices, which enables better NN retrieval. Experiments on several covariance matrix datasets show that GDL achieves performance rivaling state-of-the-art techniques.

1 Introduction

Recent times have seen a steep rise of data that are encoded as matrices or tensors. Such data goes beyond traditional vector based models, and offers new means of capturing intrinsic structure. The additional structure can in turn bring several benefits, such as richer representations, robustness to noise, and perhaps even improved empirical performance.

Some successful applications that depend on matrix valued data include: multi-camera tracking based on covariance matrices derived from appearance silhouettes [11,30]; medical diagnostics via diffusion tensor imaging [1,43]; computational anatomy [23]; robust face recognition [31]; and action recognition [41], among many others.

Like the works cited above, we too focus on matrix valued data, in particular on the highly important class of symmetric positive (semi)definite (SPD) matrices (e.g., covariance, correlation, kernel matrices). We deal with SPD matrices in the context of overcomplete dictionary learning, for which we present a simple but effective, new framework called *Generalized Dictionary Learning* (GDL).

Our framework extends the idea of dictionary learning over vectors [29,10] to dictionary learning over matrices. Consequently, GDL provides an approach to perform *sparse coding* for input covariance matrices. To illustrate the benefits of such sparse coding, we show an application to Nearest Neighbor (NN)-based

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 318–332, 2011.

object retrieval—a problem of central importance in machine learning and computer vision [42,31,38]. Experiments (see Section 4) reveal that GDL leads to NN performance rivaling state-of-the-art approaches.

To help place this paper in perspective, we list below its key contributions.

- **Framework.** We extend dictionary learning to matrices, and specialize this for the broadly useful class of SPD matrices.
- **Algorithm.** For GDL we present a scalable online algorithm that also allows rapid sparse coding.
- **Application.** We apply GDL to accelerate NN-based object retrieval; GDL leads to better accuracy than many of the competing methods.

1.1 Background and Related Work

We summarize below some relevant background material, which includes a brief sketch of literature directly relevant to our paper. We begin with the key objects of this paper: *covariance matrices*[1].

Typically covariance matrices encode input data as follows. For each input object, first, a set of vector-valued features is extracted. Then, the covariance matrix of these features is computed to obtain a *structured representation* for the object in question (also see Figure 1). Note that we are *not* talking about a covariance matrix across objects—rather, there is a separate covariance matrix for *each* input object.

What makes covariance matrices so special? Apart from encoding inter-feature dependencies, a key aspect of covariance matrices that has been found to be widely useful is their natural geometric property: they inhabit a Riemmanian manifold of negative curvature [22]. This geometric property is central to several algorithms that deal with covariance data while accounting for their manifold structure [2,42,31,38]. But efficiently handling this structure is nontrivial. The difficulty stems from two main reasons: (i) defining divergence, distance, or kernel functions on covariances is not easy; and (ii) even for basic computations such as distances, clustering, etc., the numerical burden is substantial.

These burdens are especially pronounced in the context of our application: nearest neighbor retrieval, where rapid and accurate processing is crucial. Let us, therefore, briefly review the state-of-the-art techniques for NN in general, while considering how they extend to SPD matrices in particular.

Nearest Neighbors. Efficient NN retrieval on covariance data is an area still in its infancy, so literature on it is scarce. Perhaps the simplest approach to NN retrieval on covariances is to use their natural Riemannian metric, namely, the geodesic distance [12]. This is defined for two covariance matrices X and Y, as

$$d_{geo} = \|\log(\lambda_{X,Y})\| \tag{1}$$

where $\lambda_{X,Y}$ is a vector of the generalized eigenvalues of X and Y. This metric, together with the Karcher mean algorithm [30] for computing the centroids of SPD

[1] We use the terms "symmetric positive definite" and "covariance" interchangeably.

matrices opens up the possibility of using a metric tree [8] data structure. While this seems an attractive option, it is seldom used in practice as the Karcher mean is an iterative and computationally intensive algorithm, making it undesirable in data intensive applications such as NN. Thus, the main line of investigation for NN has been towards developing fast hashing schemes. A possibility for hashing is in discarding the manifold structure by vectorizing the covariances through their tangent space, albeit using the log-Euclidean projection, so that vector space techniques such as locality sensitive hashing (LSH) [14,18,21] become applicable [37]. The log-Euclidean embedding was also used by [7] to develop a spectral hashing method.

On the other hand, NN for vector valued data has been a core research topic for many decades; several data structure based algorithms can be found in classic references such as [19,28]. But it is well-recognized that exact NN is computationally prohibitive for high-dimensional data, whereby in recent years increasingly approaches based on approximate methods, such as approximate nearest neighbors [3,16,15] and LSH [14,18,21] have emerged. These techniques exploit the geometric properties of the data to assign "similar" data points to the same bucket by using carefully chosen hash functions. More recently, machine learning techniques have been suggested to learn the hash functions directly from the data itself [39,42,20]. We also take a machine learning based approach, but since our underlying data space is actually a manifold, not a vector space, existing vector-based methods do not directly apply.

