Similarity Search in Visual Data

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Anoop Cherian

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Doctor of Philosophy

Nikolaos Papanikolopoulos, Adviser

January, 2013

© Anoop Cherian 2013
ALL RIGHTS RESERVED

Acknowledgements

If I have come this far in my journey to earning a Ph.D., it is only because I have been fortunate to work with several bright minds at the right moment. First and foremost, I would like to express my deepest gratitude to my adviser Prof. Nikolaos Papanikolopoulos for instigating in me the passion to take up challenging problems in computer vision and for giving me the freedom, support, and encouragement to investigate those problems to the best of my abilities. I am extremely grateful to him for his enduring support during my challenging times. I would like to thank Dr. Vassilios Morellas for being there as a constant source of wonderful ideas. Dr. Morellas has a unique ability to find parallels between seemingly disparate problems, investigations into which usually open new perspectives and solutions. I would like to thank Prof. Guillermo Sapiro for his innovative and inspirational style of teaching, and introducing me to the concept of sparse coding. A major part of this thesis evolved from ideas I learned from a class he offered in the Spring, 2008. Prof. Arindam Banerjee is a brilliant faculty, and an amazing teacher I got a chance to work with and who has been influential in shaping this work. A special acknowledgment goes to Dr. Suvrit Sra (Max Planck Institute), who entered my research career at the most demanding time, and who has been a great mentor, a constant source of inspiration, ideas, and possibilities, ever since. I would also like to thank members of my Ph.D. committee, Prof. Volkan Isler and Prof. Bérénice Mettler for their support and encouragement to improve this thesis.

There are several other amazing individuals who have directly or indirectly contributed to this work, and I would like to take this oppertunity to acknowledge them. First of all, I would like to thank the members of the Distributed Robotics lab for their help and collaboration, specifically (in alphabetic order) Jon Andersh, Vineet Bhatawadekar, Duc Fehr, Ajay Joshi, Hyeun Min, Evan Ribnick, Ravishankar Sivalingam, Guruprasad Somasundaram, and William Toczyski. The synergy gained by working with multiple individuals on several projects helps

one to master new techniques rapidly. I owe much to several other students I came to know during my graduate studies, and who have been great friends ever since. A few names (in alphabetic order) are Neeraj Gaur, Vineeth Mekkat, Paradeep Mohan, Ganesh Natarajan, Kailash Ramanathan and Ankur Sharma (I apologize if I have missed out anyone, that was not intentional!).

I would like to thank my parents (Mr. A.C. Cherian and Mrs. Ittiannam Cherian) and my brother (Anish Cherian) for their confidence in me. My sincere gratitude to my wife (Mrs. Preethi Ann) for being there and making sure I finish this part of my life as soon as possible with high standards.

I would also like thank my funding agencies and the Minnesota Supercomputing Institute (MSI) for their help and support all these years.

Dedication

To my parents and my wife...

Abstract

Contemporary times have witnessed a significant increase in the amount of data available on the Internet. Organizing such big data so that it is easily and promptly accessible, is a necessity that has been growing in importance. Among the various data modalities such as text, audio, etc., visual data (in the form of images and videos) constitute a major share of this available content. Contrary to other data modalities, visual data pose several significant challenges to storage and retrieval, namely (i) choosing an appropriate representation that can capture the essence of visual data is often non-trivial, and (ii) visual search and retrieval are often subjective, as a result computing semantically meaningful results is hard. On the other hand, visual data possesses rich structure. Exploiting this structure might help address these challenges. Motivated by these observations, this thesis explores new algorithms for efficient similarity search in structured visual data; "structure" is synonymous with the mathematical representation that captures desirable data properties. We will deal with two classes of such structures that are common in computer vision, namely (i) symmetric positive definite matrices as covariances, and (ii) sparse data representations in a dictionary learned from the data.

Covariance valued data has found immense success in several mainstream computer vision applications such as visual surveillance, emotion recognition, face recognition, etc. Moreover, it is of fundamental importance in several other disciplines such as magnetic resonance imaging, speech recognition, etc. A technical challenge in computing similarities on such matrix valued data is their non-Euclidean nature. These matrices belong to a curved manifold where distances between data points are no more along straight lines, but along curved geodesics. As a result, state-of-the-art measures for comparing covariances tend to be slow. To address this issue, we propose a novel similarity measure on covariance matrices-the *Jensen-Bregman LogDet divergence*-which is fast, but at the same time preserves the accuracy of retrieval compared to natural distances on the manifold. To scale our retrieval framework for large covariance datasets, we propose a metric tree data structure on this new measure. Next, as clustering forms an important ingredient for several search algorithms, we investigate this component independently and propose a novel unsupervised algorithm based on the Dirichlet process mixture model for clustering covariance valued data.

The second part of this thesis addresses similarity search problems for high dimensional vector valued data. Such data is ubiquitous not only in computer vision, but also in several other disciplines including data mining, machine learning, and robotics. As the dimensionality of the data increases, computing meaningful similarities becomes increasingly difficult due to the *curse of dimensionality*. Our approach to deal with this problem is inspired from the principles of dictionary learning and sparse coding. Our main idea is to learn an overcomplete dictionary of subspaces from the data so that each data point can be approximated by a sparse linear combination of these subspaces. We introduce a tuple based data descriptor on these sparse combinations-*Subspace Combination Tuple*-that is storage efficient, fast in retrieval, and provides superior accuracy for NN retrieval against the state-of-the-art. These benefits come at a price; the sparse representations are often sensitive to data perturbations. To circumvent this issue, we propose several algorithms for robust dictionary learning and sparse coding.

Extending the sparse coding framework to matrix valued data for hashing covariances forms the content for the third part of this thesis. Towards this end, we propose our novel *Generalized dictionary learning* framework. We describe the theoretical motivations and provide extensive experimental evidence for demonstrating the benefits of our algorithms.

Contents

List of Tables

List of Figures

Notations

Tr	Matrix trace
$\mathcal{S}^d$	Space of $d \times d$ symmetric matrices
$\mathcal{S}^d_{++}$	Space of $d \times d$ symmetric positive definite matrices
$\|.\|$	Matrix determinant
$\otimes$	Kronecker product
relint	Relative interior of a set
int	Interior of a set
$dom\ S$	Domain of a set S
$\mathcal{I}_d$	$d \times d$ identity matrix (sometimes we will drop the suffix when dimensions are unimportant)
$\langle x, y \rangle$	Inner product between two entities x and y
$f \circ g$	The function composition
$\|C\|$ for set C	The cardinality of set C
$\binom{n}{k}$	n choose k

Chapter 1

Introduction

In the recent years, there has been a significant increase in user generated visual content on the Internet. Two major reasons for this are: i) the availability of cheap digital devices such as mobile phones, cameras, camcorders, etc. that have increased the accessibility of people to record visual content, and (ii) the advent of web 2.0, that has enabled a convenient platform for data generators to upload, share, and manage their content. Recent statistics show that around 60 hours of visual content are uploaded every minute[1]. According to Google CEO, Eric Schmidt, 'Every two days now, we create as much information as we did from the dawn of civilization up until 2003'.

The growth of content in the web brings along the big concern about how to connect the information seeker to the right content. Indexing the data via keywords, followed by a similarity search on the keywords list using the query word, has been the traditional mechanism to make the data accessible. This technique is relatively easy when dealing with the text data modality (in the form of text documents, instant messages or tweets). This is because, text has fewer free variables that can 'go wrong', such as spelling or contextual meaning. On the other hand, in visual content, there are several free variables such as lighting, camera pose, color, texture, camera sensor properties, scene noise, real-world to image plane projection distortions, etc. due to which exact matching of a query to its semantic nearest neighbor is more difficult. As a result, representing visual information content, indexing and retrieving it, is still a problem demanding significant advancements. Towards this end, this thesis focuses on developing efficient mathematical models and algorithms for defining similarity in visual data content, for

[1] http://www.youtube.com/t/press_statistics as of June 2012.

improving the quality and speed of content retrieval.

Similarity search or Nearest Neighbor (NN) retrieval is a fundamental subroutine in many areas of computer science, including computer vision, machine learning, robotics and data mining. A few captivating applications from computer vision are: (i) indexing data from medical diagnostics such as CT-scans, MRI imaging, etc. for automated analysis of healthy physiology together with detecting abnormalities, (ii) analyzing patient activities from diagnostic videos for automatic detection of physical or mental disorders, (iii) recognizing the face of a traffic violator captured by a traffic video camera, and (iv) recognizing objects and landmarks for the purpose of robot localization and map building. All these and many more commercial applications depend on nearest neighbors as one of their core algorithmic components. The performance of these applications depends on the efficiency with which the nearest neighbors problem is tackled. As the size of the database to be searched scales into billions, finding the nearest neighbor efficiently for a given query entity becomes phenomenally challenging.

This thesis looks into the problem of nearest neighbors retrieval in visual data. Depending on the application, visual data is often represented in a variety of mathematical forms. Second order feature statistics such as feature covariances are useful for several computer vision applications, while vector representations such as histograms of filter outputs, raw image pixel values, etc. are useful for certain other applications. These two data representations have entirely different mathematical properties and thus require different strategies for implementing NN retrieval. This thesis will explore NN algorithms for these two classes of such data representations, namely (i) data as symmetric positive definite covariance matrices, and (ii) data as high dimensional vectors. The former datatype has been growing in importance in the recent years in computer vision due to its ability to fuse multiple vector-valued data features into a compact representation. In addition, it provides robustness to data distortions and missing information. A few important applications that use covariance matrices as their basic datatype are: (i) people appearance tracking for video surveillance, (ii) face recognition, (iii) action recognition, and (iv) diffusion tensor imaging. On the other hand, high dimensional vector-valued data is ubiquitous, not only in computer vision, but also in many sub-areas of data mining and pattern recognition. Popular examples in the vision domain for this datatype are data descriptors such as Scale Invariant Feature Transform (SIFT), Speeded-Up Robust Features (SURF), Generalized Image Transform (GIST), and shape contexts.

In the following, we will elucidate the importance of our chosen data representations explicitly through various applications. This paves the way to understanding the importance of the algorithms that we will later introduce. We will also provide an overview of this thesis, along with a summary of our contributions.

1.1 Covariance Data

Several computer vision and machine learning tasks depend on structured data where instead of vectors, one uses richer representations of data such as graphs, strings, or matrices. One particular class of such structured data that has been growing in importance in the computer vision community lately is the class of Symmetric Positive Definite (SPD) matrices. These matrices arise naturally as covariance matrices of feature vectors from image regions and because they offer compact, informative, and empirically beneficial feature descriptors, they are very popular in a variety of applications.

A covariance descriptor is nothing but the covariance matrix of features from an image region, and captures the second order statistics of that image region. Mathematically,

Definition 1. *Let $F_i \in \mathbb{R}^p$, for $i = 1, 2, \cdots, N$, be the feature vectors from the region of interest of an image. The Covariance Descriptor of this region $C \in \mathcal{S}_{++}^p$ is defined as:*

$$C = \frac{1}{N-1} \sum_{i=1}^{N} (F_i - \mu_F)(F_i - \mu_F)^T \tag{1.1}$$

where $\mu_F = \frac{1}{N} \sum_{i=1}^{N} F_i$, is the mean feature vector and $\mathcal{S}_{++}^p$ is the space of $p \times p$ Symmetric Positive Definite (SPD) matrices.

To bring out the importance of covariance matrices in computer vision, we concisely review a few applications in which these data descriptors have found success. In the medical arena, Diffusion Tensor Imaging (DTI) is an established technique that depends on 3×3 positive-definite matrices. These matrices are used to track the direction of diffusion of water molecules in the human brain. It can be shown that the largest eigenvalues of the covariances from Magnetic Resonance Imaging (MRI) scans of brain voxels correspond to the direction of water diffusion, and is used to establish the connectivity of different parts of the brain. This idea is the driving force behind several important DTI applications such as the diagnosis of neuro-psychiatric

illnesses [Alexander et al., 2002; Zhu et al., 2007]. It also plays a predominant role in diagnosing cardiac illnesses [Peyrat et al., 2007; Gil et al., 2010]. Positive definite matrices play a key role in Tensor-Based Morphometry (TBM), which deals with the detection and analysis of neuroanatomical deformations in the brain structures through MRI scans. TBM enables recovering three dimensional brain changes by registering the MRI scans to an anatomical brain structure, which is essential in the diagnosis of several diseases including Alzheimer's disease, brain atrophy and dementia [Leow et al., 2009; Lepore et al., 2007; Chiang et al., 2007].

Extending the understanding of covariances in the DTI setting, *structure tensors* have been suggested for several mainstream computer vision applications such as background subtraction [Brox et al., 2003], background subtraction [Caseiro et al., 2010], etc. The gist of such an idea is to replace every pixel in an image with a covariance tensor computed from features (such as the pixel intensity, pixel coordinates, and the intensity gradients) in the neighborhood of the given pixel. Although such an approach increases the processing requirements for the concerned application, this tensor representation is often seen to capture feature correlations adequately thereby improving the accuracy. Moreover, as we subtract off the feature mean when computing the covariances, this descriptor is invariant to static changes in the image such as DC noise or illumination.

One of the most successful application using covariances is region based visual tracking. Covariance matrices derived from appearance silhouettes are known to be robust to affine transformations, to illumination changes, and to variations in the camera parameters [F. Porikli, and O. Tuzel, 2006; O. Tuzel, F. Porikli, and P. Meer., 2006]. This enables covariances to be suitable for applications such as multi-camera object tracking. In addition to these interesting properties, it was shown in [O. Tuzel, F.Porikli, and P. Meer, 2006] that these descriptors can in fact be computed in real-time using integral image representations. This has enabled the proliferation of covariances into many other important areas such as real-world surveillance applications, human detection [Tuzel et al., 2007], object and people tracking, human computer interaction, activity analysis, contour tracking [Ma and Wu, 2011], robotics, and autonomous vehicle navigation [Min et al., 2011].

Covariances provide multi-feature fusion while capturing the second order feature statistics. Analogous to kernel matrices used in machine learning, covariances ignore the features themselves, but capture the feature correlations. As a result, they provide a single compact mathematical object that can be compared against one another, independent of the feature types or

feature dimensionality. This property of covariances has witnessed a strong interest in the recent years, resulting in many vision applications such as robust face recognition [Pang et al., 2008] that uses covariances of Gabor features, emotion recognition using SIFT descriptors [Zheng et al., 2010], and human action recognition using covariances of optical flow [Guo et al., 2010; Yuan et al., 2010]. Other interesting applications are: license-plate detection [Porikli and Kocak, 2006], traffic-sign recognition [Ruta et al., 2009], matching groups of people [Cai et al., 2010], crowd behavior monitoring [Sharif et al., 2008], and scene recognition [Ge and Yu, 2008], to name a few. Application of covariances as data descriptors is not limited to computer vision. For example, it has been used in speech recognition [Ye et al., 2008], and acoustic compression [Shinohara et al., 2010].

We note here in passing that covariance matrices are important objects in several other disciplines as well, and our ideas could be extended to these domains when required. Popular examples of symmetric positive definite matrix valued objects are kernel matrices in machine learning, as noise covariances in sensing and estimation theory, correlation matrices in signal processing, semi-definite constraints in optimization theory and control, etc. In the following, we will briefly review the contributions of this thesis to deal with NN problems on covariance datasets.

Jensen-Bregman LogDet Divergence

Covariance valued data, although has several interesting invariant properties, comes at a heavy price; they do not adhere to the traditional Euclidean geometry. Covariance matrices are Symmetric Positive Definite (SPD) objects that inhabit a Riemannian manifold of negative curvature, such that the distance between data points are no more along straight lines, rather along curved geodesics over the manifold. This introduces challenges in computing nearest neighbors for data having this representation. Traditionally, the natural Riemannian metric, such as the geodesic distance along the manifold, has been used to compare SPD matrices. Such distances ask for the computation of the generalized eigenvalues of the data matrices involved, which is a computationally expensive operation. This issue becomes worse when dealing with matrices of large dimensions or when working with gradient based algorithms such as clustering on the Riemannian manifold.

There have been several approximations to the Riemannian metric suggested to alleviate

this technical difficulty of comparing such matrices. Such approximations make the computations easier, but sacrifices accuracy. In this thesis, we propose a novel similarity measure on covariances, the *Jensen-Bregman LogDet Divergence* (JBLD). Our measure is computationally superior to existing methods, and does not sacrifice the performance of the host applications. We analyze several theoretical properties of this measure including its connections to well-established metrics on this data space. Next, we demonstrate how JBLD can be used for NN retrieval on large covariance datasets. To this end, utilizing the fact that the square-root of JBLD is a metric, we describe a metric tree data structure for NN retrieval. As a part of this framework, we propose a K-means clustering algorithm using JBLD and provide detailed analysis of its convergence properties. We also propose an alternative variant of JBLD, which we call Approximate JBLD (AJBLD). This variant is a lower bound to JBLD and takes much less storage, at the same time enabling offline computations. However, it sacrifices accuracy to some extent. We demonstrate the effectiveness of JBLD for a variety of simulated and real-world applications ranging from people tracking to activity recognition.

Dirichlet Process Mixture Model on Covariance Datasets

When datasets scale, *branch and bound* is an effective heuristic to focus the nearest neighbor search to regions of the dataset that have a greater chance of hosting the true nearest neighbor. Clustering is one of the core algorithmic components of such strategies (e.g., metric trees). When dealing with covariance data, the clustering algorithms thus far have been of K-means type, where the number of clusters in the data needs to be pre-specified. This includes models such as Expectation-Maximization (EM), spectral clustering, etc. When dealing with modern day applications such as video surveillance or face recognition, the dataset scales in no time. As a result, describing such model parameters is difficult or sometimes impossible, motivating investigations toward unsupervised clustering schemes. In the area of DTMRI, one of such non-parametric mechanisms has been to use kernel density estimation. This works well when the structure tensors uniformly occupied an image lattice. Unfortunately, in applications that we described above, such spatial structure might not exist. In this thesis, we take cues from Bayesian learning and propose a Dirichlet Process Mixture Model (DPMM) framework on covariance datasets. DPMM is a well-known statistical idea for non-parametric clustering in which the cluster distributions are sampled from hyper-priors. Inference in DPMM result in fitting appropriate models to the data automatically, which is essentially the problem we would like to

address here. Unfortunately, DPMMs for covariance valued data have not been addressed before; an important gap that we will fill in this thesis. We propose DPMMs for three probability measures on covariances, of which our primary focus will be on the Wishart-Inverse-Wishart conjugate pair. These models are analyzed for effectiveness on real-world datasets and demonstrate promise towards dataset scalability. We also introduce a new clustering evaluation metric: *purity*, which helps us understand the nature of clustering produced by an unsupervised technique.

1.2 Vector Data

Vector valued data, in the form of image feature histograms or vectorized image patches, is a common datatype not only in computer vision, but also in several other areas such as data mining and pattern recognition. As the data dimensionality increases, it becomes difficult to find the nearest neighbor of a query, due to the *curse of dimensionality*. That is, it becomes difficult to distinguish between near data points and far data points when the dimensionality is high. In this thesis, we deal with NN retrieval problems associated with such high dimensional data. Such data descriptors arise abundantly in computer vision. A few representative examples are: Scale Invariant Feature Transforms (SIFT) [Lowe, 2004], Spin Image descriptors used in 3D object recognition [Johnson, 1997], Histogram of gradients [Dalal et al., 2006], Shape Contexts [Mori et al., 2001], Speeded Up Robust Features (SURF) [Bay et al., 2008]. These descriptors form the foundations for several important vision applications such as object recognition, tracking, multiview 3D reconstruction [Snavely et al., 2006], video search engines [Sivic and Zisserman, 2003], etc. In this thesis, we base our experiments predominantly on the SIFT descriptors. There are three reasons for this, namely SIFT (i) is a point descriptor and is used in many fundamental computer vision algorithms, (ii) is high dimensional and, (iii) can be seen as representatives of histogram based data descriptors (most of the popular data descriptors in computer vision are histogram based).

Sparse Coding for NN Retrieval

Sparse coding is the technique of representing the data points as a sparse linear combination of basis vectors from a non-orthogonal dictionary. Such sparse representations have recently attracted a lot of attention in applied mathematics due to their flexibility to adapt to the data. Our

primary contribution to NN retrieval on the vector-valued datatype is in establishing the bridge between the concepts of hashing for NN retrieval and sparse coding. Towards this objective, we propose a novel tuple based representation of the data, *Subspace Combination Tuple* (SCT). Hashing schemes specify certain conditions for their successful operation. We demonstrate theoretically how sparse coding adheres to these conditions. Furthermore, sparse coding helps in the resultant *compressed* data vectors to avoid the curse of dimensionality so that any efficient NN scheme that works on low dimensional data (such as k-d trees) can be easily adapted. This, in fact, provides faster and accurate NN. Modern day NN algorithms on vector data compromise between accuracy and retrieval speed, while our scheme serves both these demands.

Even though our scheme is theoretically appealing, it has to deal with a few practical issues. An important problem is how to define a neighborhood in the sparse space. For example, while sparse coding, the data point gets projected into high dimensional non-orthogonal subspaces. Our method utilizes an indexing of these subspaces for generating a unique hash code for each data point. Since there are several subspaces along the direction of distribution of data, those points that lie at the boundaries are prone to change subspaces under data perturbations, which will fail our indexing mechanism. A way to mitigate this problem is to design the subspaces such that these subspace movements are minimized. This can be implemented in two ways: (i) by understanding the statistical distribution of data and aligning the subspaces such that perturbations are subsumed, and (ii) shifting the data back to its subspace by learning the worst case perturbations. Both these schemes come under our robust dictionary learning approach, the former is statistical, while the latter scheme utilizes the principles of robust optimization.

Probabilistic Denoising

Assuming data vectors as finite length discrete signals and data perturbations are noise, if we remove the noise from the signal, then the filtered signals should belong to the same subspaces. This idea leads to a denoising approach to NN retrieval. To this end, we study the perturbation model of SIFT descriptors and propose a Gaussian-Laplacian denoising model. Since this density model does not have a closed form, we approximate it with a two stage denoising system; a cascade of Gaussian noise model with a Laplacian model. We propose efficient algorithms to solve the resulting optimization problem, later showing experiments on simulated and real-world data. Our results delineate the effectiveness of our algorithm for NN retrieval.

Robust Optimization

The probabilistic model is limited in application since we need to know the probability density of noise, which is difficult in practice. A natural choice to deal with this impediment is to resort to robust optimization principles. The basic idea is to define the set of all probability densities that the perturbation might undergo into one *uncertainty set*. We find the worst case uncertainty in this set, which when added to the data leads to its immunity to perturbations. Along this line, we develop a min-max type of robust optimization problem on the traditional sparse coding formulation and provide a simple algorithm using Lagrangian duality. We demonstrate the effectiveness of our algorithm on SIFT datasets under various noise and deformation models.