Learning with SPD matrices. We propose to encode SPD matrices using a weighted *sparse* combination of rank-1 positive semidefinite matrices. This idea is natural, and has been previously explored in other contexts too. For example, in [34,35] the authors investigate this idea for Mahalanobis metric learning. A paper closer in spirit to our approach is [36], where a given covariance matrix is assumed to be representable as a sparse linear combination of SPD atoms in a tensor dictionary; the learning problem, however, is formulated as a Semidefinite Program (SDP) which makes it improbable to scale to large datasets. Finally, we note that in the compressed sensing community, optimizing over matrix-valued data by minimizing the trace norm (nuclear norm) [5,26,6] is popular. But the compressed sensing setup is orthogonal to ours: there one assumes that one has a dictionary, while in our case, we seek to learn a dictionary.

2 Generalized Dictionary Learning

Traditional dictionary learning processes input vectors (signals) $s_i \in \mathbb{R}^p$, $1 \leq i \leq m$, to construct a matrix $D \in \mathbb{R}^{p \times n}$ and vectors $c_i \in \mathbb{R}^n$ (n is the number of "basis" vectors; usually, $n \gg p$), so that

$$s_i \approx Dc_i, \quad \text{and} \quad c_i \text{ is sparse}, \quad \text{for } 1 \leq i \leq m.$$

The sparsity requirement on c_i is commonly enforced using ℓ_0- or ℓ_1-norm penalties (or constraints). Since, both D and c_i are unknown, the dictionary learning

problem leads to a difficult nonconvex optimization task. Nevertheless, numerically it has been a successful approach toward sparse coding [40,10].

Now, we depart from the traditional setup above: we assume that instead of vectors, we have input matrices $S_i \in \mathbb{R}^{p\times q}$, $1 \le i \le m$. Thus, instead of a matrix D, we learn a tensor $\mathcal{D}$, which we identify with a linear operator $\mathcal{D} : \mathbb{R}^{n\times r} \to \mathbb{R}^{p\times q}$. This operator maps a matrix C_i to obtain an approximate S_i; formally,

$$S_i \approx \mathcal{D}C_i, \quad \text{and} \quad C_i \text{ is sparse}, \quad \text{for } 1 \le i \le m. \tag{2}$$

Based on (2), we propose to solve *Generalized Dictionary Learning* (GDL) by casting it as the penalized optimization problem

$$\min_{C_1,\ldots,C_m,D} \quad \tfrac{1}{2}\sum\nolimits_{i=1}^{m} \|S - \mathcal{D}C_i\|_{\mathrm{F}}^2 + \sum\nolimits_{i=1}^{m} \beta_i \mathrm{sp}(C_i), \tag{3}$$

where $\beta_i > 0$ are scalars, and the function $\mathrm{sp}(C)$ enforces some notion of sparsity. Typical choices for $\mathrm{sp}(C)$ include, the cardinality function $\|C\|_0$, its convex relaxation $\|C\|_1$, the matrix rank function $\mathrm{rank}(C)$, or its convex relaxation, the trace-norm $\|C\|_{\mathrm{tr}}$.

How does (3) apply to SPD matrices? To answer this, let us restrict the input matrices S_i to $\mathscr{S}_{++}^p$, the set of $p \times p$, SPD matrices. To ensure that the approximation $\mathcal{D}C_i$ is also SPD, we must impose some structural restrictions on both the dictionary $\mathcal{D}$ and the coefficient matrix C_i. The following easily proved, classical result from matrix algebra provides the key.

Theorem 1. *If $A \succeq 0$, and B is any matrix (of suitable size), then $BAB^T \succeq 0$.*

Theorem (1) suggests that we should encode $\mathcal{D}$ by the following bilinear map

$$\mathcal{D}C := DCD^T, \quad \text{for some matrix } D, \tag{4}$$

and additionally *restrict* to $C \succeq 0$. Notice that, viewed as a linear map (4) can be written using the 'vec' operator that stacks columns of its argument as follows:

$$\mathrm{vec}(DCD^T) = (D \otimes D)\,\mathrm{vec}(C), \tag{5}$$

where the operator $\mathcal{D}$ is identified with the product $D \otimes D$. The matrix notation in (4) looks simpler. If we were to use a general matrix in (5), the storage and computational would be much higher. It can be easily seen that (5) takes $md^2 + d^2n + mn$ storage space for m covariance matrices of dimension d each, while (4) takes $md^2 + dn + mn$. Computationally, the second formulation leads to $O(d^2n)$ per iteration cost, while the first one leads to just $O(dn)$. Thus, we prefer formulation (4).

As for the coefficient matrix C, there are two fundamental choices:

1. $C = \mathrm{Diag}(c_1, \ldots, c_n)$ where $c_i \ge 0$; and
2. $C = \sum_j^k c_j c_j^T$, a general, potentially low-rank (if $k < n$) SPD matrix.

We focus on the first choice, and it is equivalent to modeling input SPD matrices as weighted sums of rank-1 matrices; specifically,

$$S \approx DCD^T = \sum\nolimits_{i=1}^{n} c_i d_i d_i^T, \quad \text{where } c_i = C_{ii}. \tag{6}$$

Although choosing a diagonal C might appear to be simple, it is quite powerful as equation (6) suggests; more importantly, this choice prevents a parameter explosion and proves crucial for GDL's computational efficiency.

2.1 Online GDL Algorithm

Now we proceed to deriving an efficient online (stochastic-gradient based) algorithm for approximately solving the GDL problem. To keep the subproblems tractable, we use the convex function $\mathrm{sp}(C) = \|C\|_1$ for enforcing sparsity. Then, using representation (6), the GDL formulation (3) becomes

$$\min_{C_1,\ldots,C_N \geq 0, D} \quad \tfrac{1}{2}\sum\nolimits_{i=1}^{m} \|S_i - DC_iD^T\|_{\mathrm{F}}^2 + \sum\nolimits_{i=1}^{m} \beta_i \|C_i\|_1, \tag{7}$$

where $\beta_j > 0$ are sparsity-tuning scalars, and $C_1, \ldots, C_m \geq 0$ are diagonal.

Like its vector space counterpart, the GDL problem (7) is also nonconvex, which makes it extremely unlikely to obtain a globally optimal solution. Fortunately, it is individually convex in D and $(C_1, \ldots, C_m)$, which suggests that a minimization strategy that alternates between optimizing over D and the matrices $(C_1, \ldots, C_m)$, could be applied. However, often the number of input data points m is very large, whereby, the alternating step over the C_i can easily become computationally prohibitive. Taking cue from the recent work in dictionary learning [10,27], we develop an online algorithm based on stochastic gradient descent. The online approach allows our GDL algorithm to easily scale to large datasets, as long as the subproblems can be solved efficiently. We now describe the key algorithmic details.

To prevent degenerate solutions, it is often useful to impose some normalization constraints on D. A practical choice is to require $\|d_j\| \leq 1$ for each column of matrix D. We denote these requirements by the feasible set $\mathscr{D}$. To run a stochastic-gradient procedure, we break up the processing into B "mini-batches." Then, we rewrite the GDL (7) over these mini-batches as

$$\min_{D \in \mathscr{D}} \quad \Phi(D) := \sum\nolimits_{b=1}^{B} \phi_b(D), \tag{8}$$

where ϕ_b denotes the objective function for batch b. Let k_b be the size of batch b $(1 \leq b \leq B)$ that contains the input matrices $\{S_{j(i)} | 1 \leq i \leq k_b\}$, where $j(i)$ denotes an appropriate index in $1, \ldots, m$. With this notation, the objective function for batch b may be written as

$$\phi_b(D) := \min_{C_{j(1)},\ldots,C_{j(k_b)} \geq 0} \quad \tfrac{1}{2}\sum\nolimits_{i=1}^{k_b} \|S_{j(i)} - DC_{j(i)}D^T\|_{\mathrm{F}}^2 + \beta_{j(i)}\|C_{j(i)}\|_1. \tag{9}$$

Our algorithm then iteratively updates the dictionary by computing

$$D_{t+1} = \Pi_{\mathscr{D}}(D_t - \eta_t \nabla_D \phi_{b(t)}(D_t)), \quad b(t) \in [1..B],\ t = 0, 1, \ldots, \tag{10}$$

where $\Pi_{\mathscr{D}}$ denotes orthogonal projection onto $\mathscr{D}$. Standard analysis (see e.g., [13]) shows that under appropriate conditions of the stepsizes η_t, the iteration above

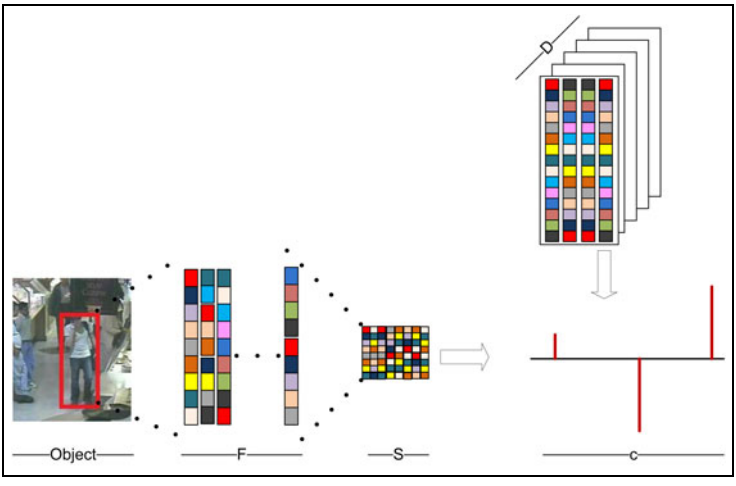

Fig. 1. Sparse coding for covariances: From the object one extracts base-features $F_1, \ldots, F_k$. These, then yield the covariance feature $S = \sum_i (F_i - \mu)(F_i - \mu)^T$ where μ is the mean feature vector, which has a sparse coding C in the dictionary $\mathcal{D}$, i.e., $S \approx \mathcal{D}C$.