Robust Sparse Coding

The earlier two approaches concentrated on aligning the basis in the sparse coding dictionary to achieve robustness. An alternative, and more intuitive scheme would be hierarchical sparse coding. That is, if a data vector is proximal to the boundary of a subspace, it is prone to change the basis even under small perturbations. A possibility to avoid such subspace transitions is to find a larger subspace that subsumes the two smaller subspaces so that the data point is no longer near any boundary. Such an idea can be efficiently implemented using the LARS algorithm for sparse coding in which the subspaces are found in successive steps, each step refining the representation of the data. This means, the first subspace should subsume all the other subspaces that will be added by later steps, and so on. Interestingly, such a scheme can be easily implemented by changing the regularization of the sparse coding problem. We investigate this possibility and propose a novel sparse coding scheme called Multi-Regularization Sparse Coding. We analyze the properties of this scheme thoroughly and demonstrate experimentally its effectiveness on a variety of datasets such as SIFT and spin images. We also show the scalability of this algorithm to large datasets.

1.3 Sparse Coding for Covariances

In the final part of this thesis, we show an idea of how we can adapt the sparse coding algorithm for NN retrieval to hash covariance matrices. Note that sparse coding of vectorized covariances

might lead to poor accuracy, as we do not take into account the manifold structure. To circumvent this issue, we use the dictionary learning idea. Our main strategy is to approximate a given covariance matrix as a sparse linear combination of rank-one dictionary atoms, where in we will be learning this dictionary from the data covariances themselves. Later, we will use the index of the active support of the sparse code for hashing covariances. We call such a dictionary learning algorithm *Generalized Dictionary Learning*. The scheme is computationally efficient and our algorithm scales well with data. We compare this algorithm on several datasets. Our experiments demonstrate the superiority of our algorithm in terms of accuracy and retrieval speed.

1.4 Thesis Organization

This thesis is organized as follows: we begin with a review of the state-of-the-art methods for NN retrieval in Chapter 2. In Part I of this thesis, we describe algorithms for covariance data, Part II deals with vector-valued data, and Part III deals with sparse coding on covariances. In Chapter 3, we will introduce our JBLD measure on covariance matrices, along with its theoretical analysis and experimental study. In Chapter 4, we will describe our non-parametric DPMM clustering algorithm on covariances. In Part II of this thesis, we will first introduce our sparse coding based NN scheme in Chapter 5. This will be followed by our algorithms for robust NN schemes. The robustness can either be on the sparsifying dictionary, which is tackled in Chapter 6, or it can be in the sparse coding stage as described in Chapter 7. Finally, in Part III of this thesis, we will describe a mechanism to sparse code covariances. This is described in Chapter 8. Chapter 9 will summarize our contributions. Since we deal with several topics on NN retrieval on several data types, a brief overview of the thesis organization is presented in Figure 1.1 for the ease of understanding.

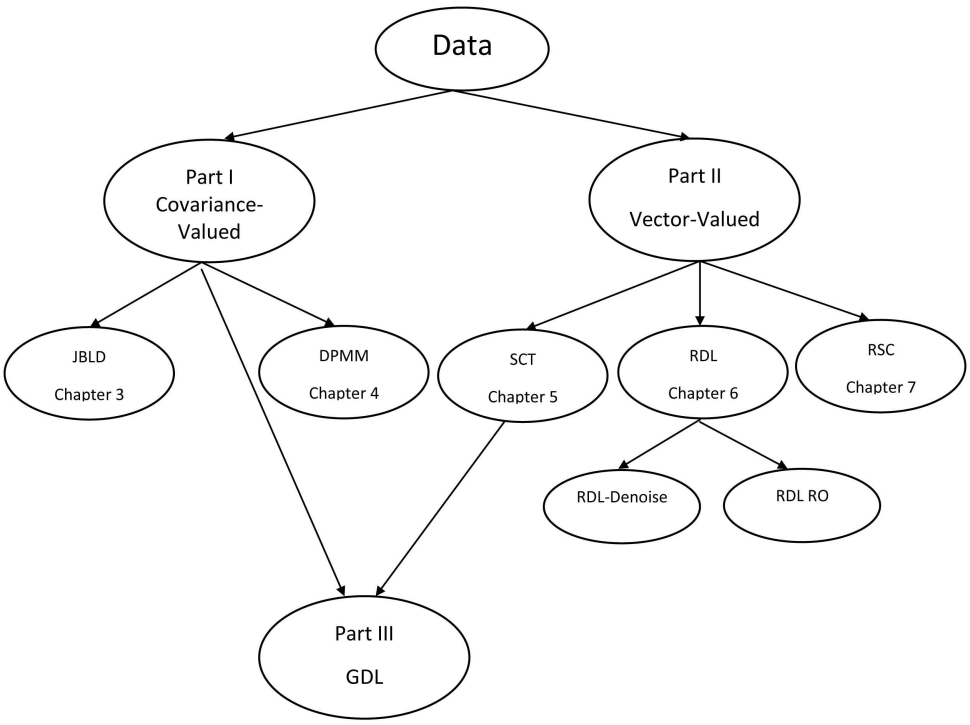


Figure 1.1: Organization of this thesis.

Chapter 2

Related Work

NN retrieval is a fundamental problem in computer science that has been investigated from several facets over the past few decades. The amount of research that has gone into this problem is overwhelming that it is impossible to provide a comprehensive survey of the approaches. Thus, in this chapter, we will restrict our discussion to a few fundamental concepts in NN retrieval with regard to applications in computer vision and data mining that are closely related to our contributions.

2.1 Exact NN Retrieval

To get started, let us first define the problem of NN retrieval in mathematical terms.

Definition 2 (NN)**.** *Given a dataset $\mathcal{D}$, a distance function $dist$ and a query q, we define the* Nearest Neighbor *(NN) of q as:*

$$\mathrm{NN}(q) := \left\{x' \in \mathcal{D} | \forall x \in \mathcal{D}, dist(q, x') \leq dist(q, x)\right\}. \tag{2.1}$$

In practical problems, it is often seen that rather than restricting oneself to the *nearest* neighbor to a query item, it is advantageous to find the k-Nearest Neighbors (kNN). This is especially useful in cases where the query is subjective, such as in a content based image retrieval application. It might be useful to retrieve the first kNNs and provide it to the user so that a decision on the most appropriate retrieved item can be made subjectively. Another such application is that of recommender systems for content recommendations. We can extend Definition 2.1 and define kNN as follows:

Definition 3 (kNN)**.** *Given a dataset $\mathcal{D}$, a distance function $dist$ and a query point q, we define the set S of* k-Nearest Neighbors *of q as:*

$$\text{kNN}(q) := \left\{ S \subseteq \mathcal{D} \wedge |S| = k | \forall x' \in S, \forall x \in \mathcal{D} - S, dist(q, x') \leq dist(q, x) \right\}. \quad (2.2)$$

A straightforward approach that one can design for NN retrieval could be to partition the data into predefined structures, so that approaches such as *divide and conquer* can be easily applied. For example, let us take the case of one dimensional data uniformly distributed along the x-axis. To find the NN for a given data point, a straightforward first step is to split the data into two parts, to the left of the y-axis (negative data) and to the right (positive). We can define several other hyperplanes perpendicular to the x-axis at intervals, such that the NN algorithm boils down to searching recursively if the query falls to the left side or right side of the hyperplane under consideration. Such NN approaches come under the broad category of *binary space partitioning* (BSP). In a multi-dimensional data setting, BSP manifests itself in the form of classical data structures such as the k-d trees [Robinson, 1981]. Analogous to our one-dimensional example, in a k-d tree, a hypercube is embedded in to the data space, each edge of the hypercube producing a binary partition of the multi-dimensional space. Other indexing structures that come under BSP are B+ trees and Range trees.

A more intelligent way to organize the data (and thereby improve the performance of NN retrieval) could be to *look* at the distribution of the data and organize the indexing structure accordingly. Algorithms confiding to such an idea fall under the broad category of *data space partitioning*. The simplest of such approaches is the binary search tree for one dimensional problems. In higher dimensions, sophisticated data structures such as R-trees, R+trees [Guttman, 1984], ball-trees [Omohundro, 1989], etc. are popular.

A specific type of such data partitioning methods that we will be using in this thesis is the Metric Tree (MT) [Ciaccia et al., 1997]. An MT is basically a ball-tree in which the underlying distance used is a metric. In this data structure, the data is partitioned recursively into non-overlapping clusters (or partitions), each cluster characterized by a centroid and a radius. Given a query entity, a procedure similar to a binary search tree is used to traverse the clusters using the underlying metric. The advantage of an MT is that the triangle inequality induced by the underlying metric helps in pruning out parts of the data from being traversed, thus providing a computational edge to NN retrieval. A survey of other NN algorithms using space and data partitioning methods can be found in books such as [Knuth, 1973; Cormen, 2001].

2.2 Approximate NN Retrieval

Exact NN algorithms are fundamentally limited because their performance degrades as the dimensionality of the data increases. This is classically known as the *curse of dimensionality*. Intuitively, as the dimensionality of the data increases, the volume of the data space increases exponentially. Assuming the number of data points is constant, the dataset becomes sparse in the data space so that it becomes difficult to apply any statistical reasoning on the distribution of these points, subsequently failing any data space partitioning technique and thereby degrading the performance of the application.

Curse of dimensionality is equally detrimental to space partitioning techniques. To see how, let us take a simple example. Assume that the data is equally distributed inside a unit hypersphere in $\mathbb{R}^d$. The volume of this hypersphere $V_s(d)$ is given by:

$$V_s(d) = \frac{\pi^{d/2}}{\Gamma(\frac{d}{2}+1)}, \tag{2.3}$$

where $\Gamma(x) = \int_0^\infty \exp(-t)t^{x-1}dt$ is the Gamma function. Now, let us assume that we use a k-d tree (with 2^d partitions) to search for NN of a given query point. As was mentioned in the previous subsection, a k-d tree embeds a d-dimensional hypercube into the data space. Assuming a side length of 2 (equal to the diameter of the hypersphere), the volume of this hypercube $V_c(d)$ will be 2^d. Comparing the sizes of these two spaces by taking the ratio of the their volumes, we see that as the dimensionality of the space increases,

$$\lim_{d\to\infty} \frac{V_s(d)}{V_c(d)} \to 0. \tag{2.4}$$

That is, as the data dimensionality increases, the number of data partitions produced by the k-d tree increases exponentially, but the volume of each of these partitions has to limit to zero. This means that the relative distance between two nodes in the k-d tree approximates to zero as the dimensionality increases. Or in other words, this suggests that it is impossible to distinguish near or far points described by the k-d tree since the distance between corners of the embedded hypercube limits to zero. A consequence of this issue is that the space partitioning method for exact NN will not provide any useful benefit against doing a linear scan on the entire data. It might seem comforting that (2.4) happens when the data dimensionality goes to infinity; unfortunately as empirically shown in [Beyer et al., 1999], this phenomenon happens when the number of dimensions crosses a value as low as 20.

A primary course of action to avoid this limitation is to relax the exact NN requirement (as stated in Definition 2.1) and search for *Approximate* NNs (ANN). Such an approach seems reasonable considering the fact that the features and the metrics used to find NNs in real-world problems are generally heuristic. For example, using the statistical measure such as KL divergence to retrieve NN. Such measures are heuristic choices, which might work well for one dataset, while it fails for another one. Asking for exact NN via such heuristic choices is probably too much to ask for in that respect. Thus finding approximations to the NN, by retrieving some neighbor in the neighborhood of the true nearest neighbor might be a better practical idea. When we accommodate this neighborhood relaxation into the definition of NN, we arrive at the following definition of Approximate NN:

Definition 4. *Given a dataset $\mathcal{D}$, a distance function $dist$ and a query data point q, we define an* Approximate Nearest Neighbor *(ANN) of q as:*

$$\text{ANN}(q) := \left\{x' \in \mathcal{D} | \forall x \in \mathcal{D}, dist(q, x') \leq (1+\epsilon) dist(q, x), \textit{for } \epsilon > 0\right\}. \tag{2.5}$$

This scheme is also sometimes referred to as c-approximate NN, where $c = (1+\epsilon)$. The above definition means that instead of requesting the algorithm for the true nearest neighbor for a query point, a neighbor anywhere inside an ϵ-ball around the true neighbor is permitted. There have been different ANN schemes explored in literature [Arya et al., 1998; Clarkson, 1994; Kleinberg, 1997; Liu et al., 2004]. In computer vision, one of the most popular ANN algorithms has been the *Best Bin First* (BBF) strategy [Beis and Lowe, 1997]. Here the key observation is that algorithms such as k-d trees in high dimensions, spend most of their time in backtracking to check if a solution exists that is better than a candidate solution. BBF strategy is to do this backtracking more efficiently by keeping a priority queue of the candidate distances. While backtracking, the candidates are considered according to their priorities in this queue for comparing against the query point. Often, it happens that early stopping of this backtracking procedure produces an approximate NN with high probability.

Of the various ANN schemes, the most popular has been *Locality Sensitive Hashing* (LSH) [Indyk and Motwani, 1998]. Since the algorithms that we develop in the second part of this thesis follow LSH techniques, we will review this topic in sufficient detail next.

2.2.1 Locality Sensitive Hashing

An intuitive way to improve ANN retrieval to sub-linear query times is to utilize properties of the data such that points with similar properties produce similar compact codes. Such codes, generally referred to as *hash codes* in the computer science literature, can then be used to directly index a hash table, leading to faster access. Locality Sensitive Hashing (LSH) [Indyk and Motwani, 1998; Indyk, 1998] is such a method that extends the ANN idea introduced in the last section into a probabilistic framework using hash functions. These functions are chosen in such a way that neighbors within an ϵ-ball are *hashed* to the same bucket with high probability. Mathematically,

Definition 5. *Let* $dist : \mathcal{D} \times \mathcal{D} \rightarrow \mathbb{R}$ *be some dist function and let* $\mathcal{H} = \{h_i : \mathcal{D} \rightarrow U, i = 1, 2, \cdots, K\}$ *is a family of hash functions. Then,* $\mathcal{H}$ *is said to be* suitable *for LSH if for every* $x \in \mathcal{D}$ *and a query point* q*, we have:*

$$dist(q,x) \begin{cases} \leq r_1 \Rightarrow Prob\left(h_i(q) = h_i(x)\right) \geq p_1 \\ > r_2 \Rightarrow Prob\left(h_i(q) = h_i(x)\right) < p_2, \end{cases}$$

for r_1, r_2 *are scalars,* $r_1 < r_2$ *and* $p_1, p_2 \in [0,1], p_1 > p_2$.

If $\mathcal{H}$ is chosen properly, then it is shown in [Indyk and Motwani, 1998] that LSH can lead to an ANN retrieval in $\mathcal{O}(|\mathcal{D}|^{\rho})$ query time, where $\rho = \log(1/p_1)/\log(1/p_2)$, and $|\mathcal{D}|$ represents the cardinality of the dataset. The constant ρ captures the efficiency of the hashing algorithm. That is, if $\mathcal{H}$ is so good that near data points in the original space produce hash codes that map to the same hash bucket with very high probability (p_1), while dissimilar points map with a low probability (p_2) to the same hash bucket, then $\rho \ll 1$. This implies that the complexity of ANN retrieval will be very small as implied by $\mathcal{O}(|\mathcal{D}|^{\rho})$.

One such family of $\mathcal{H}$ is the set of *p-stable* distributions [Indyk, 2000] (See Appendix A for a review). The main property of these distributions, with regard to ANN retrieval, is that the projection of the data points onto hyperplanes sampled from p-stable distributions preserve the distribution. As a result, we can map distances between data points into probabilities as given in Proposition 5. One of the popular such LSH implementations is E2LSH (short for Exact Euclidean LSH) [Datar et al., 2004], in which the hyperplanes are sampled from a Gaussian distribution (which is a 2-stable distribution). The algorithm projects the high dimensional data points onto the real-line, which is later organized into hash buckets. In mathematical terminology,

suppose we have a data vector x, and a set of hyperplanes $\mathcal{A} = \{a_i \in \mathbb{R}^d, i = 1, 2, \cdots, N\}$, where each coordinate of $a_i^j \sim \mathcal{N}(0, \sigma), j = 1, 2, \cdots, d$. Then we can define a hash-function $h_i(x) = \lfloor \frac{<a_i, x> + b}{w} \rfloor$, where b and w are parameters of the hash family, w adjusts the diameter of the hash bucket while b adjust its center. It is clear from above that such a scheme is very simple. More to it, we can in fact compress the hash keys into compact binary codes for efficient memory indexing, and use the Hamming distance to find the nearest neighbor. Theoretically, it has been shown that the Hamming distance on these binary codes actually approximates the Euclidean distance asymptotically [Andoni and Indyk, 2006].

2.2.2 Learning based ANN

The hash functions introduced in the previous section do not make any connection to the actual distribution of the data. Can we improve the ANN performance by learning the hash functions from the data? Further, will knowing this distribution produce shorter hash codes (recall Huffman codes), while preserving the relevant similarity properties? Trying to answer these questions resulted in finding several successful ANN algorithms lately. Early efforts toward this have been to extend indexing techniques used in text search domain (such as Latent Semantic Indexing) to visual content search. In [Salakhutdinov and Hinton, 2009], a Restricted Boltzmann Machine (RBM) is suggested. RBMs are multi-layer neural networks typically used for solving combinatorial problems. The adaptation of RBMs for ANN is achieved by a deep learning type of methodology in which each layer of the neural network tries to map the inputs onto an output layer of binary streams that has fewer bits. It is often found that training RBMs is difficult, especially when the input data dimensions are large. An alternative strategy to learn compact hash codes is to adapt Boosting [Shakhnarovich et al., 2003]. That is, to combine the results a set of weak or approximate learning algorithms to achieve superior accuracy.

The LSH algorithms that we discussed till now emphasized on mapping similar data points into the same hash bucket, while making heuristic choices on the number of hash buckets or the hash code length. Once similar data is made to collide to the same bucket, linear search is used to find the ANN inside the bucket. When working with large datasets, this results in slower access times and large memory footprints. Prioritizing fast access while sacrificing the accuracy of the search results paved the way for the design of optimal hash codes that capture the data properties, while doing away with the need to store the data points in memory. In [Weiss et al., 2009], an effort towards such a scheme can be seen in which parallels between efficient

graph partitioning and optimal codes is visited. The algorithm named *Spectral Hashing* tries to minimize the sum of the hamming distances between pairs of input data points weighted by an affinity matrix constructed on their respective feature vectors. The affinity matrix is considered to be the Gaussian kernel for tractability of the solution. To begin with, the algorithm goes about building a graph Laplacian over the training data. Utilizing the connection between the eigenvectors of the graph Laplacian to the eigenfunctions of the Laplace-Beltrami operators on manifolds and further assuming that the data comes from a separable distribution (such as the uniform distribution or multi-dimensional Gaussian), the paper proposes an efficient algorithm for producing compact codes. There have been several approaches to ameliorate this heuristic dependency on the Gaussian kernel. In [Raginsky and Lazebnik, 2009], a shift invariant kernel (SIKH) is suggested that is shown to provide better performance and at the same time providing a suitable platform for theoretical analysis. The dependence on a separable distribution is ameliorated by assuming a low-dimensional data distribution in a high dimensional ambient space in [Liu et al., 2011]. The paper tries to estimate the data distribution using a few data cluster centroids (called *anchors*) learned from the data (using algorithms such as K-means). Learning such low dimensional structures has also been adopted in other hashing algorithms such as [Jegou et al., 2008].

There have been other efforts to elevate standard machine learning techniques to address LSH. The basic LSH strategy is restricted because it works with random linear projections. In [Kulis and Grauman, 2009], the LSH idea is extended to non-linear functions through a kernel mapping, the subsequent hashing algorithm referred to as Kernelized LSH (KLSH). Other kernel based hash functions are described in [Z. et al., 2006]. A general trend in all these methods is the selection of a set of random hyperplanes to which the data is projected, later using these projections for hashing in the binary Hamming space. It is often seen that the selection of these hyperplanes have a tremendous impact on the accuracy of the respective algorithm.

In [Zhang et al., 2006], an SVM based discriminative NN algorithm is suggested. A semi-supervised metric learning strategy for learning the hash functions is suggested in [Wang et al., 2010]. Metric learning methods have been adapted to the problem of ANN and hashing in several papers [Jain et al., 2008; Chechik et al., 2009; Ram et al., 2009; Frome et al., 2007]. The general trend in these methods is to distort the natural Euclidean geometry between the data points in such a way that points that are *known* to belong to the same class are constrained to have lower distances, at the same time data points belonging to disparate classes are scaled

to be farther apart. To this end, a constrained optimization problem is solved by minimizing the Mahalanobis distance between the data points under a set of constraints that capture the nearness properties of the points. On the negative side, as the number of such constraints increases, it is often found to be difficult to solve the optimization problem.

More recently, adaptations of the standard K-means algorithm for the purpose of similarity search have been suggested in [Jegou et al., 2011]. The main idea is to build a set of codebook centroids over the database points based on subspace product quantization. That is, a given high dimensional data vector is segregated into a set of equal length short sub-vectors, each sub-vector is assumed to belong to a different data subspace. For each subspace a set of cluster centroids are learned from the data using K-means, later each sub-vector is encoded using these centroids. The paper addresses problems related to the distortion introduced by these sub-vector approximations and builds an efficient ANN scheme based on indexing the data vectors using these sub-vector approximations. Given a query vector, a set of cluster centers at a minimum Euclidean distance from the query are found from each of these subspaces and an inverted file system is used for faster lookup. The subspace determination step is more or less heuristic, and might result in redundant codes if there is dependency between the subspaces. A similar idea but using sparse coding for ANN retrieval is proposed in [Zepeda et al., 2010]. In this work, an inverted file system that uses the support set of sparse codes is used for fast retrieval, but with limited accuracy.

2.2.3 ANN on Non-Metric Spaces

Almost all the algorithms that we described in the previous sections assume that the original data inhabits a metric space. Often data arises as probability densities or histograms [Puzicha et al., 1999; Rasiwasia et al., 2007]. As a result, we might need to use non-metric (dis)similarity measures. In [Cayton, 2008], the standard metric tree data structure is extended to deal with Bregman divergences. These divergences subsume the popular divergences such as KL-divergence, Itakura-Satio diverence, etc. and will play a significant role when we talk about NN on covariance valued data.

2.2.4 ANN on Covariance Data

We note that NN retrieval for covariance matrices itself is still an emerging area, so literature on it is scarce. In [Turaga and Chellappa, 2010], an attempt is made to adapt NN techniques from vector spaces to non-Euclidean spaces, while [Chaudhry and Ivanov, 2010] proposes an extension of the spectral hashing techniques to covariance matrices. However, both these techniques are based on a Euclidean embedding of the underlying Riemannian manifold through the tangent spaces, which are often found to be approximate.

Part I

Covariance Valued Data

Chapter 3

Nearest Neighbors on Covariance Matrices

Covariance matrices provide compact and informative feature descriptors for use in several computer vision applications, such as people appearance tracking, diffusion tensor imaging, activity recognition, among others. A key task in many of these applications is to compare covariance matrices using a (dis)similarity function and the natural choice here is the Riemannian metric corresponding to the manifold inhabited by covariances. But computations involving this metric are expensive, especially for large matrices, and even more so, in gradient-based algorithms. Further, suitability of the metric to enable efficient nearest neighbor retrieval is an important requirement in the contemporary times of big data analytics. To alleviate these difficulties, this chapter proposes a novel dissimilarity measure for covariances, the *Jensen-Bregman LogDet Divergence* (JBLD). This divergence enjoys several desirable theoretical properties, at the same time is computationally less demanding (compared to standard measures). Utilizing the fact that the square-root of JBLD is a metric, we address the problem of efficient nearest neighbor retrieval on large covariance datasets via a metric tree data structure. To this end, we propose a K-means clustering algorithm on JBLD. We demonstrate the superior performance of JBLD on covariance datasets from several computer vision applications.