converges to a stationary point of the problem. For implementing (10), note that if ϕ_b is determined by a unique solution to (9), then it can be shown that the gradient $\nabla_D \phi_{b(t)}$ is well defined. Specifically, let $(C^*_{j(1)}, \ldots, C^*_{j(k_b)})$ be the argmin of (9). Some calculus then shows that (let $b \equiv b(t)$)

$$\nabla_D \phi_b(D) = 2 \sum\nolimits_{i=1}^{k_b} \Big(DC^*_{j(i)} D^T - S_{j(i)} \Big) DC^*_{j(i)}. \tag{11}$$

2.2 Sparse Coding: Computing ϕ_b

All that remains to specify is how to compute ϕ_b, i.e., how to solve (9). First, notice that (9) is just a sum of k_b independent problems, so without loss of generality we need to consider only a subproblem of the form

$$\min_{C \geq 0} \quad f(C) := \tfrac{1}{2} \|S - DCD^T\|_{\mathrm{F}}^2 + \beta \|C\|_1, \tag{12}$$

where $\beta > 0$, and C is restricted to be a diagonal matrix. Since $C \geq 0$, problem (12) further simplifies to

$$\min_{C \geq 0} \quad f(C) := \tfrac{1}{2} \|S - DCD^T\|_{\mathrm{F}}^2 + \beta \operatorname{Tr}(C), \tag{13}$$

which is nothing but a regularized nonnegative least-squares (NNLS) problem. There exist a variety of solvers for NNLS, for example, LBFGS-B [25], SPG [4], or the very recent SBB [17]. We prefer SBB, as it is not only simple, but also exhibits strong empirical performance. Implementing SBB is simple, because it

Algorithm 1. Online Algorithm for GDL

Require: Input covariances $S_1, S_2 \ldots,$; stepsizes η_t
  Initialize $t \leftarrow 0$.
  while $\neg$ converged **do**
    Obtain next mini-batch of size k_b.
    for $i = 1$ to k_b **do**
      Solve for $C_{j(i)}$ using (13).
    end for
    Update dictionary, $D_{t+1} = \Pi_{\mathscr{D}}(D_t - \eta_t \nabla_D \phi_{b(t)}(D_t))$
    $t \leftarrow t + 1$.
  end while
  return D.

only requires efficient computation of $\nabla f(C)$. Since C is diagonal, the gradient $\nabla f(C)$ is given by the diagonal matrix

$$\nabla f(C) = \mathrm{Diag}\big(D^T(DCD^T - S)D\big). \tag{14}$$

Algorithm 1 assembles all the above details into pseudocode for online GDL. Figure 1 provides a high level schema of the GDL algorithm.

3 Nearest Neighbors via GDL

Once we have a dictionary D that can sparse code an input covariance matrix S, the next step is to obtain a representation of S in terms of its sparsity. Since, we use an overcomplete dictionary of a much higher dimension than the input data, and we assume that only a few elements of the dictionary participate in the reconstruction of S, there is high probability that dissimilar input data will get unique non-zero basis combinations. In other words, suppose that we use a dictionary with n rank-1 basis matrices, and that only r of these matrices are used to reconstruct a given input covariance matrix. Then, there are $\binom{n}{r}$ unique active basis combinations possible. If n and r are chosen appropriately (by adjusting the sparsity-controlling parameters β_i), then there is a high chance that each matrix gets assigned a unique set of rank-1 matrices that encode it.

Using this observation, we hash input covariances by computing a sorted tuple representation composed of the indices (in the dictionary) of the rank-1 basis matrices involved in the reconstruction. Since each dictionary basis spans a subspace (not necessarily uniquely), the tuple may be viewed as a subspace combination corresponding to the input data. Thus, we call this representation a *Subspace Combination Tuple* (SCT), and define it formally as follows:

Definition 1. *Let $S \in \mathscr{S}^p_{++}$ be an input SPD matrix, $D \in \mathbb{R}^{p \times n}$ an overcomplete dictionary, and $\boldsymbol{u}_i$, $i \in \{1, \cdots n\}$ a unique identifier for the ith column of D. If $c = (c_1, c_2, \cdots, c_n)$ is the coefficient vector corresponding to the sparse representation of S as per* (13)*, then a tuple $h(S) = \langle \boldsymbol{u}_i, \cdots, \boldsymbol{u}_k \rangle$ is defined as a* Subspace Combination Tuple *(SCT), if $\forall j \in \{i, \cdots, k\}$, $|c_j| > \epsilon$, for some $\epsilon > 0$, and $\{i, \cdots, k\}$ is a strictly increasing sequence of dictionary indices.*