This chapter is organized as follows. We start with an extensive review of several similarity metrics on covariance matrices in Section 3.1. This is followed by an introduction to the JBLD measure, and exposition of its properties in Section 3.2. We also investigate a few properties of

our measure (such as the anisotropic index) that is useful for DTMRI applications. Section 3.3 discusses the application of JBLD for nearest neighbor retrieval on covariances. Towards this end, we propose a K-means clustering algorithm using JBLD in Section 3.2.6. Experiments and results are presented in Section 3.4.

3.1 Related Work

To put our work into perspective, let us recall some standard similarity measures for covariance matrices. The simplest but naïve approach is to view $d \times d$ covariance matrices as vectors in $\mathbb{R}^{d(d+1)/2}$, whereby the standard (dis)similarity measures of Euclidean space can be used (e.g., ℓ_p-distance functions, etc.). Recall that covariance matrices, due to their positive definiteness structure, belong to a special category of symmetric matrices and form a Riemannian manifold (which is a differentiable manifold associated with a suitable Riemannian metric). Euclidean distances on vectorized covariances ignore this manifold structure leading to poor accuracy. In addition, under this measure symmetric matrices with nonpositive eigenvalues are at finite distances to positive definite covariances. This is unacceptable for a variety of applications, e.g. DTMRI [Arsigny et al., 2008], in which there are physical interpretations associated with these positive definite matrices (such as diffusion of water molecules in brain tissues).

A more suitable choice is to incorporate the curvature of the Riemannian manifold and use the corresponding geodesic length along the curvature of the manifold as the distance metric. This leads to the *Affine Invariant Riemannian Metric (AIRM)* [Bhatia, 2007; Pennec et al., 2006] which is defined as follows: For X, Y in $\mathcal{S}_{++}^d$,

$$D_R(X,Y) := \|\log(X^{-1/2} Y X^{-1/2})\|_{\mathrm{F}}, \tag{3.1}$$

where $\log(\cdot)$ is the *principal* matrix logarithm. This metric enjoys several useful theoretical properties, and is perhaps the most widely used similarity measure for covariance matrices. As is clear from (3.1), symmetric matrices with nonpositive eigenvalues are at infinite distances. The metric is invariant to inversion and similarity transforms. Other properties of this metric can be found in [Bhatia, 2007]. Computationally, this metric can be unattractive as it requires eigenvalue computations or even matrix logarithms, which for larger matrices cause significant slowdowns. A few examples of such applications using large covariances are: face recognition [Pang et al., 2008] (40×40), and emotion recognition [Zheng et al., 2010] (30×30).

As a result, this metric might not be suitable when dealing with mainstream computer vision applications.

Amongst the many measures that have been proposed to replace AIRM, a closely related one is the *Log-Euclidean Riemannian Metric* (LERM). Considering the log-Euclidean mapping $\log : \mathcal{S}^d_{++} \rightarrow \mathcal{S}^d$, Arsigny et al. [Arsigny et al., 2006] observed that under this mapping, the Lie group of SPD matrices is *isomorphic* and *diffeomorphic* (smooth manifolds are mapped to smooth manifolds) to the space of symmetric matrices under the group multiplication operator $\odot$ defined as $X \odot Y \triangleq \exp\{\log X + \log Y\}$ for $X, Y \in \mathcal{S}^d_{++}$. That is, the $\log$ is a bijection. Using this mapping, the paper introduces LERM as:

$$D_{le}(X, Y) := \|\log(X) - \log(Y)\|_{\mathrm{F}}. \tag{3.2}$$

On the positive side, LERM maps SPD matrices to a flat Riemannian space (of zero curvature) so that the ordinary Euclidean distances can be used. The metric is easy to compute, and preserves a few important properties of the AIRM (such as invariance to inversion and similarity transforms). In addition, from a practical point of view, since this metric untangles the two constituent matrices from their generalized eigenvalues, the logarithms on each of these matrices can be evaluated *offline*, gaining a computational edge over AIRM. As a result, LERM has found many applications in visual tracking [Li et al., 2008], stereo matching [Gu and Zhou, 2009], etc. On the negative side, computing matrix logarithms can dramatically increase the computational costs. The flattening of the manifold as in LERM often leads to less accurate distance computations, affecting application performance. A more important problem that one encounters when using LERM is that its moments (such as Hessian, etc.) do not have closed forms. Moreover, it is computationally difficult even to approximate these moments due to the need to find derivatives of matrix logarithms. The following proposition shows that LERM is a lower bound to AIRM. This result will come useful later in this thesis.

Proposition 1. *For $X, Y \in \mathcal{S}^d_{++}$, we have: $D_{le}(X, Y) \leq D_R(X, Y)$. Further, the equality holds only when X and Y commute.*

Proof. See Theorem 6.1.4 in [Bhatia, 2007]. □

There have been attempts to use symmetrized *f-divergences* from information theory into developing distances on SPD matrices (Recall that f-divergence measures how much probability distributions differ from one another). One such idea is to view the SPD matrices as being the

covariances associated with zero-mean Gaussian distributions [M. Moakher, and P. Batchelor, 2006], and then use the symmetrized KL-Divergence Metric (KLDM) as the distance between the distributions. This leads to the following definition of KLDM:

$$D_{kl}^2(X,Y) := \frac{1}{2}\operatorname{Tr}\left(X^{-1}Y + Y^{-1}X - 2\mathcal{I}\right) \tag{3.3}$$

This measure does not require matrix eigenvalue computations, or logarithms, and at the same time enjoys many of the properties of AIRM. On the negative side, the measure requires inversion of the constituent covariances, which can be slow (or can even lead to instability when the data matrices are poorly conditioned). A bigger concern is that KLDM can in fact *overestimate* the Riemannian metric as the following proposition shows and thus can lead to poor accuracy.

Proposition 2. *There exist X and $Y \in \mathcal{S}_{++}^d$ such that $D_{kl} > D_R$.*

Proof. Let v_i be the ith eigenvalue of $X^{-1}Y$. Since v_i is always positive, we can write $v_i = \exp u_i$ for $u_i \in \mathbb{R}$. Then from the definitions of KLDM and AIRM, we have:

$$\begin{aligned} D_{kl}^2 &= \sum_{i=1}^{d}\left(\frac{\exp u_i + \exp -u_i}{2}\right) - 1 \\ &= \frac{D_R^2}{2}\sum_{i=1}^{d}\left(1 + 2\frac{u_i^2}{4!} + \cdots\right) - 1. \end{aligned}$$

For a suitable choice of u_i, we have the desired result. □

Another attempt to define a measure on SPD matrices can be seen in [Wang et al., 2004]. This paper introduces a distance based on the Cholesky factorization of the matrices. The idea is as follows: suppose $X = L_1 L_1^T$ and $Y = L_2 L_2^T$ represent the Cholesky decomposition of X and Y respectively, with lower triangular matrices L_1 and L_2, then the Cholesky distance is defined as:

$$D_C(X,Y) = \|L_1 - L_2\|_{\mathrm{F}}. \tag{3.4}$$

Other similarity measures on covariance matrices may be found in [Dryden et al., 2008]. Contrary to their easy formulations and properties close to those of AIRM, the above distances have not been very popular in SPD matrix based applications due to their poor accuracy.

In contrast to all these metrics, the similarity measure that we propose in this chapter is much faster to compute, as it depends only on the determinant of the input matrices, and thus

no eigenvalue computations are required. Moreover, it turns out to be empirically also very effective.

We note that NN retrieval for covariance matrices itself is still an emerging area, so literature on it is scarce. In [Turaga and Chellappa, 2010], an attempt is made to adapt NN techniques from vector spaces to non-Euclidean spaces, while [Chaudhry and Ivanov, 2010] proposes an extension of the spectral hashing techniques to covariance matrices. However, both these techniques are based on a Euclidean embedding of the Riemannian manifold through the tangent spaces, and then using LERM as an approximation to the true similarity.

3.2 Jensen-Bregman LogDet Divergence

In this section, we will introduce our new similarity measure: the *Jensen-Breman LogDet Divergence* (JBLD). Our measure is a symmetric f-divergence similar to the KLDM measure we discussed in the last section. Recall that KLDM used an arithmetic average of the asymmetric Kullback-Leibler divergence. An alternative way to symmetrize the f-divergences is to use the average asymmetric divergence of the distributions from the mean of the distributions. When we adapt this form of symmetrized divergence to covariances, we get the JBLD measure. This idea is detailed below.

We first recall some basic definitions. We remark that although our measure seems natural and simple, to our knowledge it has *not been* formally discussed in detail before. We alert the reader that JBLD should not be confused with its asymmetric cousin: the so-called LogDet divergence [Kulis et al., 2009].

At the core of our discussion lies the *Bregman Divergence* $d_\phi : S \times \text{relint}(S) \to [0, \infty)$, which is defined as

$$d_\phi(x, y) := \phi(x) - \phi(y) - \langle x - y, \nabla\phi(y) \rangle \,, \tag{3.5}$$

where $\phi : S \subseteq \mathbb{R}^d \to \mathbb{R}$ is a strictly convex function of Legendre-type on $\text{int}(\text{dom}\, S)$ [Banerjee et al., 2005]. From (3.5) the following properties of $d_\phi(x, y)$ are apparent: strict convexity in x; asymmetry; non-negativity; and definiteness (i.e., $d_\phi = 0$, iff $x = y$). Bregman divergences enjoy a host of useful properties [Banerjee et al., 2005; Censor and Zenios, 1997], but often their lack of symmetry and sometimes their lack of triangle-inequality can prove to be hindrances. Consequently, there has been substantial interest in considering symmetrized versions such as

Jensen-Bregman divergences [Nielsen and Nock, 2007, 2010; Banerjee et al., 2009],

$$J_\phi(x,y) := \frac{1}{2}\big(d_\phi(x, \frac{x+y}{2}) + d_\phi(y, \frac{x+y}{2})\big), \tag{3.6}$$

or even variants that satisfy the triangle inequality [Banerjee et al., 2009; Chen, 2007].

Both (3.5) and (3.6) can be naturally extended to matrix divergences (over Hermitian matrices) by replacing the inner-product in (3.5) by the trace. This leads to the following matrix variant of the Bregman divergence: for two matrices $A, B \in \mathbb{R}^{p\times q}$ and a convex function $\Phi : S' \subseteq \mathbb{R}^{p\times q} \to \mathbb{R}$, we will have matrix Bregman divergence $D_\Phi(.,.)$ as:

$$D_\Phi(A,B) := \Phi(A) - \Phi(B) - \operatorname{Tr}\left((A-B)^T \nabla\Phi(B)\right) \tag{3.7}$$

We focus on a particular matrix divergence, namely the *Jensen-Bregman LogDet Divergence*, which is defined for X, Y in $\mathcal{S}^d_{++}$ by

$$J_{\ell d}(X,Y) := \log\left|\frac{X+Y}{2}\right| - \frac{1}{2}\log|XY|. \tag{3.8}$$

where $|\cdot|$ denotes the determinant; this divergence is obtained from (3.6) and (3.7) by using $\Phi(X) = -\log|X|$ as the seed function.

3.2.1 Properties

In this section, we will detail several basic properties of JBLD that makes it suitable for several computer vision applications. For $X, Y, Z \in \mathcal{S}^d_{++}$ and invertible matrices A and B, we have the following properties:

(i) $J_{\ell d}(X,Y) \geq 0$ (nonnegativity)

(ii) $J_{\ell d}(X,Y) = 0$ iff $X = Y$ (definiteness)

(iii) $J_{\ell d}(X,Y) = J_{\ell d}(Y,X)$ (symmetry)

(iv) $\sqrt{J_{\ell d}(X,Y)} \leq \sqrt{J_{\ell d}(X,Z)} + \sqrt{J_{\ell d}(Z,Y)}$ (triangle inequality)

(v) $J_{\ell d}(AXB, AYB) = J_{\ell d}(X,Y)$ (affine invariance)

(vi) $J_{\ell d}(X^{-1}, Y^{-1}) = J_{\ell d}(X,Y)$ (invariance to inversion)

Proof. (i) We can rewrite the definition of $J_{\ell d}(X, Y)$ (given in (3.8)) as

$$J_{\ell d}(X, Y) = \log \left| \frac{\mathcal{I}_d + XY^{-1}}{2} \right| - \frac{1}{2} \log \left| XY^{-1} \right|.$$

Assume λ_i represents the ith eigenvalue of XY^{-1}, then

$$\begin{aligned} J_{\ell d}(X, Y) &= \sum_{i=1}^{d} \log \frac{1 + \lambda_i}{2} - \frac{1}{2} \log \lambda_i \\ &= \sum_{i=1}^{d} \log \frac{1/\sqrt{\lambda_i} + \sqrt{\lambda_i}}{2} \\ &\geq 0, \text{ for all } \lambda_i > 0. \end{aligned}$$

(ii) This follows directly from (i). Note that the minimum value of $\sqrt{\lambda_i} + 1/\sqrt{\lambda_i}$ (for any $\lambda_i > 0$) is 2, and occurs when $\lambda_i = 1$. It is clear that $\lambda_i = 1$ only when $X = Y$.

(iii) This is clear from (3.8).

(iv) See [Sra, 2011] for the proof.

(v) By direct substitution, we have

$$\begin{aligned} J_{\ell d}(AXB, AYB) &= \log \left| \frac{AXB + AYB}{2} \right| - \frac{1}{2} \log |AXB| - \frac{1}{2} \log |AYB| \\ &= \log \left| \frac{\mathcal{I} + (AXB)(AYB)^{-1}}{2} \right| - \frac{1}{2} \log \left| AXB(AYB)^{-1} \right| \text{ (refer (i) above.)} \\ &= \log \left| \frac{\mathcal{I} + XY^{-1}}{2} \right| - \frac{1}{2} \log \left| XY^{-1} \right| \\ &= J_{\ell d}(X, Y). \end{aligned}$$

(vi) The proof uses the same technique as in (v).

□

Theorem 3 (Non-Convexity)**.** *Assuming $X, Y > 0$, for a fixed Y, $J_{\ell d}(X, Y)$ is convex for $X \leq (1 + \sqrt{2})Y$ and concave for $X \geq (1 + \sqrt{2})Y$.*

Proof. Taking the second derivative of $J_{\ell d}(X, Y)$ with respect to X, we have

$$\nabla_X^2 J_{\ell d}(X, Y) = -(X + Y)^{-1} \otimes (X + Y)^{-1} + \frac{X^{-1} \otimes X^{-1}}{2}. \tag{3.9}$$

This expression is positive definite for $X \leq (1+\sqrt{2})Y$ and negative definite for $X \geq (1+\sqrt{2})Y$. □

Now, we will show some properties of JBLD useful for DTMRI applications. Note that we restrict our investigation to such properties of JBLD that use *nearness* computation of the matrices. One such useful quantity is the Anisotropic index of a given covariance with respect to a given similarity measure. We will understand this facet of JBLD in the next subsection.

3.2.2 Anisotropic Index

As we alluded to earlier, diffusion tensor imaging is the process of mapping diffusion of water molecules in the brain tissues and helps in the diagnosis of neurological disorders *non-invasively*. When the tissues have an internal fibrous structure, water molecules in these tissues will diffuse rapidly in directions aligned with this structure. Symmetric positive definite matrices are important mathematical objects in this field useful in the analysis of such diffusion patterns [Alexander et al., 2002]. *Anisotropic Index* (AI) is a useful quantity that is often used in this area [Moakher and Batchelor, 2006], which is the distance of a given SPD matrix from its Nearest Isotropic Matrix (NIM). When the diffusion of the water molecules is almost the same in all directions, AI will be close to zero, while for diffusions in specific directions, we will have a large AI value. Mathematically, the NIM $\alpha\mathcal{I}$ ($\alpha > 0$) from a given tensor $P > 0$ with respect to a distance measure $\mathcal{D}(.,.)$ is defined as:

$$\min_{\alpha>0} \mathcal{D}(\alpha\mathcal{I}, P) \tag{3.10}$$

There are closed form expressions for α when $\mathcal{D}$ is AIRM, LERM, or KLDM (see [Moakher and Batchelor, 2006] for details). Unfortunately, for $J_{\ell d}$ there is no closed form for this in general. In the following, we investigate this property of JBLD and propose a few theoretical properties.

Theorem 4. *Suppose $P \in \mathcal{S}^d_{++}$ and let $S = \alpha\mathcal{I}$ be such that $J_{\ell d}(P,S)$ is convex (see Theorem 3). Then the NIM to P is the minimum positive root of the following polynomial equation:*

$$\begin{aligned} p(\alpha) :=& d\alpha^d + (d-2)\sum_i \lambda_i \alpha^{d-1} + (d-4)\sum_{i,j,i\neq j} \lambda_i\lambda_j\alpha^{d-2} \\ &+ \cdots + (2-d)\sum_i \prod_{i\neq j} \lambda_j \alpha - d\prod_i \lambda_i = 0, \end{aligned} \tag{3.11}$$

where $\lambda_i, i = 1, 2, \cdots, d$ are the eigenvalues of P.

Proof. Using the definition of $J_{\ell d}$ in (3.10), and applying the assumption that $J_{\ell d}$ is convex, at optimality we have $\frac{\partial J_{\ell d}(\alpha\mathcal{I}, P)}{\partial \alpha} = 0$. This leads to:

$$\frac{1}{\alpha} = \frac{2}{d} \sum_{i=1}^{d} \frac{1}{\alpha + \lambda_i}.$$

Rearranging the terms, we have the polynomial equation in 3.11. Since the coefficient of α^{d-1} is always positive (for $d > 2$), there must always exist at least one positive root. □

Corollary 5. *When $d = 2$, we have $\alpha = \sqrt{\det P}$, which is the same as NIM for the Riemannian distance.*

DTMRI generally uses 3×3 SPD matrices. Since $p(\alpha)$ is a cubic in this case, one can obtain its roots in closed form. Nevertheless, we show below a useful property that helps bracket its positive root.

Lemma 6. *Let $P \in \mathcal{S}^d_{++}$ and suppose $\|P\|_2 < 1$, then*

$$\frac{1 + \operatorname{Tr}(P)/d}{1 + \operatorname{Tr}(P^{-1})/d} > \det P. \tag{3.12}$$

Proof. Suppose $P \in \mathcal{S}^d_{++}$ and $\|P\|_2 < 1$, then $\operatorname{Tr}(P) < d$. Suppose $\lambda_i, i = 1, 2, \cdots, d$ represents the eigenvalues of P, we have the following to prove from the lemma:

$$\frac{d + \operatorname{Tr}(P)}{d \det P + \sum_i \prod_{j \neq i} \lambda_i \lambda_j} > 1 \tag{3.13}$$

Since $\det P < \operatorname{Tr}(P)/d$ (due to AM-GM inequality) and since $\sum_i \prod_{j \neq i} \lambda_i \lambda_j < d$, we have the desired result. □

Theorem 7. *Let $P \in \mathcal{S}^3_{++}$, and if $S = \alpha\mathcal{I}, \alpha > 0$ is the NIM to P, then $\alpha \in (0, 1)$.*

Proof. Substituting $d = 3$ in (3.11), we have the following third degree polynomial equation:

$$p(\alpha) := 3\alpha^3 + \operatorname{Tr}(P)\alpha^2 - \det P \operatorname{Tr}(P^{-1})\alpha - 3 \det P = 0 \tag{3.14}$$

Analyzing the coefficients of $p(\alpha)$ shows that only one root is positive. Now, we have $p(0) < 0$. Applying Lemma 6, we have $p(1) > 0$, which concludes that the smallest positive root lies in $(0, 1)$. □

3.2.3 Connections to Other Metrics

We summarize below some of the interesting connections $J_{\ell d}$ has with the standard metrics on covariances.

Theorem 8 (Relations)**.** *For $X, Y \in \mathcal{S}_{++}^d$,*

$$\begin{aligned} (i) \quad & J_{\ell d}(X,Y) \leq D_R^2(X,Y) \\ (ii) \quad & J_{\ell d}(X,Y) \leq D_{kl}^2(X,Y) \end{aligned}$$

Proof. Let $v_i = \lambda_i(XY^{-1})$. Since $X, Y \in \mathcal{S}_{++}^d$, the eigenvalues v_i are also positive, whereby we can write each $v_i = e^{u_i}$ for some $u_i \in \mathbb{R}$. Using this notation, the AIRM may be rewritten as $D_R(X,Y) = \|u\|_2$, and the JBLD as

$$J_{\ell d}(X,Y) = \sum\nolimits_{i=1}^{d} (\log(1+e^{u_i}) - u_i/2 - \log 2), \tag{3.15}$$

where the equation follows by observing that $J_{\ell d}(X,Y) = \log|I + XY^{-1}| - \frac{1}{2}\log|XY^{-1}| - \log 2^d$.

To prove inequality (i), consider the function $f(u) = u^2 - \log(1+e^u) + u/2 + \log 2$. This function is convex since its second derivative

$$f''(u) = 2 - \frac{e^u}{(1+e^u)^2},$$

is clearly nonnegative. Moreover, f attains its minimum at $u^* = 0$, as is immediately seen by solving the optimality condition $f'(u) = 2u - e^u/(1+e^u) + 1/2 = 0$. Thus, $f(u) \geq f(u^*) = 0$ for all $u \in \mathbb{R}$, which in turn implies that

$$\sum\nolimits_{i=1}^{d} f(u_i) = D_R^2(X,Y) - J_{\ell d}(X,Y) \geq 0. \tag{3.16}$$

Similarly to prove inequality (ii), consider the function $g(u) = D_{kl}^2 - J_{\ell d}$, which expands to:

$$g(u) = \frac{1}{2}(e^u + \frac{1}{e^u}) - \log(1+e^u) + \frac{u}{2} + \log 2 - 1 \tag{3.17}$$

Going by the same steps as before, it is straight-forward to show that $g(u)$ is convex and attains its minimum when $u = 0$, proving the inequality. □

Theorem 9 (upper bound)**.** *If* $0 \prec mI \preceq X, Y \preceq MI$*, then*

$$D_R^2(X,Y) \leq 2\log(M/m)(J_{\ell d}(X,Y) + \gamma), \tag{3.18}$$

where $\gamma = d\log 2$*.*

Proof. Observe that

$$\sum\nolimits_{i=1}^{d}(\log(1+e^{u_i}) - u_i/2 - \log 2) \geq \sum\nolimits_{i=1}^{d}(|u_i|/2 - \log 2),$$

which implies the bound

$$J_{\ell d}(X,Y) + d\log 2 \geq \frac{1}{2}\|u\|_1. \tag{3.19}$$

Since $u^T u \leq \|u\|_\infty \|u\|_1$ (Hölder's inequality), using (3.19) we immediately obtain the bound

$$D_R^2(X,Y) = \|u\|_2^2 \leq 2\|u\|_\infty(J_{\ell d} + \gamma), \tag{3.20}$$

where $\gamma = d\log 2$. But $mI \preceq X, Y \preceq MI$ implies that $\|u\|_\infty \leq \log(M/m)$, which concludes the proof. □

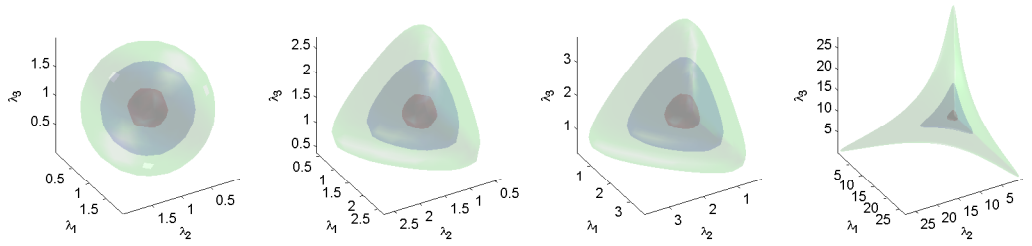
Our next result touches upon a condition when $J_{\ell d}(X,Y) < D_{le}^2(X,Y)$. A more general treatment of this relationship is outside the scope of this thesis, mainly because the Hessian of D_{le} does not have closed forms.

Theorem 10. *If* $X, Y \in \mathcal{S}_{++}^d$ *commute, then* $J_{\ell d}(X,Y) \leq D_{le}^2(X,Y)$*.*

Proof. We use the fact that when X, Y commute, $D_{le}(X,Y) = D_R(X,Y)$ (See Proposition 1). Now, using the connection between AIRM and JBLD (refer Theorem 8), we have the result. □

3.2.4 JBLD Geometry

In Figure 3.1, we plot the three dimensional balls (isosurfaces) associated with JBLD for various radii (0.1, 0.5 and 1) and centered at the identity tensor. We also compare the JBLD ball with the isosurfaces of Frobenius distance, AIRM and KLDM. As is expected Frobenius distance is isotropic and thus its balls are spherical, while AIRM and KLDM induce convex balls. Against these plots, and as was pointed by Theorem 3, the isosurfaces of JBLD are convex in some range while become concave as the radius goes large.



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

Figure 3.1: Isosurface plots for various distance measures. First, distances for arbitrary three dimensional covariances from the identity matrix are computed, and later isosurfaces corresponding to fixed distances of 0.1, 0.5 and 1 are plotted. The plots show the surfaces for: (a) Frobenius distance, (b) AIRM, (c) KLDM, and (d) JBLD respectively.

3.2.5 Computational Advantages

The greatest advantage of $J_{\ell d}$ against the Riemannian metric is its computational speed: $J_{\ell d}$ requires only computation of determinants, which can be done rapidly via 3 Cholesky factorizations (for $X + Y$, X and Y), each at a cost of $(1/3)d^3$ flops [Golub and Van Loan, 1996]. Computing D_R on the other hand requires generalized eigenvalues, which can be done for positive-definite matrices in approximately $4d^3$ flops. Thus, in general $J_{\ell d}$ is much faster (see also Table 3.1). The computational advantages of $J_{\ell d}$ are much more impressive when comparing evaluation of gradients[1] . Table 3.2 shows that computing $\nabla J_{\ell d}$ can be even more than 100 times faster than ∇D_R. This speed proves critical for NN retrieval, or more generally when using any algorithm that depends on gradients of the similarity measure, e.g., see [Fillard et al., 2005] and the references therein. Table 3.3 provides a summary of the various metrics, their gradients and computational complexities.

3.2.6 K-means with $J_{\ell d}$

Clustering using a measure is an important component fundamental to several algorithms in machine learning and computer vision. In fact, clustering is a prerequisite for metric tree based NN retrieval: a topic that we will be dealing with in the next section. In this section, we derive

[1] From a technical point, $J_{\ell d}$ computation for matrices over $d = 13$ was seen faster when the determinants were computed using the Cholesky decomposition.

d	D_R	$J_{\ell d}$
5	$.025 \pm .012$	$.030 \pm .007$
10	$.036 \pm .005$	$.040 \pm .009$
15	$.061 \pm .002$	$.050 \pm .004$
20	$.085 \pm .006$	$.061 \pm .009$
40	$.270 \pm .332$	$.123 \pm .012$
80	$1.23 \pm .055$	$.393 \pm .050$
200	$8.198 \pm .129$	$2.223 \pm .169$
500	$77.311 \pm .568$	22.186 ± 1.223
1000	492.743 ± 15.519	119.709 ± 1.416

Table 3.1: Average times (millisecs/trial) to compute function values; computed over 10,000 trials to reduce variance.

d	$\nabla_X D_R^2(X,Y)$	$\nabla_X J_{\ell d}(X,Y)$
5	$0.798 \pm .093$	$.036 \pm .009$
10	$2.383 \pm .209$	$.058 \pm .021$
20	$7.493 \pm .595$	$.110 \pm .013$
40	24.899 ± 1.126	$.270 \pm .047$
80	99.486 ± 5.181	$.921 \pm .028$
200	698.873 ± 39.602	8.767 ± 2.137
500	6377.742 ± 379.173	94.837 ± 1.195
1000	40443.059 ± 2827.048	622.289 ± 37.728

Table 3.2: Average times (millisecs/trial) to compute gradients; computed over 1000 trials (except for the last two experiments, where to save time only 100 trials were used) to reduce variance.

a K-means clustering algorithm based on $J_{\ell d}$. Let $S_1, S_2, \cdots, S_n$ be the input covariances that we need to be clustered. A standard K-means algorithm gives rise to the following optimization problem:

$$\min_{C_1, C_2, \cdots, C_K} \sum_{k=1}^{K} \sum_{S \in C_k} J_{\ell d}(X_k, S), \tag{3.21}$$

where X_k is the *centroid* of cluster C_k. Following the traditional K-means algorithm, we can alternate between the centroid computation and the clustering stages to minimize (3.21). The only significant step then amounting to the computation of the centroid for the kth cluster, which

Measure	$D^2(X,Y)$	$FLOPS$	Gradient (∇_{X})
AIRM	$\|\log(X^{-1/2}YX^{-1/2})\|_{\mathrm{F}}^2$	$4d^3$	$X^{-1}\log(XY^{-1})$
LERM	$\|\log(X)-\log(Y)\|_{\mathrm{F}}^2$	$\frac{8}{3}d^3$	$2X^{-1}\left(D_{\mathrm{X}}\log X-\log Y\right)$
KLDM	$\frac{1}{2}\operatorname{Tr}\left(X^{-1}Y+Y^{-1}X-2\mathcal{I}\right)$	$\frac{8}{3}d^3$	$Y^{-1}-X^{-1}YX^{-1}$
JBLD	$\log\left\|\frac{X+Y}{2}\right\|-\frac{1}{2}\log\|XY\|$	d^3	$(X+Y)^{-1}-\frac{1}{2}X^{-1}$

Table 3.3: A comparison of various metrics on covariances and their complexity against $J_{\ell d}$.

can be written as:

$$F := \min_{X_k} \sum_{S\in C_k} J_{\ell d}(X_k, S) \tag{3.22}$$

$$:= \min_{X_k} \sum_{S\in C_k} \log|\frac{X_k+S}{2}| - \frac{1}{2}\log|X_kS| \tag{3.23}$$

Unfortunately, as we saw earlier, $J_{\ell d}$ is neither a Bregman divergence, nor is it convex and thus we cannot use the traditional centroid computation. The good news is that, we can write (3.23) as the sum of a convex function $F_{vex}(X_k,S) = -\sum_{S\in C_k}\frac{|C_k|}{2}\log|X_k|$ and a concave term $F_{cave}(X_k,S)=\sum_{S\in C_k}\log|\frac{X_k+S}{2}|$. Such a combination of convex and concave objectives can be efficiently solved using Majorization-Minimization through the Convex-ConCave Procedure (CCCP)[Yuille and Rangarajan, 2003]. The main idea of this procedure is to approximate the concave part of the objective by its first order Taylor approximation around the current best estimate X_k^t; that is, for the $(t+1)$st step:

$$X_k^{t+1} = \operatorname*{argmin}_{X_k} F_{vex}(X_k,S) - X_k^T\nabla_{X_k}F_{cave}(X_k^t,S). \tag{3.24}$$

Substituting (3.24) in (3.23), later taking the gradient of (3.23) with respect to X_k and setting it to zero (recall that now we have a convex approximation to (3.23)), we have:

$$\sum_{S\in C_k}\nabla_{X_k}F_{vex}(X_k^{t+1},S) = -\sum_{S\in C_k}\nabla_{X_k}F_{cave}(X_k^t,S). \tag{3.25}$$

Expanding the gradient terms for $J_{\ell d}$, we have the following *fixed-point* iteration:

$$X_k^{t+1} = \left[\frac{1}{|C_k|}\sum_{S\in C_k}\left(\frac{S+X_k^t}{2}\right)^{-1}\right]^{-1}. \tag{3.26}$$

We now derive conditions guaranteeing the convergence of the fixed point iteration in (3.26). The uniqueness of a centroid thus found is reported in [Nielsen and Boltz, 2011].

Lemma 11. *The function $f(X) = X^{-1}$ for $X \in \mathcal{S}^d_{++}$ is matrix convex, i.e., for $X, Y \in \mathcal{S}^d_{++}$ and for $t \in [0, 1]$,*

$$f(tX + (1-t)Y) \leq tf(X) + (1-t)f(Y). \tag{3.27}$$

Proof. See Exercise V.1.15 [Bhatia, 1997] for details. □

Lemma 12. *If $X, Y \in \mathcal{S}^d_{++}$ and suppose $X \geq Y$, then $X^{-1} \leq Y^{-1}$.*

Proof. See Corollary 7.7.4 [Horn and Johnson, 1990]. □

Theorem 13. *Let $S_1, S_2, \cdots, S_n$ be the input covariances and let X^* be the centroid returned found by* (3.26)*. Then X^* lies in the* compact *interval*

$$\left(\frac{1}{n}\sum_{i=1}^{n} S_i^{-1}\right)^{-1} \leq X^* \leq \frac{1}{n}\sum_{i=1}^{n} S_i \tag{3.28}$$

Proof. Proving the left inequality: Applying Lemma 11 to (3.26), we have:

$$X^{-1} \leq \frac{1}{n}\sum_{i=1}^{n}\left(\frac{S_i^{-1} + X^{-1}}{2}\right) \tag{3.29}$$

$$\leq \frac{1}{n}\sum_{i=1}^{n}\frac{S_i^{-1}}{2} + \frac{1}{2}X^{-1}. \tag{3.30}$$

Now, applying Lemma 12, the result follows.

Proving the right inequality: As one can see, the right side of (3.26) is essentially the harmonic mean of $\frac{X+S_i}{2}$ for $i = 1, 2, \cdots, n$. Since the harmonic mean is always less than or equal to the arithmetic mean [Lawson and Lim, 2001], we have the result. □

Theorem 14. *Let $\left\{X^t\right\}$ (for $t \geq 1$) be the sequence of successive iterates generated as per* (3.26)*. Then, $X^t \to X^*$, where X^* is a stationary point of* (3.23)*.*

Proof. It is clear that F_{vex} and $-F_{cave}$ are strictly convex functions and $-\nabla F_{cave}$ is continuous. Further, from Theorem 13 it is clear that the solution lies in a compact interval inside $\mathcal{S}^d_{++}$. Thus, following the conditions of convergence stipulated in [Sriperumbudur and Lanckriet, 2009] (CCCP-II, Theorem 8), the iterations in (3.26) converges for a suitable initialization inside the compact set. □

3.3 Fast Nearest Neighbor Retrieval using JBLD

Now we turn to the key application that originally motivated us to investigate $J_{\ell d}$: Nearest Neighbor (NN) Retrieval for covariance matrices. Here, we have a dataset $\{S_1, \ldots, S_n\}$ of $d \times d$ covariance matrices that we must organize into a data structure to facilitate rapid NN retrieval. Towards this end, we chose to use the metric tree data structure (recall that the square-root of JBLD is satisfies the triangle-inequality) as we wanted to show the performance on an exact NN algorithm for covariances and for which approximations can be easily effected for faster searches.

3.3.1 NN Using Metric Tree

Metric Trees (MT) [Samet, 2006] are one of the fundamental tree based algorithms for fast NN retrieval useful when the underlying similarity measure is a metric. NN using the MT involves two steps: (i) Building the tree, and (ii) Querying the tree. We discuss each of these steps in detail below.

Building MT

To build the MT, we perform top-down partitioning of the input space by recursively applying the JBLD K-Means algorithm (introduced above). Each partition of the MT is identified by a centroid and the ball radius. For n data points, the total build time of the tree is $O(n \log n)$. To save time, we stop partitioning a cluster when the number of points in it goes below a certain threshold; this threshold is selected as a balance between the computational time to do exhaustive search on the cluster elements against doing k-means on it.

Querying using MT

Given a query point q, one first performs a greedy binary search for the NN along the most proximal centroids at each level. Once a leaf partition is reached, exhaustive search is used to localize to the candidate centroid X_c. Then one backtracks to check if any of the sibling nodes (that were temporarily ignored in the greedy search) contain a data point that is closer to q than X_c. To this end, we solve the following optimization problem on each of the sibling centroids

C:

$$d(X_c, q) > \min_{X; d(X,C)=R} d(X, q) \tag{3.31}$$

where X is called the projection of q onto the ball with centroid C, radius R and d is some distance function. If (3.31) is satisfied, then the sibling node should be explored, otherwise it can be pruned. When d is a metric, (3.31) has a simple solution utilizing the triangle inequality as is described in [Brin, 1995]. The mechanism can be extended to retrieve k-NN by repeating the search ignoring the (k-1) NNs already retrieved. This can be efficiently implemented by maintaining a priority queue of potential sub-tree centroids and worst case distances of the query to any candidate node in this subtree, as described in [Ciaccia et al., 1997].

3.3.2 NN retrieval via JBLD Approximation

Recall that in LERM, we embed the covariances in the Euclidean space by taking the matrix logarithm followed by their vectorization. Such an offline process leads to simple and fast Euclidean comparisons of the covariances, even though we sacrifice some accuracy in NN retrieval. In this section, we will propose such an approximation to JBLD, which we call *Approximate JBLD* (AJBLD) that allows offline computations. AJBLD is more efficient in storage compared to LERM, and at the same time potentially faster. In the following, we will review the necessary theory that will help understand AJBLD.

Suppose S is any matrix, then we will use the notation $\lambda^{\downarrow}(S)$ to denote the diagonal matrix with the diagonal as the eigenvalues of S arranged in the descending order. Similarly $\lambda^{\uparrow}(S)$ will represent the diagonal matrix with eigenvalues in the ascending order.

Theorem 15. *Let $X, Y \in \mathcal{S}^d_{++}$, then*

$$\det\left(\lambda^{\downarrow}(X) + \lambda^{\downarrow}(Y)\right) \leq \det\left(X + Y\right) \leq \det\left(\lambda^{\downarrow}(X) + \lambda^{\uparrow}(Y)\right) \tag{3.32}$$

Proof. See Exercise VI.7.2 [Bhatia, 1997]. □

Theorem 15 leads to the following bounds on $J_{\ell d}$:

Corollary 16. *For $X, Y \in \mathcal{S}^d_{++}$, then*

$$J_{\ell d}(\lambda^{\downarrow}(X) + \lambda^{\downarrow}(Y)) \leq J_{\ell d}(X + Y) \leq J_{\ell d}(\lambda^{\downarrow}(X) + \lambda^{\uparrow}(Y)). \tag{3.33}$$

Using Corollary 16, we propose the following approximation to $J_{\ell d}$.

Definition 6. *Let $X, Y \in \mathcal{S}_{++}^d$, and let $\lambda_i^{\downarrow}(Z)$ represents the ith eigenvalue of Z in a sorted list of eigenvalues of Z in the descending order, then we define Approximate JBLD ($J_{a\ell d}$) as:*

$$J_{a\ell d}(X,Y) = \sum_{i=1}^{d} \left\{ \log\left(\lambda_i^{\downarrow}(X) + \lambda_i^{\downarrow}(Y)\right) - \frac{1}{2}\log\left(\lambda_i^{\downarrow}(X)\lambda_i^{\downarrow}(Y)\right) \right\}. \tag{3.34}$$

As is clear from Definition 6, we do not need to store the entire matrices for computing the $J_{a\ell d}$, instead we just need the eigenvalues in the sorted order. This will reduce the storage expenses of the matrices from $\mathcal{O}(d^2)$ to $\mathcal{O}(d)$. Further, compared to LERM, the computational expense is lowered for similar reasons. Since we are now dealing with a lower bound to $J_{\ell d}$, we might need to sacrifice accuracy with this approximation and thus $J_{a\ell d}$ is useful in cases where storage is at a premium, while lower accuracy is permitted.

3.4 Experiments

We are now ready to describe our experimental setup and results to substantiate the effectiveness of $J_{\ell d}$. We first discuss the performance metric on which our experiments are based, later providing simulation results exposing various aspects of our metric, followed by the results on four real-world datasets. All algorithms were implemented in MATLAB and tested on a machine with 3GHz single core CPU and 4GB RAM.

3.4.1 Performance Metric

Accuracy@K: Suppose we have a covariance dataset $\mathcal{D}$ and a query set $\mathcal{Q}$. Accuracy@K describes the average accuracy when retrieving K nearest covariances from $\mathcal{D}$ for each item in $\mathcal{Q}$. Suppose G_q^K stands for the ground truth labels associated with the qth query, and if M_q^K denotes the set of K nearest covariances found using a metric M, then we formally define:

$$Accuracy@K = \frac{1}{|Q|}\sum_{q \in Q} \frac{|G_q^K \cap M_q^K|}{|G_q^K|}. \tag{3.35}$$

Note that *Accuracy* is technically equivalent to both the standard measures of *precision* and *recall* in our case. Most often we work with $K = 1$, in which case we will drop the suffix and will refer as *Accuracy*. Since some of the datasets used in our experiments do not have ground truth data available, the baselines for comparison were decided via a linear scan using the AIRM metric as this metric is deemed the state-of-the-art on covariance data.

3.4.2 Simulations

Before we delve into the details of our experiments, we highlight here the base experimental configurations that we used for all the simulation experiments. Since there are a variety of code optimizations and offline computations possible for the various metrics, we decided to test all the algorithms with the base implementation as provided by MATLAB. An exception here are the experiments using LERM. It was found that computing LERM projecting the input matrices into the log-Euclidean space (through matrix logarithms) resulted in expensive computations, as a result of which the performances were incomparable with the setup used for other metrics. Thus, before using this metric, we took the logarithm of all the covariances offline.

For the NN experiments, we used a metric tree with four branches and allowed a maximum of 100 data points at the leaf nodes. With regard to computing the cluster centroids (for k-means), LERM and FROB metrics used the ordinary Euclidean sample mean, while AIRM used the Frechet mean using the iterative approximation algorithm described in [Bini and Iannazzo, 2011]. The centroid for KLDM boils down to computing the solution of a Riccati equation as described in [Myrvoll and Soong, 2003]. For the simulation experiments, we used the results produced by AIRM as the ground truth.

Now we are ready to describe our base configuration for the various simulation experiments. We used 1K covariances of 10D with 50 true number of clusters as the dataset and a collection of 100 covariances as the query set. The plots that we are about to show resulted from average performances by repeating the experiments 100 times using different database and query sets. Next, we consider the various experiments and present the results.

3.4.3 Accuracy Against Noise

Stability of a given metric towards perturbations in the data is important in a variety of applications; e.g., people tracking. Given that the metrics on covariances are nonlinear, the primary goal of this experiment is to validate the robustness of JBLD against noise in the covariance descriptors for the task of NN retrieval. Towards this end, we created a base set of 1K covariances from a set of simulated feature vectors (we in fact used a set of random vectors for the simulation). Subsequently, Gaussian noise of varying magnitude (relative to the signal strength) was added to the feature vectors to obtain a set of 100 noisy covariances (our basic idea here was to simulate an appearance tracking session, in which the actual feature vectors from the silhouette

of the appearance are fixed, while subsequent tracked appearances are noise corrupted versions of these feature vectors. Covariances are then generated from these noisy feature vectors that are then used for tracking). The base covariances were used as queries while the noisy ones as the database. A linear scan through the data using the Riemannian metric (AIRM) to measure nearness *defined* the ground truth. Figure 3.2 shows the average accuracy values for decreasing SNR for three different covariance dimensions (10D, 20D and 40D). It is clear that JBLD is more robust than LERM and KLDM, at the same time yields accuracy almost close to the baseline Riemannian metric, irrespective of the dimension of the matrix. It is to be noted that a retrieval using the Frobenius distance (FROB) is clearly seen to perform poorly. We would also like to highlight that we noticed a small drop in the accuracy of KLDM (as seen in Figure 3.2(c)) as the data dimensionality increases, which we suspect is due to the poor conditioning of the data matrices as the dimensionality grows, impacting the matrix inversions.

3.4.4 Effect of Cluster Size

This section analyzes the scalability of $J_{\ell d}$ to an increasing number of true data clusters (given fixed database size). The basic goal of this experiment is to expose the clustering performance of our $J_{\ell d}$-K-means algorithm against the K-means based on other metrics. The performance comparison is analyzed on three aspects: (i) the average accuracy of NN retrieval, (ii) average metric tree creation time (which includes K-means clustering for each internal node of the metric tree), and (iii) average search time using a metric tree. Figure 3.3 shows results from this experiment. There are a few very important properties of the metrics that are revealed by these plots: (i) the accuracy of $J_{\ell d}$ matches perfectly with that of AIRM (note that AIRM is used as the ground truth), (ii) assuming the metric tree is constructed optimally, the search time for AIRM and $J_{\ell d}$ are comparable, and (iii) (which is the most important) the metric tree construction for AIRM almost increases quadratically with increasing number of true clusters, while that for other metrics is more favorable. Together, the three plots substantiate the superior performance of $J_{\ell d}$. Later in this chapter, we will get back to proving these claims on real-data.

3.4.5 Effect of Matrix Dimension

One of the major motivations for proposing $J_{\ell d}$ as a replacement for existing metrics on covariances is its scalability to increasing matrix dimensions. Figure 3.4 shows the results of

accuracy, metric tree creation time and search time using a metric tree. As is clear from the plots, the metric tree creation time increases at many orders of magnitude worser with AIRM than with other metrics, while $J_{\ell d}$ performs better at accuracy and retrieval time against other metrics. Similar to what we noticed in Figure 3.2, the accuracy of KLDM worsens as the matrix dimension increases.

3.4.6 Effect of Increasing Database Size

This experiment shows the performance of $J_{\ell d}$ against searching in larger datasets. Towards this end, we kept the number of true clusters constant and same as in other experiments, but increased the number of data points (covariances) associated with each cluster. The results of this experiment in terms of accuracy, tree buildup time and retrieval performance is shown in Figure 3.5. Similar to the previous plots, it is clear that $J_{\ell d}$ provides promising results in all the three properties, while maintaining nearly perfect retrieval accuracy, showing that it does not get distracted from the nearest neighbor even when the datasize increases.