In our case, we assume that the u_i's are just integers from $1, \cdots, n$ and that the hashing tuple is a set of these indices in sorted order. The threshold ϵ helps to select significant coefficients from the sparse coding, so that the chosen non-zero indices are robust to noise. This encoding of input SPD matrices as tuples opens up the possibility of using hash tables for fast locality sensitive hashing. Figure 2 illustrates this idea. Each column of the dictionary is identified by its index; so each hash-key is a set of integers encoded as a character string. To tackle collisions in the hash buckets, the colliding input matrices are organized in a linked list. If the linked list gets too long, the data within a hash bucket can be further organized using a metric tree or any other efficient data structure.

Given a query (SPD matrix), we solve (13) to first obtain its coefficient matrix, from which we obtain the SCT and query the hash table. If there are several entries in a matching bucket, we run a linear scan using the geodesic distance (1) to find the best matches (the bucket can also be organized for faster than linear scans, if desired).

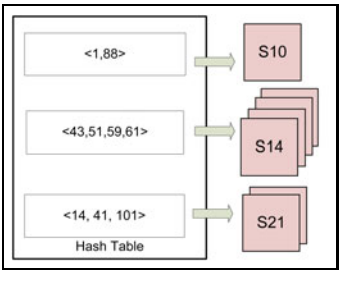

Fig. 2. An illustration of the hashing idea. The input covariance matrix is sparse coded and the nonzero active coefficients are formed into a tuple, which is then used for indexing into a hash table. To resolve hash table collisions, the colliding covariances are arranged in a suitable data structure. The covariance matrices are denoted as Si (for random values of i) in the figure.

4 Experiments and Results

We now present experimental results of applying the GDL framework to NN retrieval. Specifically, we perform extensive experiments of GDL, and compare it against the state-of-the-art NN techniques applied to NN retrieval for covariance matrices.

Since there are no publicly available datasets of covariance matrices, we had to resort to deriving them from other datasets. To this end, we selected the following types of data: (i) real-world noisy data; (ii) covariance datasets of relatively large dimensions; and (iii) a dataset with a large number of points. Our key aim in selecting such data were to highlight the applicability and relevance of our method.

For (i) we used the LabelMe object annotation dataset, for (ii) the FERET face recognition dataset, and for (iii) texture datasets. Details of each of these datasets follow.

Object dataset. We obtained the LabelMe dataset[2] of annotated images. We used approximately 10K images from this dataset, where each image contains one or more annotated objects, along with bounding box information about the location of each object in the image. We extracted these annotated blobs from each image, and created covariance matrices for these blobs. The feature vector F corresponding to each pixel in the blob of interest had the form: $F = [I_R, I_G, I_B, I_x, I_y, I_{xx}, I_{yy}]$, where the first three dimensions encode the RGB color intensities, and the last four capture the first- and second-order gradients, respectively. Thus, our covariances were 7×7, and we created a dataset containing 25K covariances from this dataset.

Face recognition. Here, we downloaded the FERET face dataset [33,32]. This dataset contains facial appearances segregated into multiple classes. Each class has different views of the face of the same person for varying poses. We selected six images from each class. Inspired by the success of covariances created from Gabor filters for face recognition [24], we applied 40 Gabor filters on each image, later combining the filters into a covariance of size 40×40. We created a covariance dataset of approximately 10K descriptors using this approach.

Texture classification. Texture is an essential cue in many data mining applications like satellite imagery, industry inspection systems, etc. Thus, we used a combination of the Brodatz dataset and the Curret dataset [9] for creating a texture covariance dataset. Brodatz contained approximately 111 texture classes, while Curret contained approimately 60 classes. To create the covariance, we used the feature vector $F = [x, y, I, I_x, I_y]$, where the first two dimensions are the relative location of the pixel with respect to the texture patch, the third dimension encodes the grayscale intensity, and the last two dimensions capture the pixel gradients. Thus, each covariance is 5×5, and we created approximately 40K such covariances.

Sample images from each of these datasets are shown in Figure 3.

4.1 Methods Compared against GDL

Log-Euclidean Embedding. The main mechanism that the state-of-the-art techniques use for NN on covariance datasets is vectorization. An input SPD matrix is projected onto its tangent plane through a log-Euclidean mapping, which results in a symmetric matrix. Since this resultant matrix is not confined to the manifold of SPD matrices, it can easily be embedded into the Euclidean space through vectorization. Then, the vectorized version can be hashed using any of the conventional hashing techniques. In our experiments, we tried two popular hashing algorithms: (1) ℓ_2-distance based LSH ($h(x) = \lfloor \frac{x.r-b}{w} \rfloor$), where x is the data vector to be hashed (L2LSH), and r and b are parameters of the

[2] http://labelme.csail.mit.edu/

Fig. 3. Sample image from LabelMe object dataset (top), FERET face appearances (mid) and Brodatz texture database (bot)

hash function; and (2) using Hamming functions (HAM) via using the binary encoding of the embedded vector.