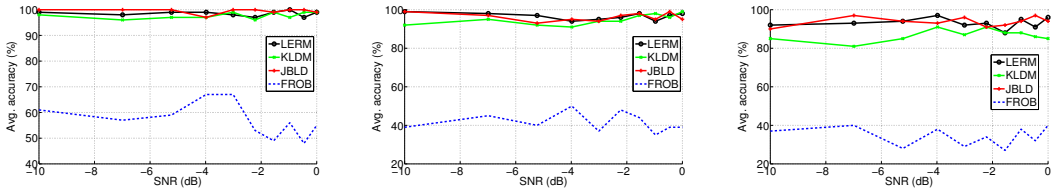
3.4.7 Anisotropic Index

As we alluded to briefly in Subsection 3.2.2, Anisotropic Index (AI) of a matrix over a divergence (or metric) is a useful quantity in several DTMRI applications [Moakher and Batchelor, 2006]. Figure 3.6 plots the average Fractional Anisotropic Index[2] $(\mathrm{FAI}(X) = \mathrm{AI}(X)/(1 + \mathrm{AI}(X)))$ for the various distances on 3×3 SPD tensors for increasing levels of tensor anisotropy (increasing condition number). As is clear from the plots, JBLD was found to have lower anisotropy when compared to other metrics on covariances.

3.4.8 Real Data Experiments

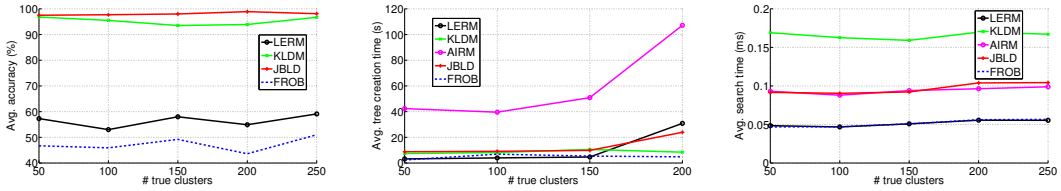
Continuing upon the simulated performance figures of $J_{\ell d}$ against other metrics, this subsection provides results on real-data. First, we will showcase a few qualitative results from some important applications of covariances from literature.

[2] FAI is in the range of $[0, 1]$ irrespective of the underlying divergence and thus provides a fair comparison instead of just using AI.



(a) (b) (c)

Figure 3.2: Accuracy against increasing noise for various matrix dimensions n; (a) $n = 10 \times 10$, (b) $n = 20 \times 20$, (c) $n = 40 \times 40$. It is assumed that the AIRM is the ground truth. MFD stands for the Matrix Frobenius Distance.



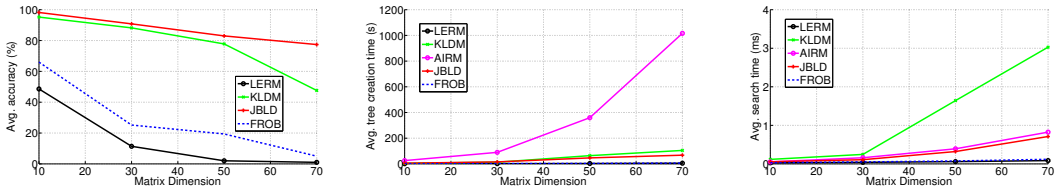
(a) (b) (c)

Figure 3.3: Fixed dataset size of 1K, query size of 100 and for increasing number of true clusters: 3.3(a) accuracy of search, 3.3(b) time to create the metric tree, and 3.3(c) speed of retrieval using the metric tree. The average is computed over 100 trials.

3.4.9 Tracking Using Integral Images

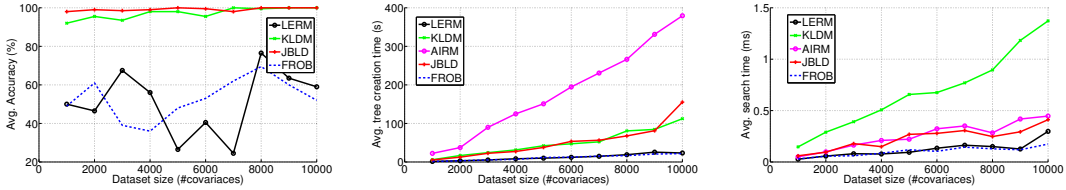
People appearance tracking has been one of the most successful applications using covariances. We chose to experiment with some of the popular tracking scenarios: (i) face tracking under affine transformations, (ii) face tracking under changes in pose, and (iii) vehicle tracking. For (i) and (ii), the tracking dataset described in [Maggio et al., 2007] was used, while the vehicle tracking video was taken from the ViSOR repository[3]. The images from the video were resized to 244×320 for speed and integral images computed on each frame. An input tracking region was given at the beginning of the video, which is then tracked in subsequent images using the integral transform, later computing covariances from the features in this region. We used the color and the first order gradient features for the covariances. Figures 3.7(a),3.7(b),3.8(a), and 3.8(b) show qualitative results from these experiments. We compared the window of tracking for both AIRM and JBLD, and found that they always fall at the same location in the video

[3] http://www.openvisor.org



(a) (b) (c)

Figure 3.4: Fixed dataset size of 1K, query size of 100 and for increasing covariance matrix dimensions: 3.4(a) accuracy of search, 3.4(b) time to create the metric tree, and 3.4(c) speed of retrieval using the metric tree. The average is computed over 100 trials.



(a) (b) (c)

Figure 3.5: Fixed number of true number clusters, query size of 100 and but increasing the covariance dataset size: 3.5(a) accuracy of search, 3.5(b) time to create the metric tree, and 3.5(c) speed of retrieval using the metric tree. The average is computed over 100 trials.

(and hence not shown).

3.4.10 Texture Segmentation

Another important application of covariances has been in texture segmentation [O. Tuzel, F.Porikli, and P. Meer, 2006] which has further application in DTMRI, background subtraction [Caseiro et al., 2010], etc. In Figure 3.4.11, we present a few qualitative results from segmentation on the Brodatz texture dataset. Each of the images were a combination of two different textures, the objective being to find the boundary and separate the classes. We first transformed the given texture image into a tensor image, where each pixel was replaced by a covariance matrix computed using all the pixels in a $p \times p$ patch around the given pixel. The 5×5 covariances were computed using features such as image coordinates of the pixels in this patch, image intensity at each pixel, and first order moments. Next, we applied the JBLD K-means algorithm for the texture mixture, later segregated the patches using their cluster labels.

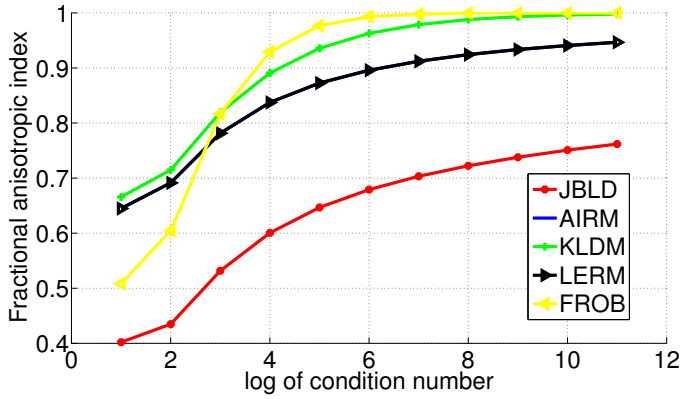


Figure 3.6: Plots of fractional anisotropic indices of various distance measures on 3×3 SPD tensors for increasing condition numbers (increasing anisotropy) of the constituent tensors. The FAI for AIRM and LERM were found to be very close.

3.4.11 Real-Data NN Experiments

Now we are ready to present quantitative results on real-world datasets. For real-world experiments that are described in the subsequent sections, we use four different vision applications for which covariance descriptors have shown produce promising results: (i) texture recognition, (ii) action recognition, (iii) face recognition, and (iv) people appearance tracking. We briefly review below each of these datasets and how covariances were computed for each application. See Figure 3.10 for sample images from each dataset.

Texture Dataset: Texture recognition has been one of the oldest applications of covariances spanning a variety of domains, e.g., DTMRI, satellite imaging, etc. The texture dataset for our experiments was created by combining the 160 texture images in the Brodatz dataset and the 60 texture classes in the CURET dataset [K.Dana et al., 1999]. Each texture category in the Brodatz dataset consisted of one 512×512 image. To create the covariances from these images, we followed the suggestions in [O. Tuzel, F.Porikli, and P. Meer, 2006]. First patches of size 20×20 were sampled from random locations in each image, later using the image coordinate of each pixel in a patch, together with the image intensity, and the first order gradients to build 5D features. The covariance matrices computed such feature vectors on all the pixels inside the patch constituted one such data matrix and approximately 5K covariances from all the texture images in all the categories from the Brodatz dataset. To build a larger dataset for textures, we combined this dataset with texture covariances from the CURET dataset [K.Dana

et al., 1999] which consists of 60 texture catetgories, with each texture having varying degrees of illumination and pose variations. Using the RGB color information, together with the 5 features described before, we created approximately 27K covariances each of size 8×8. To have covariances of the same dimensionality across the two datasets, we appended a unit matrix of very small diagonal for the RGB to the covariances computed from the Brodatz dataset.

Action Recognition Dataset: Activity recognition via optical flow covariances is a very recent addition to the family of applications with covariance descriptors, and shows great promise. For every pair of frames in a given video, the optical flow is initially computed; the flow is then threshold and 12D feature vectors were extracted from each non-zero flow location (refer [Guo et al., 2010] for details on this feature vector). It is proposed that the covariance computed from the optical flow features captures the profile of that activity uniquely. To build the optical flow covariance dataset, we used a combination of activity videos from the Weizmann activity dataset [Gorelick et al., 2007], the KTH dataset[4], and the UT tower dataset [Chen et al., 2010]. This resulted in a large dataset of approximately 63.5K covariances each of dimension 12×12.

Face recognition: Face recognition is still a very active area of research in computer vision and there has been many effective ideas suggested. In [Pang et al., 2008], the idea of covariance matrices were extended for recognizing faces, where each face image was convolved with 40 Gabor filters, the outputs of which were then collated to form 40×40 covariances. Although the covariance descriptors are not the state-of-the-art in face recognition, our choice of this application for this chapter is to analyze the performance of our metric for real-data of large dimensions. Towards this end, we used the images from the *Faces in the Wild* dataset [Jain and Learned-Miller, 2010], which consists of approximately 31K face images mainly collected from newspapers. We used the same approach as in [Pang et al., 2008] for computing the covariances, along with incorporating the RGB color information of each pixel and the first and second order intensity gradients to form 48×48 covariances.

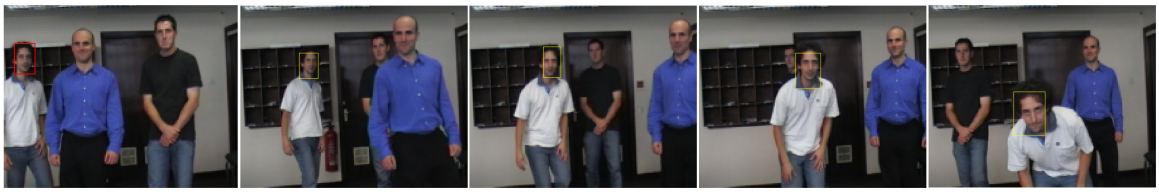
People Appearances: An important real-time application of covariances is people tracking from surveillance cameras [O. Tuzel, F.Porikli, and P. Meer, 2006]. To analyze the suitability of our metric for such applications, we illustrate empirical results on tracking data. For this experiment, we used videos of people appearances tracked using multiple cameras[5]. The background was first learned using a mixture of Gaussians, then the silhouettes of people were extracted.

[4] http://www.nada.kth.se/cvap/actions/

[5] http://cvlab.epfl.ch/research/body/surv/#data

The first and second order image gradients along with the color information were used to obtain approximately 10K covariances of size 8×8.

Ground Truth: Note that the texture dataset, the action dataset, and the faces dataset have ground truth labels associated with each data point and thus for accuracy comparisons, we directly use this class label of the query set against the class label associated with the NN found by a metric. Unfortunately, the people appearances dataset does not have a ground truth and thus we use the label of the NN found by AIRM as the ground truth.



(a)

(b)

Figure 3.7: Tracking using JBLD on covariances computed from integral images: (a) affine face tracking, (b) tracking face with pose variations.

3.4.12 NN via Exhaustive Search

Here we present our experiments and results on exhaustive search using the various metrics. Exhaustive search is important from a practical point of view as most real-time applications (such as tracking) cannot spend time in building a metric tree. In this section, we analyze the performance of JBLD in terms of accuracy and retrieval speed on each of the datasets we described in the previous section.

3.4.13 Accuracy

We divided each of the datasets into database and query sets, and then computed accuracy against either the available ground truth or the baseline computed using AIRM. The query set

(a)

(b)

Figure 3.8: Vehicle tracking results. The red rectangle in the first image in each row shows the object being tracked. The yellow rectangles in the subsequent images are the nearest objects returned by JBLD.

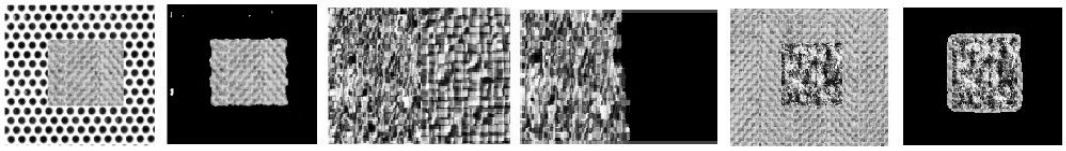
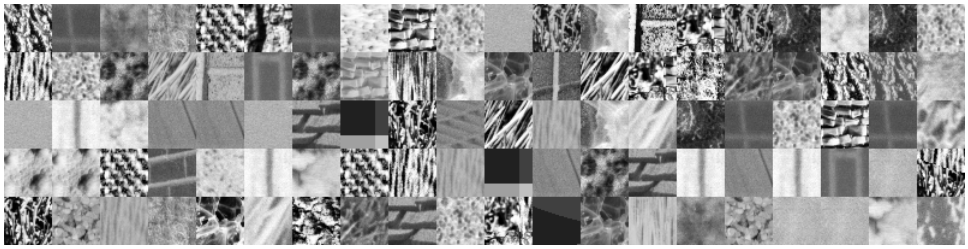


Figure 3.9: Illustration of qualitative results from multiple texture segmentations. The left image in each pair shows the original mixed texture image and the right image in each pair shows the output of segmentation, with one texture masked out.

typically consisted of 1K covariances. The results are shown in Table 3.4. Clearly, JBLD outperforms all the other metrics in accuracy, without compromising much on the speed of retrieval. In the case of LERM, we had to vectorize the covairances using the log-Euclidean projections for tractability of the application. The time taken for this operation for each of the datasets is also shown in the table. Since this embedding uses the eigen decomposition of the matrices, this operation is seen to be computationally expensive, deterring the suitability of LERM for real-time applications. We also compare the performance of JBLD against other distances such as the Cholesky (CHOL) distance and the Frobenius (FROB) distance. Frobenius distance was seen to perform poorly in all our experiments, although as expected, it is the fastest. The numerical results are averaged over 10 trials, each time using a different database and a query set.



(a)

(b)

(c)

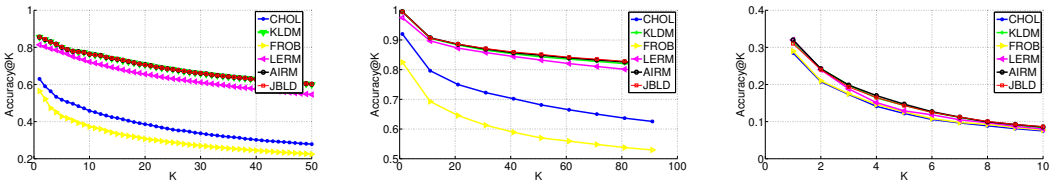
Figure 3.10: Sample images from: (a)Texture images from Brodatz dataset, (b) Faces in the Wild dataset, and (c) people appearance tracking dataset.

Dataset(size)	AIRM	JBLD	LERM	KLDM	CHOL	FROB
Texture (25852)						
Avg. Accuracy(%)	85.5	**85.5**	82.0	85.5	63.0	56.5
Avg. Time (s)	1.63	**1.50**	1.16 (4.21)	1.71	1.81	1.21
Activity(62425)						
Avg. Accuracy(%)	99.5	**99.5**	96.5	99.5	92.0	82.5
Avg. Time (s)	4.04	**3.71**	2.42 (10.24)	4.34	4.98	2.53
Faces Wild(29700)						
Avg. Accuracy(%)	32.5	**33.0**	30.5	31.5	29.5	26.5
Avg. Time (s)	10.26	**4.68**	2.44 (24.54)	10.33	12.13	2.13
Appearance (8596)						
Avg. Accuracy(%)	–	**100**	83.3	70.0	91.0	52.1
Avg. Time (s)	0.44	**0.40**	0.17 (1.7)	0.42	0.28	0.15

Table 3.4: Performance of JBLD on different datasets and against various other metrics for 1-NN query using exhaustive search averaged over 1K queries. Note that for the appearance dataset, we used AIRM as the baseline (and thus the accuracy not shown). Avg. time is in seconds for going over the entire dataset once to find the NN. The time taken for the offline log-Euclidean projections is shown in brackets under LERM.

3.4.14 K-Nearest Neighbor Retrieval

We take the previous experiments of 1-NN a step further and present results on K-NN retrieval for an increasing K. The idea is to generalize the power of 1-NN to a K-nn application. We plot in 3.11, the results of Accuracy@K, where the maximum value of K is determined by the cardinality of a ground truth class. The plots clearly show that JBLD performs well against almost all other metrics in terms of accuracy for increasing K.



(a) (b) (c)

Figure 3.11: Accuracy@K plots for (a) texture dataset, (b) activity dataset, and (c) faces dataset.

3.4.15 NN Performance Using Metric Tree

Building the Tree: The time required to build the NN data structure plays a critical role in the deployment of a measure. In Table 3.5, we show a comparison of the build time of the metric tree for each of the datasets, with comparisons of JBLD against AIRM. As is clear from the table, the performance of AIRM is poor and worsens with the increase in the matrix dimensions (see the face dataset). JBLD, on the other hand, takes far lesser time to initialize and shows consistent performance even against increasing dataset size and matrix dimensions.

Dataset (size)	AIRM	JBLD
Texture (25852)	769.96	**131.31**
Activity (62425)	2985.62	**746.67**
Faces (29700)	13776.30	**854.33**
People (8596)	213.41	**53.165**

Table 3.5: Comparison of metric tree buildup times (in seconds) for the various datasets.

3.4.16 NN Retrieval

Exact NN via a Metric Tree

Next, we compare the accuracy and the speed of retrieval of JBLD against the other metrics using the metric tree. For this experiment, we used a metric tree with four branches at each internal node and 1K leaf nodes, for all the datasets. Since K-means using AIRM was found to take too much time until it converged (it was found that with the face dataset with 48x48 covariances took more than 3 hours with approximately 26K covariances), we decided to stop the clustering process when there was less than 10% of data movements in the underlying Loyd's algorithm. This configuration was forced on K-means using other metrics as well for fairness of comparison of the results. We show in Table 3.6 the average results of 1-NN using the metric tree with 500 queries, and with averages computed over 10 trials, each time using a different sample set for the database and the query. As is clear from the table, JBLD provides accuracy equal to AIRM with at least 1.5 times speedup with the matrices of small size, while more over 7 times speedup for the face dataset. The retrieval speed of LERM and FROB is high, while the accuracy is low. KLDM was seen to provide accuracy similar to JBLD, but with low retrieval speed. In short, JBLD seems to provide the best mix of accuracy and computational expense.

Approximate NN via Metric Tree

It is well-known that the worst case computational complexity of metric tree is linear. Thus in Table 3.7, we also evaluate the performance of an approximate variant of metric tree based retrieval in which we limit the search for NNs while backtracking the metric tree to atmost n items, where in our experiments we used $n = 5$. This heuristic is in fact a variant of the well-known Best-Bin-First (BBF) [Beis and Lowe, 1997] method, the idea being to sacrifice the accuracy a little bit for a large speedup in retrieval. As is clear from the table, such a simple heuristic can provide a speedup of approximately 100 times that of the exact NN, while not much of a lose in the accuracy. Also, it is clear from the table that JBLD gives the best accuracy among other metrics with reasonable retrieval results.

3.4.17 Summary of Results

Here we summarize our findings about JBLD and the other metrics with regard to our experiments. As is clear from the above tables and plots, JBLD was seen to provide the best accuracy compared to other metrics, with accuracies sometimes even superseding that of the Riemannian metric. It might seem from Table 3.7 that the speed of retrieval of JBLD is close to that of AIRM; this result needs to be seen together with the results in Table 3.5 which shows that building a metric tree for AIRM is extremely challenging, especially when the data is large dimensional. KLDM was seen to perform almost equal to JBLD in accuracy in almost all of the experiments, while was often seen to take approximately twice as much computational time. LERM seemed superior in retrieval speed due to the capability of offline computations, while was seen to have lower accuracy. AJBLD although saves in storage was found to produce inferior results to other metrics. This is expected as it is a lower bound approximation to JBLD. From the results it is apparent that AJBLD is in general more accurate than the FROB distance, although takes slightly more computations. This we believe is an implementation issue as there are fast Euclidean distance computations in Matlab. Technically, ABJLD has fewer dimensions than FROB or LERM, and thus is expected to be computationally superior. Finally, FROB was found to perform the best in speed as would be expected, but has the lowest accuracy. In summary, JBLD is seen to provide the most consistent results among all the experiments, with the best accuracy, scalability and moderate retrieval speeds.

Before we conclude, we would like to comment on the performance of JBLD on the Faces in

the Wild dataset. It is clear that the recognition accuracy on this dataset is low using covariances. Actually, our motivation for this experiment was to show the performance of JBLD for NN retrieval on large covariances. Face recognition using 48×48 covariances showed state-of-the-art results on the FERRET dataset as described in [Pang et al., 2008]. In fact, our own paper [Cherian et al., 2011] showed better results (accuracy of 60.5%) than the ones shown here on the FERRET dataset. Unfortunately, the FERRET dataset is quite small (3010 data points) for evaluating NN retrieval and thus we decided to use the Faces in the Wild dataset. This dataset has more variations (mostly in the pose) in the face images, as a result the second order statistics, as computed by covariances might not directly capture the recognizing properties of the dataset (as is implied by the results). We believe a stage of preprocessing (so that the covariances computed are from relevant face parts) could be able to improve this accuracy. Nevertheless, we would like to point out here that we are not interested in improving the accuracy of face recognition, instead our intention is to compare the accuracy of JBLD against other metrics given the high dimensional covariance valued data. In this respect, we believe our scope is restricted to showing that the accuracy of JBLD is comparable to AIRM, but is much faster in computation, which our results substantiate.

Dataset	AIRM	JBLD	LERM	KLDM	FROB	AJBLD
Texture						
Acc. (%)	83.00	**83.00**	78.40	83.00	52.00	65.8
Time (ms)	953.4	**522.3**	396.3	1199.6	522.0	456.57
Activity						
Acc. (%)	98.8	**99.00**	95.80	98.60	85.60	94.4
Time (ms)	3634.0	**3273.8**	1631.9	4266.6	1614.92	2176.6
Faces						
Acc. (%)	26.6	**26.6**	22.8	26.1	20.6	18.0
Time (ms)	9756.1	**1585.1**	680.8	2617.7	658.6	921.4
People						
Acc. (%)	–	**100**	92.0	98.1	43.3	49.4
Time (ms)	354.3	**229.7**	214.2	701.1	163.7	246.37

Table 3.6: True NN using the metric tree. The results are averaged over 500 queries. Also refer to Table 3.5 for comparing the metric tree creation time.

Dataset	AIRM	JBLD	LERM	KLDM	FROB	AJBLD
Texture						
Acc. (%)	80.2	**81.40**	76.80	81.40	48.80	64.8
Time (ms)	34.28	**21.04**	18.18	52.98	17.73	15.4
Activity						
Acc. (%)	95.6	**96.20**	93.60	95.6	78.00	92.8
Time (ms)	38.1	**30.39**	20.3	85.9	12.2	20.4
Faces						
Acc. (%)	22.4	**24.2**	20.2	22.2	18.6	17.0
Time (ms)	26.16	**23.2**	20.6	55.7	16.6	18.58
People						
Acc.(%)	–	**91.3**	85.6	91.1	36.4	44.7
Time (ms)	4.81	**4.78**	3.31	8.12	3.07	3.56

Table 3.7: ANN performance using Best-Bin-First strategy using metric tree. The results are averaged over 500 queries. Also refer to Table 3.5 for comparing the metric tree creation time.

Chapter 4

Non-Parametric Covariance Clustering

As we saw in the last chapter, covariance matrices of feature vectors can be used as an efficient data descriptor for a variety of computer vision applications. This efficiency arises because of the ability of covariances to fuse multiple features into one compact representation, invariance to affine transformations and amenability to real-time computations. As a result these descriptors are by now used in several important computer vision applications such as people and object tracking, face recognition, action recognition, etc. In the last chapter, we dealt with the problem of nearest neighbor retrieval on covariance datasets organized as a metric tree, in which data clustering using K-means was an important intermediate step. Although our metric tree was hand-crafted to be an n-ary tree, it would have been more efficient if the number of potential data clusters was learned from the data itself. That is, instead of the user providing an estimate of the number of clusters in the data, we would like the algorithm to estimate the number of clusters in an unsupervised way. Among several unsupervised learning mechanisms used in machine learning, one idea is non-parameteric Bayesian learning, which is statistically well based, and its behaviors well-understood over the years. Unfortunately, there has been little work in using such non-parametric learning methods on the space of covariance matrices; a gap that we will fill in this chapter. Towards this end, we offer a novel Dirichlet process mixture model framework on symmetric positive definite matrices. Our experiments demonstrate that this framework is not only appealing in scalability, but also is computationally more efficient

against other soft clustering algorithms on the covariance manifold.

4.1 Introduction

We will motivate this section with an application of covariances for clustering people appearances for video surveillance. Clustering of object appearances from video cameras is a long studied problem in computer vision. Appearances of the same object can vary significantly from one camera to the next in a video surveillance application. This can be due to camera sensor noise, changes in scene illumination, presence of shadows, changes in the pose of the object, partial occlusion, etc. Figure 4.1 shows images from a real-world people appearance tracking session and delineates some of these difficulties.

Figure 4.1: Images from a multi-camera people tracking session. The rectangles drawn on the images mark the person being tracked.

As is clear from Figure 4.1, the tracked images can be very noisy and the appearance of the person being tracked can undergo considerable changes. Clustering of appearances in such a setting is a challenging problem. A survey of recent developments in this area can be seen in [Moeslund et al., 2006]. As we saw in the last chapter, covariance matrices of features extracted from the appearances produce significant robustness to the above problems [O. Tuzel, F.Porikli, and P. Meer, 2006]. These matrices provide a non-linear fusion of various image features and have been deployed as good feature descriptors; not only in people tracking, but also in many other areas of computer vision, including face recognition [Pang et al., 2008], and DTI imaging [Dryden et al., 2008].

A real-time algorithm to compute covariance matrices of object appearances using integral

images is proposed in [O. Tuzel, F.Porikli, and P. Meer, 2006]. This makes covariance descriptors an ideal platform for people tracking in video surveillance applications. On the negative side, from the last chapter, we know that covariances do not conform to the Euclidean geometry; but span the cone of SPD matrices and form a connected Riemannian manifold with an associated Riemannian metric [Forstner and Moonen, 1999]. The mean of a set of covariance matrices can be computed using the Karcher mean algorithm [Pennec et al., 2006], thus suggesting the viability of a K-means type clustering [O. Tuzel, F. Porikli, and P. Meer., 2006] scheme. In [Sivalingam et al., 2009], a semi-supervised learning framework is suggested by embedding these matrices into the Euclidean space by vectorizing the symmetric matrices created through a log-Euclidean mapping. An EM algorithm for clustering SPD matrices using a mixture of Wishart distributions is suggested in [Hidot and Saint-Jean, 2010]. Since the number of clusters in the model needs to be specified in any soft-clustering algorithm, these models are not scalable to a real-world scenario (for example, clustering appearances of people in a camera surveillance network), motivating investigations toward unsupervised models in which the number of clusters is estimated according to the complexity of an ever increasing volume of data. To this end, this chapter presents a novel application of the Dirichlet Process Mixture Model (DPMM) framework for clustering SPD matrices.

The fundamental problem to be tackled in developing a mixture model is to define a probability measure on the underlying data that captures its structure effectively. Covariances provide a wide range of interpretations. For example, their positive definiteness structure favors an interpretation as points on a smooth Riemannian manifold, allowing us to utilize tools from differential geometry for clustering (such as spectral clustering using the Riemannian metric). Covariance matrices belong to a special category of invertible matrices due to their positive eigenvalues. As a result, they offer several algebraic properties and forms a lie-group [Arsigny et al., 2008]. Positive definite matrices arise as natural generalizations of density functions in quantum information theory [Kak, 2007]. Covariances are fundamental objects in statistics and defines the spread of the data around the distribution mean. Since our approach is statistical, we will mainly stick to the statistical interpretation of covariances in this chapter; although we will also explore a potential alternative of embedding the covariances into a vector space through its log-Euclidean projection, which will enable the extension of well-known DPMM models (such as the Gaussian-Inverse Wishart model) to these embeddings. In Appendix C, we will derive a potential distribution on the Riemannian manifold of covariances with the Riemannian metric

as the probability measure.

Some of the well-known statistical measures on SPD matrices are the matrix Frobenius norm, log-Euclidean distance and the LogDet divergence. The first two measures can be used to embed the data matrices into the Euclidean space, while the third measure operates directly in the matrix space. Since a Euclidean embedding can essentially distort the structure of the data, we primarily focus in developing the framework in this chapter based on the LogDet divergence measure, and systematically derive the associated model components toward Bayesian clustering.

4.2 Related Work

DPMMs provide a standard technique for unsupervised Bayesian clustering and has been successfully utilized in a variety of domains like genomics [Xing et al., 2007], vision [Sudderth et al., 2006], data modeling [Bouguila and Ziou, 2010], etc. All these references use a Gaussian-Inverse-Wishart (GIW) DPMM for clustering vector valued data. A multivariate Gaussian distribution is assumed on the data; the cluster mean of which is sampled from a prior Gaussian model, and the covariance matrix is sampled from an inverse-Wishart (IW) distribution. In [Vogt et al., 2010], a translation invariant Wishart DPMM is presented for clustering distance data. This is similar to the GIW model, except that the cluster covariance matrix comes from a Wishart distribution. Bayesian density estimation for functional data analysis is proposed in [Rodríguez et al., 2007]. To the best of our knowledge, the Dirichlet process framework has never been applied to clustering SPD matrices before.

4.3 Dirichlet Process Mixture Model

Non-parametric Bayesian techniques seek a predictive model for the data so that the complexity and accuracy of the model grows with the data size. The existence of such a statistical model is invariably dependent on the property of exchangeability [Pitman and Picard, 2006] of observations. This leads to the De Finetti's theorem, which states that if a sequence of observations $y_1, y_2, \cdots, y_n$ is infinitely exchangeable, then there exists a mixture representation for the joint

distribution of these observations. That is,

$$p(y_1, y_2, \cdots, y_n) = \int_{\Theta} p(\theta) \prod_{i=1}^{n} p(y_i|\theta) d\theta \tag{4.1}$$

where Θ is an infinite-dimensional space of probability measures and $d\theta$ defines a probability measure over distributions.

A Dirichlet Process, $DP(\alpha, H)$, parameterized by a concentration α and a prior H, is a stochastic process that defines a distribution over probability distributions adhering to (4.1) such that if $A_1, A_2, \cdots, A_r$ represent any finite measurable partition of Θ. If $G \sim DP(\alpha, H)$, then the vector of joint distributions of samples from G over these partitions follow a Dirichlet distribution, $Dir(.)$ [Ferguson, 1973]. That is,

$$(G(A_1), \cdots, G(A_r)) \sim Dir\left(\alpha H(A_1), \cdots, \alpha H(A_r)\right) \tag{4.2}$$

In our pertinent problem of finding the number of clusters in the given dataset, we would like to find the distribution G over each of the clusters automatically, by computing the posterior distribution of G given the observations and the prior model H. Fortunately, as is shown in [Ferguson, 1973], the posterior distribution has the following simple form:

$$\begin{aligned} p\left(G|y_1, \cdots, y_n\right) &\sim Dir\left(\alpha H(A_1) + n_1, \cdots, \alpha H(A_n) + n_r\right) \\ &\sim DP\left(\alpha + n, \frac{1}{\alpha + n}\left(\alpha H + \sum_{i=1}^{n} \delta_{y_i}\right)\right) \end{aligned} \tag{4.3}$$

where $n_1, n_2, \cdots, n_r$ represent the number of observations falling in each of the partitions $A_1, A_2, \cdots, A_r$ respectively, n is the total number of observations, and δ_{y_i} represents the delta function at the sample point y_i. There are some subtle points to be noted from the (4.3): (i) DP acts as a conjugate prior for the distribution over distributions, and (ii) each observation only updates the corresponding component in the Dirichlet distribution. The latter property implies that the underlying distribution is essentially discrete with probability one and is agnostic over the topology of the underlying space in which the data lie. It was shown in [Sethuraman, 1994] that this property leads to a concentration of the posterior distribution towards the mean, leading to a clustering effect [Sudderth, 2006]. This also paves the way for a pòlya-urn [Blackwell and MacQueen, 1973] scheme or a Chinese Restaurant Process (CRP) model [Pitman and Picard, 2006] for sampling from the posterior distribution, a variation of which is used in this chapter.

For further reading on DPMMs and their application to computer vision problems, refer [Sudderth, 2006].

For the above model to be practically useful and tractable, it is a general practice to model the dependency of the observations y_j to G through a parameterized family $F(\theta_i)$ (where F is the likelihood function with parameters θ_i). This leads to a mixture model characterization of the DP as follows:

$$\begin{aligned} y_i|\theta_i &\sim F(\theta_i) \\ \theta_i|G &\sim G \\ G &\sim DP(\alpha, H) \end{aligned} \tag{4.4}$$

Since the exact computation of the posterior is infeasible when data size is large, we resort to a variant of MCMC algorithms, namely, the collapsed Gibbs sampler [Neal, 2000] for faster approximate inference.

This is largely the model we use in this chapter and can be shown pictorially using the plate model as follows:

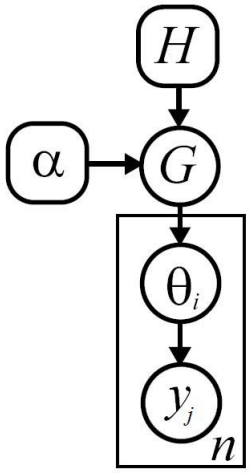


Figure 4.2: Plate model showing the relations between the various model distributions. The data y_j is assumed to be sampled from a distribution parameterized by θ_i. These parameters are sampled from a distribution G that defines a probability measure over the infinite-dimensional space Θ defined by the $DP(\alpha, H)$.

Even though this is a simplified model of the DPMM with an adequate sampling mechanism, the exact computation of posterior expectations is infeasible when the data size is large

[Neal, 2000]. This motivated us to consider Markov Chain Monte Carlo (MCMC) methods for inference. MCMC algorithms lead to the posterior as their stationary distributions when the underlying distributions (F and H) are conjugate distributions. In this chapter, the collapsed Gibbs sampling strategy [Neal, 2000] is used for faster posterior convergence. In this method, we iteratively remove data items one at a time from each of the clusters and compute the predictive distribution of this item to belong to each of the existing clusters, along with its probability to belong to a new cluster. A sample is then drawn from this predictive distribution over the clusters and the data item is added to the respective cluster (which this sample points to). This process is repeated until convergence to a stationary posterior.

The discussion in this section and the formulas we seek in the context of covariance matrices are summarized in Table 4.1.

1) Define a likelihood distribution F on data using a suitable distance measure.
2) Find a prior H that is conjugate to F.
3) Model a collapsed Gibbs sampler over the posterior distribution from (1) and (2) as follows:
3a) Remove a data point y from a cluster C and update the sufficient statistics of C.
3b) Compute the predictive distribution $p(y\|y_{C_i})$ for each of the existing clusters; y_{C_i} represents data in cluster C_i. Also, compute the probability of formation of a new cluster defined by the concentration parameter α of the DP model.
3c) Sample a cluster from $p(y\|y_{C_i})$ and assign the data point to that cluster.
3d) Repeat the steps (3a), (3b) and (3c) until convergence.

Table 4.1: Overview of DPMM algorithm

4.4 Mathematical Preliminaries

This chapter approaches the problem of deriving the DPMMs for covariance clustering as an extension to the hard-clustering algorithms. Thus we find analogues of three traditional distance measures in the Bayesian context, namely (i) LogDet divergence (ii) matrix Frobenius norm, and (iii) log-Euclidean embedding. This section details the mathematical preliminaries required for our analysis in the subsequent sections.

4.4.1 Bregman Divergence

Let us recall some basic facts about Bregman divergences that we discussed in the last chapter. This discussion will be useful for understanding the LogDet divergence measure on which we will build our first DPMM for covariances. Bregman divergence [Bregman, 1967] is a generalized distance measure over convex functions. It has the following general form: Let $\psi : S \rightarrow \mathbb{R}, (S \subseteq \mathbb{R}^d)$ be a function of Legendre type in the $relint(S)$. The Bregman divergence $d_\psi : S \times relint(S) \rightarrow [0, \infty)$ is defined as:

$$d_\psi(x, y) = \psi(x) - \psi(y) - \langle x - y, \nabla\psi(y)\rangle \tag{4.5}$$

where $\nabla\psi(y)$ is the gradient vector of ψ evaluated at y. The squared Euclidean distance, $d_\psi(a, b) = \|a - b\|^2$, is an example of a Bregman divergence corresponding to the convex function $\psi(x) = \frac{1}{2}\|x\|^2$. See [Banerjee et al., 2005] for details.

Bregman divergences can be extended to matrices as follows. If X and Y are matrices, and if Ψ is a real-valued, strictly convex function over matrices, then the Bregman matrix divergence can be defined as:

$$D_\Psi(X, Y) = \Psi(X) - \Psi(Y) - tr\left(\nabla\Psi(Y)^T(X - Y)\right). \tag{4.6}$$

For example, when $\Psi(X) = \|X\|_F^2$, then the corresponding Bregman matrix divergence for two matrices A and B is the squared Frobenius norm $\|A - B\|_F^2$. See [Davis and Dhillon, 2007], [Dhillon and Tropp, 2007] for other examples.

4.4.2 Exponential Family

Let Ω be a sample space (discrete or continuous) and let $\Theta \subseteq \mathbb{R}^d$ be an open set of parameters. Further, assume, $\Phi = \{\phi_\alpha, \alpha \in I\}$ be a vector of sufficient statistics. Let $x \in \Omega$ be a random variable. If $\theta \in \Theta$ and if $p_0 : \Omega \rightarrow \Theta$ is a probability base measure, then a *regular exponential family* is defined as the family of probability distributions of the form:

$$p(x|\theta) = p_0(x) \exp\left\{\langle\theta, \phi(x)\rangle - \eta(\theta)\right\}. \tag{4.7}$$

Here $\phi(x)$ is the *natural statistic* and θ is the *natural parameter*. $\eta(\theta)$ is the *cumulant function* and normalizes $p(x|\theta)$ so that it integrates to one [Sudderth, 2006], [Banerjee et al., 2005].

4.4.3 Bregman Divergence-Exponential Family Bijection

A useful property of Bregman divergences is their connection to the exponential family distributions. It is shown in [Banerjee et al., 2005] that there exists a unique Bregman divergence corresponding to every regular exponential family. Thus, if D_ψ is a Bregman divergence associated with the convex function ψ and if ϕ is a conjugate function to ψ, then the regular exponential family, $p_\psi(x|\theta)$, parameterized by θ, can be written as:

$$p_\psi(x|\theta) \propto \exp\left(-D_\psi(x, \mu(\theta))\right) g_\phi(x) \tag{4.8}$$

where $\mu(\theta)$ is the mean of the distribution and $g_\phi(x)$ is a function uniquely determined by ϕ. See [Banerjee et al., 2005] for details.

4.4.4 LogDet Divergence

The LogDet divergence, D_{ld}, (also known as *Stein's loss*) defines the KL-divergence between two equal-mean Gaussian distributions [Davis and Dhillon, 2007]. Given two SPD matrices $C_1, C_2 \in \mathcal{S}^p_{++}$, we have:

$$D_{ld}(C_1, C_2) = Tr(C_1 C_2^{-1}) - log|C_1 C_2^{-1}| - p \tag{4.9}$$

where $|\ .\ |$ stands for the matrix determinant. This divergence is not a metric, since it is not symmetric and does not satisfy the triangle inequality. Nevertheless, it is a Bregman matrix divergence for the convex function $-log|.|$, and has been utilized in soft clustering algorithms [Davis et al., 2007].

4.4.5 Matrix Frobenius Norm

Given $C_1, C_2 \in \mathcal{S}^p_{++}$, the matrix Frobenius distance $D_F(C_1, C_2)$ measures the Euclidean distance between each of the elements of the matrices and is defined as:

$$D_F(C_1, C_2) = \|C_1 - C_2\|_F = \|Vec(C_1) - Vec(C_2)\|_2 \tag{4.10}$$

where $Vec(.)$ is the matrix *vectorization* operator. It is easy to show that $D^2_F(.,.)$ is a Bregman divergence.

4.4.6 Log-Euclidean distance

Given $C \in \mathcal{S}_{++}^p$, the matrix logarithm $log(C)$, is a symmetric matrix and is no more restricted to the cone of SPD matrices. Using this observation, [Arsigny et al., 2008] proposes the log-Euclidean Riemannain metric $D_{le}(C_1, C_2)$ as follows:

$$D_{le}(C_1, C_2) = \| \log(C_1) - \log(C_2) \|_F \tag{4.11}$$

where $C_1, C_2 \in \mathcal{S}_{++}^p$. Note that this metric is similar to (4.10), except that it first projects the matrices into their tangent space using the $\log$ operator and later embed them into the Euclidean space. Such an indirect projection of the covariances is assumed to reduce the distortion produced otherwise from a direct embedding.

4.4.7 Wishart Distribution

Let $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N$, ($\mathbf{x}_i \in R^p$), be iid random vectors such that $\mathbf{x}_i \sim \mathcal{N}(0, \Sigma)$, for $i = 1, 2, \cdots, N$ and let $X \in \mathcal{S}_{++}^p$ such that $X = \sum_{i=1}^N \mathbf{x}_i \mathbf{x}_i^T$. If we define $n = N - 1$, then X is said to follow a non-singular p-dimensional Wishart distribution $W(n, p, \Sigma)$, with n degrees of freedom ($n > p - 1$) and scale matrix Σ if it has a probability density defined by:

$$W(X; n, p, \Sigma) = c(n, p) \frac{|X|^{n-p-1/2}}{|\Sigma|^{n/2}} exp \left\{ -\frac{1}{2} \operatorname{Tr}(\Sigma^{-1} X) \right\}, \tag{4.12}$$

where $c(n, p)$ is the normalizing constant defined as follows:

$$c(n, p) \; = \; \frac{1}{2^{rp/2} \Gamma_p(\frac{r}{2})} \tag{4.13}$$

Here, $\Gamma_p(s)$ is the multivariate Gamma function. Similarly, using the same notation, we define Inverse-Wishart distribution with scale parameter S,

$$IW(\Sigma; n, p, S) \; = \; \frac{|S|^{\frac{n}{2}} |\Sigma|^{-\frac{(n+p+1)}{2}}}{c(n, p)} exp \left(-\frac{1}{2} \operatorname{Tr}(\Sigma^{-1} S) \right). \tag{4.14}$$

4.5 DPMM on SPD Matrices

In this section, we derive the probability densities and the predictive distributions associated with the base measures introduced above.

4.5.1 LogDet-Wishart Connection

The LogDet divergence is a Bregman matrix divergence and thus utilizing the Bregman-exponential family bijection [Banerjee et al., 2005], we can derive the associated likelihood distribution. It turns out that this exponential family is the Wishart distribution as stated in the following theorem:

Theorem 17. *Let X_c be the covariance matrix of N iid zero-mean Gaussian-distributed random vectors $\mathbf{x}_i$, i.e. $X_c = \frac{1}{n}\sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i{}^T$ where $\mathbf{x}_i \in \mathbb{R}^p$, $\mathbf{x}_i \sim \mathcal{N}(0, \Sigma)$ and $n = N - 1$. Then the probability density of X_c follows:*

$$p(X_c|N, p, \Sigma) = W(X_c; n, p, \Sigma) \propto \exp\left\{-\frac{1}{2} D_\psi(\Sigma, X_c)\right\} p_0(X_c),$$

where D_ψ is the Bregman matrix divergence function for the convex function $\psi(X) = -n\, log|X|$ and p_0 is the base measure.

Proof. Let $X \in \mathcal{S}_{++}^d$ and assume $X = \sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i^T$. Thus, $X = nX_c$. From the definition of the Wishart distribution, we have $X \sim W(n, p, \Sigma)$. Substituting for X and rearranging the terms, we get:

$$\begin{aligned} p(X|N, p, \Sigma) &\propto |X_c|^{-(p+1)/2} \exp\left\{-\frac{n}{2}\left[Tr\left(\Sigma^{-1} X_c\right) - log|\Sigma^{-1} X_c|\right]\right\} \\ &\propto \exp\left\{-D_\psi(\Sigma, X_c)\right\} |X_c|^{-(p+1)/2} \end{aligned}$$

□

4.5.2 Inverse-Wishart Conjugacy

In Bayesian statistics, priors capture the belief about the unknown parameters used in the experiment. In order for inference to be tractable, in general, we try to model this belief in terms of priors that are conjugate to the likelihood of the model. A probability distribution $p_c(.|.)$ is said to be conjugate to a likelihood distribution $l(.)$, if the posterior distribution $l(.)p_c(.|.)$ is of the same distribution model as $p_c(.|.)$. Further, the distribution is said to be closed under sampling and given the prior, we can find formulas for the posterior that can be easily updated [Fink, 1997], [Box and Tiao, 1992]. In this section, we derive the prior model for the Wishart distribution and show that it is the Inverse-Wishart distribution.