Simple vectorization. A mere vectorization of the covariance matrix that identifies it with in Euclidean space (without projecting it using the Log-Euclidean mapping, but rather by just stacking columns), does not lead to a good hashing (VEC). However, this provides a baseline.

Kernelized LSH. A recently proposed, sophisticated technique built on LSH, is Kernelized LSH (KLSH) [20]. A major impediment in using KLSH on the space of covariances is the lack of known, effective kernel functions. We experimented with a number of potential kernels on SPD matrices (e.g., trace-kernel, log-Euclidean kernel, etc.), but found KLSH's performance to be the best when using the Riemannian kernel (which is actually a pseudo-kernel because it fails to be positive definite) generated by the Riemannian metric [12]. This kernel K has the following form: For two SPD matrices X and Y,

$$K(X,Y) := e^{-\gamma \|\log(\lambda(X,Y))\|^2}, \tag{15}$$

where $\gamma > 0$ is a "bandwidth" parameter, and $\lambda(X,Y)$ stands for the vector of generalized eigenvalues of X and Y.

4.2 Dictionary Learning

To determine the correct dictionary size for each of the datasets, we used cross validation. Assuming the dataset $S_i \in \mathscr{S}^p_{++}, i = 1, 2, \cdots, n$, we computed dictionaries of sizes $p \times kp$ for $k = 2, 3, \cdots$. For each of the dictionary, we compute the hashing accuracy on a subset of the dataset, and used the dictionary for which the best accuracy was best. This resulted in a dictionary of size 7×28 for the object dataset, 40×160 for the faces dataset, and 5×50 for the texture dataset. GDL took approximately 20 minutes for the texture dataset with 30K, 5×5 matrices, and approximately 2 hours for faces with 10K, 40×40 matrices. We ran 300 iterations of sparse coding and dictionary learning. Each of the sparse coding subproblems was seen to converge in 50–80 iterations of the SPG algorithm. For regularization, we set all β values to the same value—$\beta = 0.05$ for objects, $\beta = 0.65$ for faces, and $\beta = 0.4$ for texture, respectively. These values for β were also determined via cross-validation, while ensuring that the tuple size remained between 5–10.

4.3 Experimental Setup

GDL was implemented in Matlab; for L2LSH, VEC, and HAM we used the C-implementation from the Caltech Toolbox[3]. Since the programs have different computational baselines, we cannot compare their retrieval speed. Rather, we show in Table. 1 the average portion of each of the datasets scanned by GDL to find the nearest neighbor. The geodesic distance was used to resolve hash table collisions. As is seen from the table, the coverage (amount of database scanned) is small, which is exactly as desired.

Table 1. Percentage of the database searched by GDL to find the nearest neighbor

Dataset	Objects	Faces	Texture
Avg. coverage (%)	5.11	3.54	6.26

Next, we substantiate the effectiveness of our algorithm for NN retrieval over the three datasets. For this purpose, we split each of the datasets into database and query sets (approximately 5% of the data). To compute the ground truth, we used a linear scan via the geodesic distance over the entire database for each query. Since it is hard to find exact NN, we restrict ourselves to Approximate Nearest Neighbors (ANN). Assume Q is a query point, X_{ls} is the exact NN found by a linear scan and X_{algo} is the neighbor returned by an NN algorithm. If d_{geo} defines the geodesic distance, then we classify an NN as correct if $\frac{d_{geo}(Q,X_{ls})}{d_{geo}(Q,X_{algo})} > \epsilon$. We used $\epsilon = 0.75$. Fig. 4 shows the accuracy of NN for each of the datasets, where accuracy is defined as:

$$\text{Accuracy} := \frac{\#\text{correct matches}}{\#\text{query size}}. \tag{16}$$