Theorem 18. *Given a Wishart distribution $W(n, p, \Sigma)$, the conjugate prior for this distribution is the Inverse-Wishart distribution parametrized as $IW(n, p, S)$ where S is the inverse scale matrix.*

Proof. The *Neyman-Pearson factorization theorem* [Neyman and Pearson, 1933] is used to prove this result. Using the definition of Wishart distribution in (4.12), the joint likelihood of m independent Wishart distributed SPD matrices $X_i \sim W(n, p, \Sigma)$ can be written as:

$$p(X_1, \cdots, X_m|N, p, \Sigma) \propto \left\{\prod_{i=1}^{m} |X_i|^{(n-p-1)/2}\right\} \exp\left\{-\frac{1}{2}tr\left(\Sigma^{-1}\sum_{i=1}^{m} X_i\right) + \frac{nm}{2}\log\left|\Sigma^{-1}\right|\right\} \tag{4.15}$$

This equation can be easily factorized as $h(X)\mathbf{G}_{\Sigma^{-1}}(\bar{X})$, where $\bar{X}$ is the maximum likelihood estimate (sufficient statistic) for the scale matrix Σ, $h(.)$ is a function purely of the data matrices X_i and $\mathbf{G}_{\Sigma^{-1}}(.)$ is a function dependent on the data only through the sufficient statistic. Thus, as suggested in [Fink, 1997], an ideal prior is a distribution over the inverse of the scale matrix Σ^{-1}, which is the Inverse-Wishart distribution [Johnson and Wichern, 1998] IW(n, p, S) for an inverse scale matrix S. □

4.5.3 Predictive Distribution

As was mentioned in the algorithm described in Table 4.1, the next step to build the DPMM is to formulate a collapsed Gibbs sampling framework for inference, which requires deriving the predictive distribution. This is given in the following theorem.

Theorem 19. *Let $X_i \in \mathcal{S}_{++}^p$, $i = 1, 2, \cdots, N-1$, belong to a cluster C such that $X_i \sim W(n, p, \Sigma)$, where Σ is the Wishart scale matrix and n, the degrees of freedom. Let $\Sigma \sim IW(n, p, S)$ with inverse scale matrix S. Then the predictive distribution of a data matrix X_N to belong to the cluster $C = \{X_1, X_2, \cdots, X_{N-1}\}$ will be:*

$$\begin{aligned} p(X_N|X_1X_2...X_{N-1}) &= \int_{\delta_p^+} p(X_N|\Sigma)\, p(\Sigma|X_1X_2...X_{N-1})d\Sigma \\ &= \frac{\omega\left((N+1)n, p\right)}{\omega(n,p)\,\omega(Nn,p)} \frac{|X_N|^{\frac{(n-p-1)}{2}}\, |\sum_{i=1}^{N-1} X_i + S|^{\frac{Nn}{2}}}{|\sum_{i=1}^{N} X_i \,+\, S|^{\frac{(N+1)n}{2}}} \end{aligned}$$

where $\omega(n,p) = \int_{\mathcal{S}_{++}^p} |Y|^{\frac{n-p-1}{2}} \exp\left\{-\frac{1}{2}tr(Y)\right\} dY$.

Proof. See Appendix B. □

4.5.4 Frobenius Norm based DPMM

Using the base measure as the matrix Frobenius norm, it can be shown that the exponential family for the associated Bregman divergence is the multivariate normal distribution. That is, given $X,\ \mu_x \in \mathcal{S}^p_{++}$, and a variance σ^2,

$$p(X|\mu_X, \sigma^2) \propto \exp\left(-\frac{\|X-\mu_X\|_F^2}{2\sigma^2}\right) \propto \exp\left(-\frac{\|\mathcal{V}(X)-\mathcal{V}(\mu_X)\|_2^2}{2\sigma^2}\right) \tag{4.16}$$

where $\mathcal{V} : \mathcal{S}^p_{++} \rightarrow \mathbb{R}^{p(p+1)/2}$ is the half-vectorization operator. The variance σ^2 in the (4.16) can be generalized using a covariance matrix Σ between the vectorized matrices, leading to a standard GIW DPMM, where the $\mathcal{V}(\mu_X) \sim \mathcal{N}(\mu, S_1)$ and $\Sigma \sim IW(n, p, S_2)$; S_1, S_2 being the hyper-scale matrices and μ is the hyper-mean [Beal et al., 2002].

4.5.5 Log-Euclidean based DPMM

Similar to the approach above, we can derive the associated density function for the log-Euclidean distance, (4.11). Using the Euclidean embedding suggested in [Pennec et al., 2006], the density function takes the form:

$$p(X|\mu_X, \sigma^2) \propto \exp\left(-\frac{1}{2}\frac{\|\mathcal{V}(log(X))-\mu_x\|_2^2}{\sigma^2}\right) \tag{4.17}$$

where $\log(.)$ is the matrix logarithm, $\mu_X = \mathcal{V}(1/N\sum_{i=1}^{N}\log(X_i))$ and σ^2 is the assumed variance. We can approximate μ_X to follow a multivariate normal distribution, in which case the DPMM follows the standard GIW model as mentioned earlier.

4.6 Experiments and Results

This section details the empirical evaluation of the DPMMs to the state-of-the-art clustering techniques on covariance matrices. First, we introduce two metrics on which the performance is defined. Later, results on simulated data and real appearance data are presented.

4.6.1 Purity

A standard way to compare clustering performance is the rand-index measure [Rand, 1971]. However, to better explore the representation power of covariance matrices as a means of data

representation and to evaluate the ability of the DPMM to automatically cluster the data, we define variants of the rand-index measure which we call *purity*. Performance results of our methodology are analyzed and presented in the light of two such purity measures: (i) cluster purity and, (ii) class purity. Cluster purity captures the ability of the proposed methodology (and the associated metric employed) to partition the symmetric positive definite matrix data in the multi-dimensional space they exist. It is defined for every cluster automatically created by the DPMM process as the fraction of class instances that dominate the respective cluster. For example, in Figure 4.3, cluster 1 is dominated by class instances denoted by the triangles, although instances belonging to another class (denoted by the stars) are also included in the same cluster. Mathematically, we can write this measure as follows: for a cluster C_i,

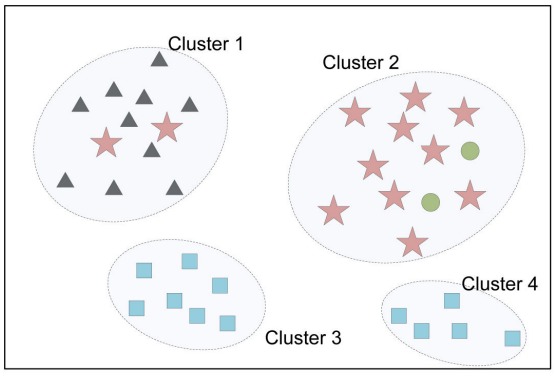


Figure 4.3: Illustration of the measure of purity

if $label(C_i)$ represents the set of labels of all the data points in C_i, then we define the cluster Purity, $P_{cluster}$, of C_i as:

$$P_{cluster}(i) = \frac{\#\left\{label(C_i) = \ell^*\right\}}{\#C_i} \text{ where } \ell^* = \max_{\ell} \#\left\{label(C_i) = \ell\right\}, \tag{4.18}$$

where $\#\{.\}$ defines the cardinality of the set. This metric alone might not be a good measure of the clustering performance. For example, in the cluster 2 in Figure 4.3, there are circle labels which have a low cardinality as compared to star labels and are ignored in $P_{cluster}$. Another problem occurs when data gets split into multiple clusters, each cluster is pure in itself (for example, cluster 3 and cluster 4 in Figure 4.3). To account for this, we use the class purity P_{class} metric. Class purity captures the complexity of the data exposed by the form of data representation used (in our case covariance matrices of intensity and gradient intensity information). It also reflects the clustering challenge experienced by the DPMM whereby instances of

a single class are assigned to multiple clusters not necessarily dominating the assigned cluster. Thus class purity helps better understand if the feature vectors that we chose for building the covariances adequately capture the differentiating properties of the classes. For a label ℓ,

$$P_{class}(\ell) = \frac{\#\left\{label(C_{k^*}) = \ell\right\}}{\#C_{k^*}} \text{ where } k^* = \max_k \#\left\{label(C_k) = \ell\right\} \tag{4.19}$$

The *Purity* P of clustering is defined as the weighted sum of (4.18) and (4.19) for all the clusters, i.e.

$$P = \frac{1}{2}\left[\sum_{i=1}^{i=K} \frac{P_{cluster}(i)}{K} + \sum_{\ell=1}^{\ell=L} \frac{P_{class}(\ell)}{L}\right] \tag{4.20}$$

for K clusters produced by the algorithm and L true class labels. For a perfect clustering, purity will be unity and the number of clusters discovered by the DPMM should match the true number.

4.6.2 Simulated Experiment

This section evaluates the correctness of the WIW DPMM on a small simulated dataset. The dataset consisted of 100 covariance matrices of dimension 5×5. Each covariance matrix was generated from 1K normal distributed random vectors of dimension 5×1, thus fixing the degrees of freedom parameter in the WIW model. Six different Gaussian distributions were used to create the dataset. Figure 4.4 plots an ISOMAP embedding of the clusters found by the DPMM after 10 iterations of the Gibbs sampling. The purity for this clustering was found to be 0.97.

Increasing Cluster Sizes The accuracy of clustering for an increasing number of true clusters is evaluated in this section for the WIW model. The true clusters were increased from 10 to 100, with each of the true clusters having at least five covariance matrices from the same underlying normal distribution. Figure 4.5 shows the results of this experiment. As is clear from the plot, the model was able to find the clusters with an average purity score of more than 0.8, while maintaining the number of clusters discovered close to the ground truth.

4.6.3 Experiments on Real-Data

This section details the datasets used for performance evaluation. The choice of the datasets were based on three properties of the DPMM that we thought to evaluate: (i) performance

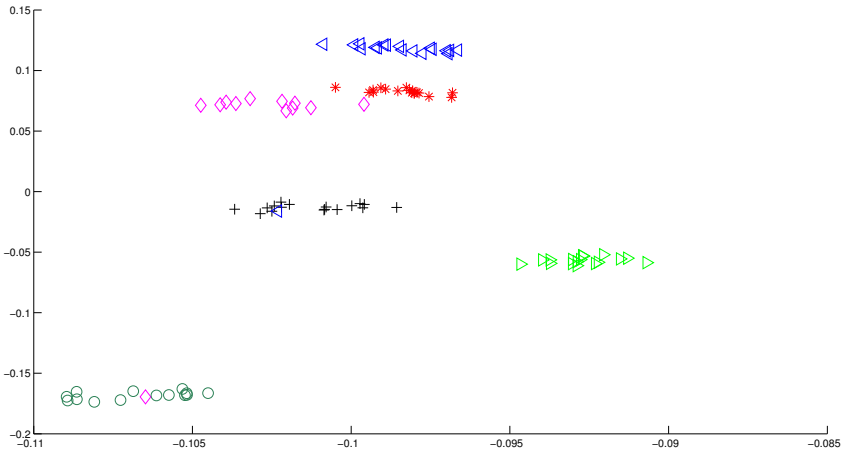


Figure 4.4: ISOMAP embedding of clustered covariances on 100 covariance matrices with six true clusters. Data belonging to each cluster is shown with a unique symbol.

on noisy real-world data, (ii) robustness to clustering appearances from multiple cameras, and (iii) performance against the dimensionality of covariances. For (i) and (ii), the dataset shown in Figure 4.6 was used, which contained background subtracted people appearances from real-time tracking sessions in our lab recorded using two cameras. The dataset contained 900 covariances and 30 true clusters. We used a five dimensional feature vector, $F = [I_R, I_G, I_B, I_x, I_y]^T$, to construct each covariance matrix, where the first three dimensions of F are the pixel intensities in the three color channels red, green and blue respectively, while I_x, I_y stand for the gray level pixel gradients in x and y directions respectively. Thus each covariance matrix used in this experiment was 5×5. All the appearances were resized to 100×100 and 3000 points were randomly sampled, thus fixing the number of degrees of freedom in the DPMM. The true cardinality of the clusters varied from 5 to 50. To evaluate (iii), we used the FERET face appearance dataset [Phillips et al., 2000]; a few sample images of which are shown in Figure 4.6. We used 110 different face classes from this dataset, with each class containing 7 different poses of the same person. Covariances of size 40×40 were created using 40 Gabor-filters as suggested in [Pang et al., 2008].

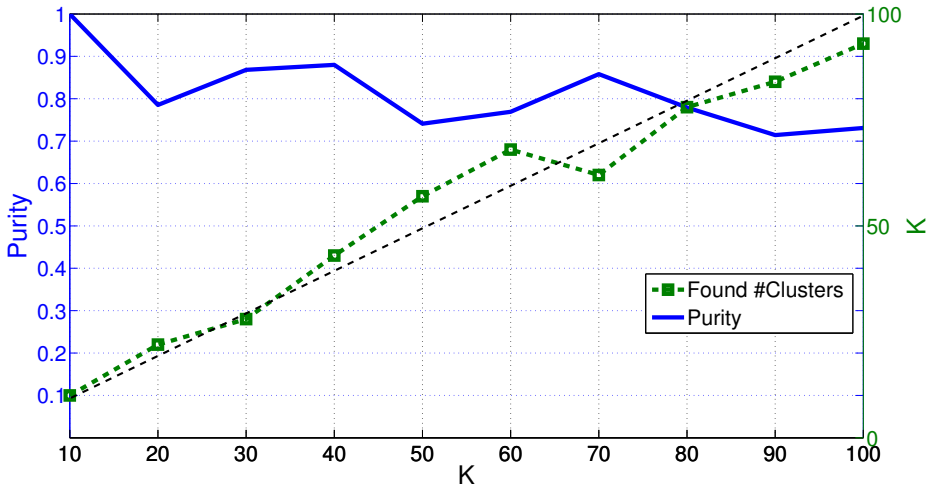


Figure 4.5: Simulations with the true number of clusters increasing from 10 to 100. Left y-axis is Purity, right y-axis is the number of clusters discovered and x-axis shows the ground truth. The dotted straight line is drawn for an ideal case comparison.

4.6.4 Setup

The algorithms were implemented in Matlab. The concentration parameter of the DPMM was initialized to 1 and then re-estimated at each iteration from a mixture of gamma distributions as described in [West, 1992]. A gamma prior, G(30,60), gave the best performance for all the three DPMMs. Hyperparameters of the models were estimated by taking the mean of all the covariances in the dataset, while the initial allocation of the data points was done using K-means algorithm. The collapsed Gibbs sampler for the inference model converged in less than 20 iterations.

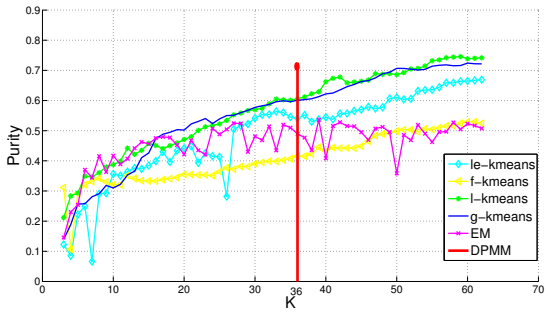
4.6.5 Results

The clustering performance of the WIW model for the two datasets is given in Figure 4.7(a) and Figure 4.7(b). The model is compared against K-means algorithms using the Riemannian geodesic distance (g-kmeans), the symmetric LogDet divergence (l-kmeans), Matrix Frobenius distance (f-kmeans), log-Euclidean distance (le-kmeans) and the EM-algorithm based on mixture of Wishart distributions [Hidot and Saint-Jean, 2010]. The cluster means for the K-means was computed using the Karcher mean algorithm described in [Pennec et al., 2006]. Since we

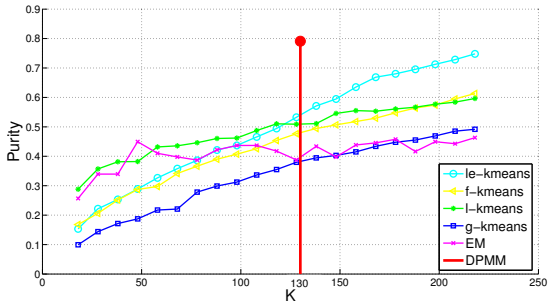
Figure 4.6: Top: Sample images from the appearance tracking dataset. Bottom: Sample images from the FERET face appearance dataset.

assume we do not know the true number of components in the hard/soft-clustering algorithms, the number of components in each model was increased from 3 to 60 for the people dataset and from 18 to 220 for the face dataset; which forms the x-axis of the plots. As is clear from the plots, the WIW model automatically figures out the true clusters and its purity score was found to outperform the scores of all the other algorithms, at the same time keeping the identified number of clusters close to the true number. This dominance of the WIW model is more discernible in the plots from the face dataset Figure 4.7(b).

Next, we evaluate the performance of the DPMMs against each other on the above datasets. Figure 4.8 shows the results on the people appearances dataset for the Frobenius-DPMM (P-Frob), the log-Euclidean-DPMM (P-LE) and the WIW model (P-WIW), along with the results on the face appearance dataset (F-Frob, F-LE and F-WIW respectively). As is clear from the plots the WIW model performs better than the vectorization based methods. This argument goes stronger with the face dataset, where vectorization methods did not seem applicable at all. This is perhaps due to the curse of dimensionality as the vectorization of a 40×40 matrix produces an 820 dimensional vector. This vector requires a very large dataset for effective clustering. This seems to be a major limitation with the vectorization methods. In addition, we also observed that for all the three models, the number of clusters discovered in the people appearance data remained in the range of 24 to 36 (30 is the ground truth), while the F-WIW model found 130



(a)



(b)

Figure 4.7: (a) compares the clustering performance using the People dataset and, (b) compares the clustering performance using the FERET face dataset. x-axis is the number of components used for the hard/soft clustering algorithms and y-axis measures purity.

clusters (110 is ground truth). Those for the vectorization methods of the face dataset deviated considerably from the true number. Another observation to be pointed out from the class purity scores from Figure 4.8 is their high value for the faces dataset, while relatively lower scores for the people appearances. This points to the inadequacy of the features used in creating the covariances for the latter dataset and the scope for improvement. A sample visualization of clustering using the WIW model for a random subset of the people appearances dataset (115 appearances and 3 true clusters) is shown in Figure 4.9.

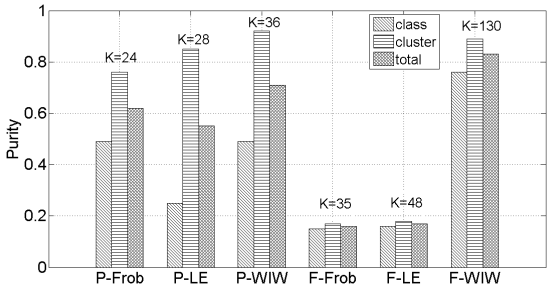


Figure 4.8: Comparison of various DPMMs for people appearance and face datasets. The K value in the x-axis shows the number of clusters found by the DPMM; the ground truth being K=30 for the first three plots and K=110 for the last three.

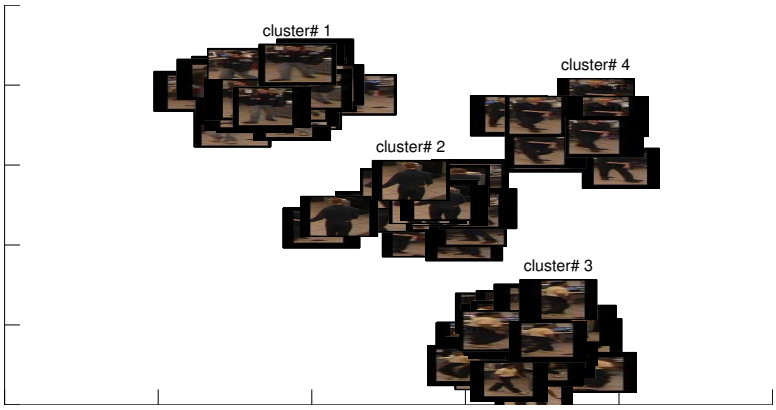


Figure 4.9: A visualization of clustering using WIW model on a subset of the people appearances dataset. The result is for a purity score of 0.83, while the true number of clusters is 3 (Note that appearances in clusters 2 and 4 are actually of the same person, but differs slightly).

Computational Cost: Table 4.2 compares the cost per iteration of various algorithms. Since the complexity is proportional to the number of clusters the algorithm operates on, this value is also shown. WIW model seems to scale well with the dimensionality of the data compared to other algorithms. This is justified as there is no closed form solution to compute the mean of the covariances as in the K-means case, and the need to compute the inverse of a matrix in the case of EM algorithms with mixture of Wishart distributions.

$Dataset$	EM	$K-means$	$DPMM$
People(#clusters)	1.49(50)	2.3(50)	**3.48(50)**
Face(#clusters)	160.11(10)	116.54(10)	**11.26(50)**

Table 4.2: Computational performance of various clustering algorithms. Each entry in the table lists the time taken (in seconds) for one iteration of the algorithm. Number of clusters is shown inside bracket.

Part II

Vector Valued Data

Chapter 5

Nearest Neighbors via Sparse Coding

In the previous chapters, we discussed NN problems on covariance valued data. In this chapter and the next few chapters, we will be discussing NN algorithms for efficient NN retrieval on vector valued datasets. Such datasets are important not only in computer vision, but also in several other areas including data mining, pattern recognition and robotics. Let us recall a few vector based data descriptors commonly used in computer vision. All the histogram based descriptors (such as Scale Invariant Feature Transform (SIFT) descriptors [Lowe, 2004], shape contexts [Mori et al., 2001], Histogram of Oriented Gradients (HOG) [Dalal et al., 2006], Speeded Up Robust Feature (SURF) [Bay et al., 2008], etc.) fall under the category of vector valued descriptors. This class also includes descriptors formed from filter outputs (such as the GIST descriptor [Murphy et al., 2006]) or vectorized image intensity patches. NN retrieval on these data descriptors is an important algorithmic component in applications that use them. A few important applications are 3D reconstruction from images [Snavely et al., 2006] (which is the basic ingredient of applications such as the Google Street View), object recognition [Murphy et al., 2006], action recognition [Dalal et al., 2006], etc. Most of these data descriptors are high dimensional; as a result the problem of NN retrieval is computationally expensive (for reasons we saw in Chapter 2.2). Such high-dimensional data descriptors are not uncommon in other disciplines, such as machine learning or data mining. Motivating examples are document search, content-based information retrieval [Datta et al., 2008], and multimedia retrieval [Bohm et al., 2001].

The problem of NN on vector valued data has been investigated thoroughly over the past few decades and several state-of-the-art methods proposed by several authors (as we reviewed

in Chapter 2). Unfortunately, the problem is still far from being sufficiently solved due to the ever increasing volume of data generated by the new generation devices (such as mobile phones, surveillance cameras, etc.). Such humongous data demands efficient retrieval algorithms, which provided us the impetus to look at this problem from a fresh perspective.

Our approach to solve NN retrieval on vector-valued data is based on the concepts of sparse coding and dictionary learning. In this chapter, we will provide the basics of these concepts, along with details on how these ideas relate to NN retrieval. We will apply our sparse coding based NN retrieval algorithm on the real-world problem of estimating the motion parameters of a camera moving in space (attached to a remote controlled helicopter). This experiment will show the desirable properties of our algorithm, and also point out the caveats. These drawbacks and ways to address them forms the content of the later chapters of this thesis.