[3] http://www.vision.caltech.edu/malaa/software/research/image-search/

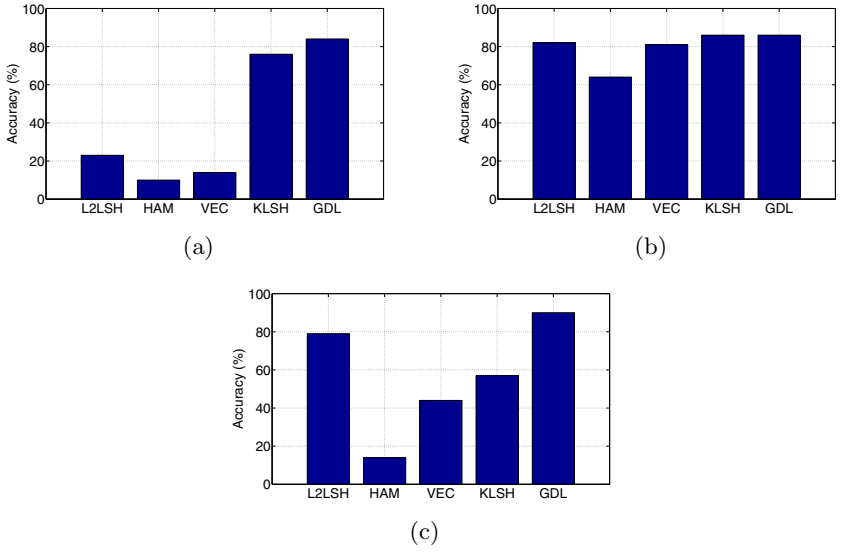

Fig. 4. Plots of the *accuracy* of NN retrieval of GDL compared to various techniques; (a) objects dataset, (b) faces dataset, and (c) texture dataset

As is evident from the plots, GDL performs well across all datasets, while the performance of the other methods vary. The vectorization approach fail on all datasets, while KLSH performed reasonably well. However, a disadvantage of KLSH compared with our method is that the former needs to compute the kernel matrix for the query point, against the *entire* dataset—this slows it down drastically. On the face dataset, all methods had high accuracy, most probably because this dataset is noise free. The other data are predominantly either outdoor images (as in LabelMe) or heavy changes in the reflectance in texture (as in the Curret dataset). For such data, adjusting the regularization parameter helps counter the effects of noise.

5 Conclusions

In this paper, we introduced a novel dictionary learning algorithm for SPD matrices which represents each input SPD matrix as a non-negative, sparse linear combination of rank-1 dictionary atoms. The key advantages of this framework are (i) the algorithm is simple and scalable, (ii) it enables fast and accurate NN on covariance datasets. We substantiated our claims by showing several experiments on real-world data. Going forward, it will be interesting to see how our framework can be extended to other matrix manifolds, beyond just SPD matrices. Another area is learning distance metric on tensors using the sparse framework, which will enable a classification or regression for SPD datasets.

References

1. Alexander, D., Pierpaoli, C., Basser, P., Gee, J.: Spatial transformations of diffusion tensor magnetic resonance images. IEEE Tran. Med. Imaging 20(11), 1131–1139 (2002)
2. 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)
3. Arya, S., Mount, D., Netanyahu, N., Silverman, R., Wu, A.: An optimal algorithm for approximate nearest neighbor searching fixed dimensions. Journal of the ACM (JACM) 45(6), 891–923 (1998)
4. Birgin, E., Martínez, J., Raydan, M.: Nonmonotone spectral projected gradient methods on convex sets. SIAM Journal on Optimization 10(4), 1196–1211 (2000)
5. Cai, J., Candes, E., Shen, Z.: A singular value thresholding algorithm for matrix completion. Arxiv preprint arXiv:0810.3286 (2008)
6. Candes, E., Plan, Y.: Matrix completion with noise. Proceedings of the IEEE 98(6), 925–936 (2010)
7. Chaudhry, R., Ivanov, Y.: Fast Approximate Nearest Neighbor Methods for Non-Euclidean Manifolds with Applications to Human Activity Analysis in Videos. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010. LNCS, vol. 6312, pp. 735–748. Springer, Heidelberg (2010)
8. Ciaccia, P., Patella, M., Zezula, P.: M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. In: Proceedings of the 23rd VLDB Conference, Athens, Greece, pp. 426–435 (1997)
9. Dana, K., Van Ginneken, B., Nayar, S., Koenderink, J.: Reflectance and texture of real-world surfaces. ACM Transactions on Graphics (TOG) 18(1), 1–34 (1999)
10. Elad, M., Aharon, M.: Image Denoising Via Sparse and Redundant Representations Over Learned Dictionaries. IEEE Tran. Image Processing 15(12), 3736–3745 (2006)
11. Porikli, F., Tuzel, O.: Covariance tracker. In: CVPR (2006)
12. Forstner, W., Moonen, B.: A metric for covariance matrices. Qua vadis geodesia, pp. 113–128 (1999)
13. Gaivoronski, A.A.: Convergence properties of backpropagation for neural nets via theory of stochastic gradient methods. Part 1. Optimization Methods and Software 4(2), 117–134 (1994)
14. Gionis, A., Indyk, P., Motwani, R.: Similarity search in high dimensions via hashing. In: Proceedings of the 25th International Conference on Very Large Data Bases, pp. 518–529 (1999)
15. Indyk, P.: On approximate nearest neighbors in non-euclidean spaces. In: Proceedings of the 39th Annual Symposium on Foundations of Computer Science, p. 148 (1998)
16. Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, pp. 604–613 (1998)
17. Kim, D., Sra, S., Dhillon, I.: A non-monotonic method for large-scale non-negative least squares. Preprint on: Optimization Online (2011)
18. Kleinberg, J.: Two algorithms for nearest-neighbor search in high dimensions. In: Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing, p. 608 (1997)
19. Knuth, D.: The art of computer programming. Sorting and Searching, vol. 3. Addison-Wesley, Reading (1973)

20. Kulis, B., Grauman, K.: Kernelized locality-sensitive hashing for scalable image search. In: ICCV (2009)
21. Kushilevitz, E., Ostrovsky, R., Rabani, Y.: Efficient search for approximate nearest neighbor in high dimensional spaces. In: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, p. 623 (1998)
22. Lang, S.: Fundamentals of differential geometry. Graduate Texts in Mathematics, vol. 191 (1999)
23. Lepore, N., Brun, C., Chou, Y., Chiang, M., Dutton, R., Hayashi, K., Luders, E., Lopez, O., Aizenstein, H., Toga, A., et al.: Generalized tensor-based morphometry of HIV/AIDS using multivariate statistics on deformation tensors. IEEE Tran. Med. Imaging 27(1), 129–141 (2007)
24. Liu, C.: Gabor-based kernel PCA with fractional power polynomial models for face recognition. IEEE PAMI 26(5), 572–581 (2004)
25. Liu, D., Nocedal, J.: On the limited memory BFGS method for large scale optimization. Mathematical Programming 45(1), 503–528 (1989)
26. Liu, Z., Vandenberghe, L.: Interior-point method for nuclear norm approximation with application to system identification. SIAM Journal on Matrix Analysis and Applications 31(3), 1235–1256 (2009)
27. Mairal, J., Bach, F., Ponce, J., Sapiro, G.: Online dictionary learning for sparse coding. In: Proceedings of the 26th Annual International Conference on Machine Learning, pp. 689–696. ACM, New York (2009)
28. Mehlhorn, K.: Data structures and algorithms 3: multi-dimensional searching and computational geometry. Springer-Verlag New York, Inc., New York (1984)
29. Murray, J., Kreutz-Delgado, K.: Sparse image coding using learned overcomplete dictionaries. Machine Learning for Signal Processing, 579–588 (September 2004)
30. Tuzel, O., Porikli, F., Meer, P.: Covariance Tracking using Model Update Based on Lie Algebra. In: CVPR (2006)
31. Pang, Y., Yuan, Y., Li, X.: Gabor-based region covariance matrices for face recognition. IEEE Tran. Circuits and Sys. for Video Tech. 18(7), 989–993 (2008)
32. Phillips, P., Moon, H., Rizvi, S., Rauss, P.: The FERET evaluation methodology for face-recognition algorithms. Pattern Analysis and Machine Intelligence 22(10), 1090–1104 (2000)
33. Phillips, P., Wechsler, H., Huang, J., Rauss, P.: The FERET database and evaluation procedure for face-recognition algorithms. Image and Vision Computing 16(5), 295–306 (1998)
34. Shen, C., Welsh, A., Wang, L.: PSDBoost: Matrix-generation Linear Programming for Positive Semidefinite Matrices Learning. In: Advances Neural Information Processing Systems (2008)
35. Shen, C., Kim, J., Wang, L.: Scalable large-margin mahalanobis distance metric learning. Neural Networks 21(9), 1524–1530 (2010)
36. Sivalingam, R., Boley, D., Morellas, V., Papanikolopoulos, N.: Tensor sparse coding for region covariances. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010. LNCS, vol. 6314, pp. 722–735. Springer, Heidelberg (2010)
37. Turaga, P., Chellappa, R.: Nearest-neighbor search algorithms on non-Euclidean manifolds for computer vision applications. In: Indian Conf. Comp. Vis. Graph. and Img. Proc., pp. 282–289 (2010)
38. Wang, C., Blei, D., Fei-Fei, L.: Simultaneous image classification and annotation. In: Computer Vision and Pattern Recognition (2010)
39. Weiss, Y., Torralba, A., Fergus, R.: Spectral hashing. Advances in Neural Information Processing Systems 21, 1753–1760 (2009)

40. Wright, J., Ma, Y., Mairal, J., Spairo, G., Huang, T., Yan, S.: Sparse representation for computer vision and pattern recognition. In: CVPR (2009)
41. Yuan, C., Hu, W., Li, X., Maybank, S., Luo, G.: Human action recognition under log-euclidean riemannian metric. In: ACCV, pp. 343–353 (2010)
42. Zhang, H., Berg, A., Maire, M., Malik, J.: SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. In: Computer Vision and Pattern Recognition, vol. 2, pp. 2126–2136. IEEE, Los Alamitos (2006)
43. Zhu, H., Zhang, H., Ibrahim, J., Peterson, B.: Statistical analysis of diffusion tensors in diffusion-weighted magnetic resonance imaging data. Journal of the American Statistical Association 102(480), 1085–1102 (2007)