5.1 Introduction

Our key tool for NN retrieval on vector valued data is derived from the recent advances in the theory of sparse signal processing, particularly the compressive sensing idea. Compressive sensing deals with the idea of sampling and reconstructing signals that are sparse in an appropriate overcomplete basis, so that perfect reconstruction can be achieved via only a very few samples (as compared to the number of samples prescribed by the Shannon's sampling theorem) [Donoho, 2004]. For general discrete signals, it is non-trivial to find such an overcomplete basis dictionary. To circumvent this problem dictionary learning techniques have been suggested [Murray and Kreutz-Delgado, 2004; Elad and Aharon, 2006] in which the basis is learned from the data itself, by solving an L1 regularized least-squares problem (namely LASSO) [Osborne et al., 2000]. This idea of dictionary learning has been leveraged in the recent times into very successful applications such as image denoising, object classification [Mairal et al., 2009b], etc. A natural question that one might have along these lines is if the sparsity can be utilized to achieve fast NN retrieval on high dimensional data. This chapter looks at this problem in detail and proposes efficient algorithms based on sparse coding and dictionary learning for NN retrieval on image data.

The chapter is organized as follows: We begin with a review of dictionary learning and sparse coding literature in Section 5.2. In Section 5.3, we combine the paradigms of sparsity and NN by introducing our indexable representation of data. In Section 5.4, we will discuss

the data space partitioning produced by dictionary learning, along with exposing the relation of our algorithm compared to other hashing algorithms. In Section 5.5, we will discuss the advantages of our algorithm against the state-of-the-art. Section 5.6 provides details on hashing using sparse coding, which will be followed by an application of our algorithm to a real-world motion estimation problem in Section 5.7. This experiment will help us bring out the challenges that need to be addressed when using our representation. These difficulties will be explicitly discussed in Section 5.8.

5.2 Background

5.2.1 Sparse Coding

Traditional signal processing theory suggests that for reconstructing a signal without loss of information, one has to sample it at twice its maximum frequency. This is the classical Shannon sampling theorem. This idea was practical when working with audio data, for which the maximum frequency range never crossed a few thousand Hertz. In the modern days, we work predominantly with image or video data, for which the frequency ranges at several megahertz or even gigahertz that sampling at a frequency as prescribed by the above theory becomes practically impossible. Over the years, there have been several approaches suggested to overcome this challenge. Examples include the invention of various compression schemes such as those using Discrete Cosine Transforms (DCT), wavelets, curvelets [Candes and Donoho, 2000], etc. In the recent years, a different approach *compressive sensing* [Donoho, 2006; Candes, 2006], has emerged and revolutionized the domain of signal acquisition and reconstruction.

Shannon-Nyquist theorem is a *pessimistic* approach because we allot bandwidth for all frequencies that can possibly arise in the spectrum. In most real-world situations, the signal is often found to be sparse in an appropriate basis. A pedagogical illustration of this idea could be the space-time signals and their frequency domain counterparts. For example, a sine wave which is a space time signal has a dense support in the spatial domain, but can be represented or reconstructed using a single spike in the frequency domain. Now, this sparse support property facilitates a new sampling approach which has predominantly two steps: (i) projecting the dense data vector into the appropriate basis dictionary in which it has a sparse support, and (ii) projection of this sparse support onto a random matrix (that has more columns than rows). Now, it might seem intriguing how one can recover the original signal after projecting onto

a random matrix. Apparently, it can be shown that under suitable conditions, which we will discuss in detail below, the original signal can be recovered with minimal error by solving a simple non-linear optimization problem on the random projection given the random matrix and the sparsifying basis. Below, we ground these ideas in mathematical terms.

Formally, let $x \in \mathbb{R}^n$ be the input signal, and let $\Phi \in \mathcal{O}(n)$ an orthonormal basis such that $x = \Phi w$, for a sparse coefficient vector $w \in \mathbb{R}^n$. Utilizing the sparsity of x in Φ, compressive sensing says that given a suitable *sensing* matrix $\Psi \in \mathbb{R}^{d \times n}$ (where $d \ll n$), a low dimensional sample $v \in \mathbb{R}^d$ (where $v = \Psi\Phi w$) is enough to recover the underlying signal x. This reconstruction of x from v (which is essentially equivalent to finding the underlying sparse vector w from v) can be cast as the following optimization problem: assuming w is k-sparse in Φ, we have

$$\min_w \|v - \Psi\Phi w\|_2^2 \text{ subject to } \|w\|_0 = k. \tag{5.1}$$

where the ℓ_0 pseudo-norm constraints the number of non-zero elements in w to be k.

The formulation (5.1) is practically limited because the ℓ_0 pseudo-norm leads to a combinatorial optimization problem. Solving this optimization problem is known to be NP-hard. A further deterrent is that the problem is non-convex and thus any real value relaxation is futile. To circumvent these issues the method of choice has been to relax the ℓ_0 constraint to its closest convex surrogate, the ℓ_1 norm. The resulting optimization problem (popularly known in statistics literature as the L1 regularized least squares or LASSO) can be cast as:

$$\mathrm{SC}_{\text{LASSO}}(t; v) := \min_w \|v - \mathcal{B}w\|_2^2 \text{ subject to } \|w\|_1 \leq t, \tag{5.2}$$

for some bound t on the sparsity and $\mathcal{B} = \Psi\Phi$ is called a basis dictionary ($\mathcal{B} \in \mathbb{R}^{d \times n}$). We will be using the Constrained Basis Pursuit (CBP) [Yang and Zhang, 2011] in our subsequent discussions that will constrain the reconstruction error instead of the sparsity term.

$$\mathrm{SC}_{\text{CBP}}(\delta; v) := \min_w \|w\|_1 \text{ subject to } \|v - \mathcal{B}w\|_2^2 \leq \delta^2, \tag{5.3}$$

where $\delta \geq 0$ captures potentially noisy measurements. Usually the LASSO problem in (5.3) is also written in the Lagrangian form (this formulation is often called *Basis Pursuit (BP)* [Chen et al., 2001]):

$$\mathrm{SC}_{\text{BP}}(\lambda; v) := \min_w \|v - \mathcal{B}w\|_2^2 + \lambda\|w\|_1, \tag{5.4}$$

for a Lagrangian multiplier $\lambda > 0$.

With reference to the LASSO formulation $\mathrm{SC}_{\mathrm{BP}}(\lambda; v)$, it is clear that the regularization λ weighs the ℓ_1 constraint with respect to the ℓ_2 least squares constraint. This regularization has two effects: (i) model selection and (ii) shrinkage estimation. That is, the ℓ_1 constraint selects the subset of the basis vectors that best represent the given data vector (setting the coefficients for other basis to zero) and at the same time, the non-zero coefficients selected are shrunk towards zero. Further, it can be shown that under the so-called *restricted isometry* conditions, the solution to (5.3) perfectly recovers the desired signal [Candes, 2006].

Over the past few years, there have been several efficient algorithms designed to solve (5.4); popular techniques are: (i) Least Angle Regression (LARS) [Efron et al., 2004], (ii) Orthogonal Matching Pursuit (OMP) [Mallat and Zhang, 1993], and (iii) Iterative Shrinkage [Daubechies et al., 2004]. One of the most popular algorithms for solving the LASSO formulation is the LARS [Efron et al., 2004]. Since we use some of the properties of the LARS solution path for generating robust sparse codes, we provide a brief review of this algorithm here.

5.2.2 Least Angle Regression (LARS)

Amongst several efficient algorithms for sparse coding that solves either the formulation (5.3) or (5.4) [Lee et al., 2005; Malioutov et al., 2005; Gill et al., 2011; Kim et al., 2007], we will be particularly interested in the *Least Angle Regression* (LARS) algorithm; reasons for which will be explicitly described after reviewing this algorithm briefly. Note that, in the future sections, we will use the notation $b_i \in \mathcal{B}$ to say "b_i is a basis vector that represents the ith column of the dictionary $\mathcal{B}$".

The LARS algorithm essentially solves the LASSO formulation in (5.2), that is $\mathrm{SC}_{\mathrm{LASSO}}$ (although it can be easily modified to solve $\mathrm{SC}_{\mathrm{CBP}}$ as we will show below). With reference to (5.2), LARS starts with a coefficient vector $w = 0$ and builds the sparse w in steps, each step finding a basis from the dictionary $\mathcal{B}$ that is most correlated with the residual $C^k = (v - \mathcal{B}w^k)$ at step k. Let us assume, $b \in \mathcal{B}$ is such a basis at the step k. Next, LARS will compute a descent direction $\hat{b}$ that is equiangular to b and all the bases in $\mathcal{B}$ corresponding to the support in w^k, and does a steepest gradient descent in the direction $\hat{b}$, that is $w^{k+1} = w^k + \gamma\hat{b}$, where the step size γ is chosen such that after this single descent, the residual C^{k+1} will be equally correlated with $\hat{b}$ and some other basis $b' \in \mathcal{B}$. The efficiency of LARS comes from the fact that this step size γ can be precomputed. This procedure is repeated until the ℓ_1 constraint on w is violated or the residual is lower than a pre-specified threshold. Algorithm 1 reviews the essential steps

of the LARS algorithm briefly (reader is advised to refer to [Efron et al., 2004] for details).

Algorithm 1 LARS

Require: $\mathcal{B}, v, t$ {or δ so that $\|v - \mathcal{B}w\|_2 \leq \delta$}

$k \Leftarrow 0, w^k \Leftarrow 0, \mathcal{A} \Leftarrow \{\}$
while $\|w^k\|_1 \leq t$ (or $\|v - \mathcal{B}w^k\|_2 > \delta$) **do**
  $C^k \Leftarrow (v - \mathcal{B}w^k)$
  $b^* \Leftarrow \text{argmax}_{\forall b \in \mathcal{B} \setminus \mathcal{A}}(|b^T C^k|)$
  $\mathcal{A} \Leftarrow \mathcal{A} \cup \{b^*\}$
  Compute $\hat{b}$ such that $\widehat{\hat{b}, b_i} = \widehat{\hat{b}, b_j}, \forall b_i, b_j \in \mathcal{A}, i \neq j$
  $w^{k+1} \Leftarrow w^k + \gamma \hat{b}$; Refer [Efron et al., 2004] for details on γ
  $k \Leftarrow k + 1$
end while
return w^k

We would like to highlight a few interesting properties of LARS that will be useful for NN retrieval, (i) the computation of equiangular directions naturally provides a radial partitioning of the data space, each new basis added to the active set $\mathcal{A}$ diminishes the size of the respective partition, and (ii) starting with an empty active set, LARS offers an efficient way to compute the entire trajectories of the bases in the active set, in a single pass. Lower bounds (δ in Algorithm 1) lead to closer approximations in case the data vector is not exactly k-sparse in $\mathcal{B}$. These two properties help control the approximation to the NN we desire. Further, LARS is *less* greedy compared to other sparse coding approaches such as *Orthogonal Matching Pursuit* [Mallat and Zhang, 1993], which helps improve the robustness of the sparse code to data perturbations.

On the other hand, LARS suffers from an issue introduced due to the non-orthonormal nature of the dictionary. While sparse coding using LARS, it is often seen that a few bases already in the active set $\mathcal{A}$ might produce a gradient descent direction that is negatively correlated to some other basis in $\mathcal{A}$. Doing a descent in this direction will reduce the absolute magnitude of the coefficient associated with this basis or can even make this coefficient value to zero, thus prompting LARS to drop this basis from the active set. Heuristic techniques have been suggested in [Efron et al., 2004] to predict when this situation happens. The heuristic is to remove this basis from the active set for one step of the LARS algorithm. Since our data partitioning scheme for NN retrieval is based on the existence of subspaces defined by the active support, such a basis drop will cause difficulties in our algorithm. We will introduce schemes to avoid such basis drop in Chapter 7.

5.2.3 Dictionary Learning

The sparse coding formulation introduced above is useful in a practical setting only when we know the sensing matrix Ψ and the sparsifying basis set Φ. Unfortunately, it is often non-trivial to know these parameters for real-world data, such as images or video. To address this issue, there have been approaches suggested that tries to learn this dictionary from the data itself [Murray and Kreutz-Delgado, 2004]. Assume that we are given a collection of data samples v_k, $(k = 1, \ldots, N)$. Our task is to learn an overcomplete dictionary $\mathcal{B} \in \mathbb{R}^{d \times n} \triangleq \{b_1, b_2, \cdots, b_n\}$ (where each $b_i \in \mathbb{R}^d$ represents the ith column of the dictionary $\mathcal{B}$). With this setup, we define the Dictionary Learning (DL) problem as follows:

$$\min_{\mathcal{B}, w_1, w_2, \cdots, w_N} \sum_{i=1}^{N} \|v_i - \sum_j w_i^j b_j\|_2^2 + \lambda \|w_i\|_1$$
$$\text{subject to } \|b_j\|_2 \leq 1, \ \forall j \in \{1, ..., n\}, \tag{5.5}$$

where the vector w_i is called the *activation vector* or the coefficient vector and the notation w_i^j stands for the jth component of the ith activation vector that sparsifies the data vector v_i. The objective function in (5.5) balances two terms: (i) the quadratic term which minimizes the L_2 error between the sparse representation and the data vector v_i and, (ii) the L_1 minimization term which imposes sparsity on the weights w_i. The parameter λ regularizes the penalty imposed by the L_1 constraint. The problem is convex in either $\mathcal{B}$ or w_i separately, but is not convex in both together, suggesting an alternating minimization strategy which solves each of the convex subproblems iteratively towards a local minimum. There are several methods available today to solve (5.5). A few popular examples are the K-SVD algorithm [Aharon et al., 2006], and MOD algorithm [Engan et al., 2000]. Once the overcomplete basis $\mathcal{B}$ is learned for the data vectors, it can be plugged in the LASSO formulation in (5.4) to find the sparse code associated to a given data vector from the same data distribution.

A quantity that we frequently allude to in this document is the *coherence* of the dictionary, which is defined as follows:

Definition 7. Coherence: *Suppose* $\mathcal{B} = \{b_1, b_2, \cdots, b_n\}$ *is a dictionary where each* $b_i \in \mathbb{R}^d, i = 1, 2, \cdots, n$. *Then we define the* coherence μ *of* $\mathcal{B}$ *as:*

$$\mu = \max_{i<j} |b_i^T b_j|, \ \forall i, j \in \{1, 2, \cdots, n\}. \tag{5.6}$$

Coherence captures the absolute correlation between the bases in the dictionary. Since the bases are unit norm, coherence refers to the absolute cosine of the minimum angle between all pairs of basis vectors in the dictionary. A higher coherence means that the basis vectors might be closer in angles. This might not be favorable when applying sparse coding for NN retrieval, because we will have several bases that can represent a given data point leading to non-unique hash codes (we will detail consider this problem in detail later).

Another quantity that we will be referring to in this chapter is that of the *Regularization Path* (RP). Recall that, the greater the regularization on the ℓ_1 norm (in (5.3) or (5.4)), the more sparsity in the coefficient vector w. It can be shown that the coefficients in w corresponding to the active bases form piecewise linear trajectories [Mairal and Yu, 2012] as a function of λ and RP refers to this path. Formally, we will define the regularization path as follows:

Definition 8 (Regularization Path)**.** *Referring to* (5.4)*, let* $w(\lambda; v)$ *be the solution to* (5.4) *for a data vector* v *and regularization* λ*, and let* $w_i(\lambda; v)$ *be the* i*th dimension of* $w(\lambda; v)$ *corresponding to a basis* $b_i \in \mathcal{B}$*. Then, we define the* Regularization Path *of* b_i *as the set of all values of* $w_i(\lambda)$ $\forall \lambda \in [0, \infty)$.

In addition, we will use the notation $\widehat{p, q}$ to denote the absolute angle between any two vectors p and q.

5.2.4 Dictionary Learning for Image Processing

Usage of overcomplete sparse basis for image models was introduced in [Olshausen and Field, 1997] for modeling the spatial receptive fields of the mammalian visual cortex. The non-orthogonality of the bases in the dictionary offers a degree more flexibility in representing the input signal, compared to an orthogonal basis. Putting sparse constraints on this representation leads to simpler and interpretable representations of the input data. As a result, dictionary learning and sparse coding has been successfully applied to several computer vision problems in the recent times. A few important applications include image denoising [Elad and Aharon, 2006], supervised learning and classification [Mairal et al., 2009b], image super resolution [Mairal et al., 2008b], image and video restoration [Mairal et al., 2008b,a], deblurring [Zhang et al., 2011], among others.

5.3 Sparse Coding and NN Retrieval

In this section, we extend the sparse coding and dictionary learning framework for the problem of NN retrieval. First, we establish the connection between sparse coding and nearest neighbor retrieval, later introducing a simple indexable representation of the sparse codes.

The following result connects the Euclidean distance between sparse coefficient vectors and the Euclidean distance between the actual data vectors. This is a fundamental connection that is necessary to use sparse coding for NN retrieval.

Theorem 20. *If $v_1, v_2 \in \mathbb{R}^d$ are two zero-mean data points and if $\mathcal{B} \in \mathbb{R}^{d \times n}$ is a given dictionary. If $w_i = \mathrm{SC}_{CBP}(\delta; v_i)$ for $i \in \{1, 2\}$, so that $\|v_i - \mathcal{B}w_i\|_2^2 \leq \delta^2$. Then we have:*

$$\|v_1 - v_2\|_2 \leq 2\delta + \|\mathcal{B}\|_2 \|w_1 - w_2\|_2. \tag{5.7}$$

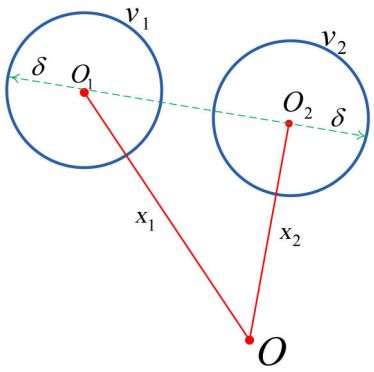


Figure 5.1: With reference to Theorem 20: O is the origin, $x_1 = \mathcal{B}w_1$ and $x_2 = \mathcal{B}w_2$ are two approximate reconstructions of the two zero-mean data vectors v_1 and v_2 respectively. The circles (hyperspheres in the general case) of radii δ and centers O_1 and O_2 represent the locus of the two data points given their approximate reconstructions.

Proof. Let us assume that $x_1 = \mathcal{B}w_1$ and $x_2 = \mathcal{B}w_2$. With reference to Figure 5.1, let us consider the triangle formed by O, O_i and any point on the hypersphere with center O_i (for $i \in \{1, 2\}$). On this triangle, we have the following inequalities:

$$\|v_i\|_2 \leq \|x_i\|_2 + \delta \tag{5.8}$$

Further,

$$\|v_1 - v_2\|_2 \leq \|v_1\|_2 + \|v_2\|_2 \quad \text{(using triangles formed by any points } v_1 \text{ and } v_2 \text{ on their trajectories shown)} \tag{5.9}$$

$$\leq \|x_1\|_2 + \|x_2\|_2 + 2\delta \quad \text{(using (5.8))} \tag{5.10}$$

$$\leq \|x_1 - x_2\|_2 + 2\delta \quad \text{(using triangle inequality on } O, O_1, O_2) \tag{5.11}$$

$$\leq \|\mathcal{B}(w_1 - w_2)\|_2 + 2\delta \quad \text{(substitution of the definitions of } x_1 \text{ and } x_2) \tag{5.12}$$

$$\leq \|\mathcal{B}\|_2 \|w_1 - w_2\|_2 + 2\delta \quad \text{(using the definition of spectral norm).} \tag{5.13}$$

□

5.3.1 Subspace Combination Tuple

Once we have a dictionary to describe a given data vector as a sparse linear combination of the basis vectors, the next step is to formulate a representation of the data vectors in terms of their sparsity. In this section, we will introduce our indexable representation, which we call *Subspace Combination Tuple* (SCT) for reasons that will be apparent soon. Our SCT representation will be synonymous with a hash code for a given data vector.

Definition 9 (Subspace Combination Tuple: SCT0)**.** *Given a data vector $v \in \mathbb{R}^d$ and a dictionary $\mathcal{B} \in \mathbb{R}^{d \times n}$, suppose $w = \mathrm{SC}_{BP}(\lambda; v)$ is the k-sparse code generated by a LASSO solver for v and regularization λ. Further, let $\mathcal{I} = \{1, 2, \cdots, n\}$ be the set of the first n natural numbers. We define the mapping $h : \mathbb{R}^n \rightarrow I$, ($I \subset \mathcal{I},\ |I| = k$) as a* Subspace Combination Tuple *of w, if $h(w) = \{i_1, i_2, \cdots, i_k\}$, where i_j represents the index of the dimensions of w, such that*

1. *$i_j \in h(w)$, iff $w_{i_j} \neq 0,\ \forall i_j \in \mathcal{I}$, and*
2. *$h(w)$ is a well-ordered set with the indices arranged in* monotonically increasing order.

We use the notation $h_{(BP;\lambda)}(v)$ to denote the composition $h \circ \mathrm{SC}_{BP}(\lambda; v)$ and represents the SCT associated with v for a regularization λ using the Basis Pursuit algorithm.

The above definition means that the indices of the active support in the tuple are arranged in the ascending order. To make our representation clear, let us take a simple example.

Example 1. *Assume that for a data vector $v \in \mathbb{R}^{100}$, we have a w that is 3-sparse in some $\mathcal{B}$, where $w \in \mathbb{R}^{1000}$ is produced by some sparse coding algorithm that uses a regularization λ. Let us further assume that the support set of w is given by $\{w_{25}, w_{49}, w_{972}\}$, where each w_i corresponds to a non-zero dimension in w. Then, we have $h_{(BP;\lambda)}(v) = \{25, 49, 972\}$, where SCT definition insists that the indices form an increasing sequence.*

This simple representation will be used in the following chapters, although in Chapter 7 we will introduce other variants of SCT that take advantage of properties of the LARS algorithm to produce robust hash codes.

The following theorem establishes the connection between between SCTs of two data points and their angular separation. This theorem will be useful in our subsequent analysis for establishing a connection between our scheme and the standard LSH technique. Note that, for two well-ordered sets T_1 and T_2, we will use the notation $T_1 = T_2$ to mean component wise equality between the two sets. For example, if $T_1 = \{i_1, i_2, i_3\}$ and $T_2 = \{j_1, j_2, j_3\}$, then $T_1 = T_2$ implies $|T_1| = |T_2|$ and $i_k = j_k$ for $k \in \{1, 2, 3\}$.

Theorem 21. *For any two zero mean unit norm data vectors v_1 and v_2, let $w_i = \mathrm{SC}_{CBP}(\delta; v_i)$ and $T_i = h_{(CBP;\delta)}(v_i)$ for $i \in \{1, 2\}$ be the respective k dimensional SCTs using a dictionary $\mathcal{B}$. Suppose $\|v_1 - v_2\|_2 \leq 2\delta$, then for a probability density defined on a Lebesgue measure given by the volume of overlap of the two hypershperes with centroids v_i, loci $x_i = \mathcal{B}w_i$ and radii δ, $P(|T_1 \cap T_2| = k)$* monotonically increases *with decreasing $\widehat{v_1, v_2}$.*

Proof. Since the v_i is zero mean unit norm, it is easy to see that $\widehat{v_1, v_2} \propto \|v_1 - v_2\|_2$. As the centroids v_i and the radii δ of the two hyperspheres are fixed, they must have a non-zero intersection when $\|v_1 - v_2\|_2 \leq 2\delta$. Using the fact that the volume of intersection of two hyperspheres increases monotonically with decreasing centroid distances (see [Gromov, 1987][Theorem 1.A]), we have $P(|T_1 \cap T_2| = k) \propto |T_1 \cap T_2|/|T_1 \cup T_2| \propto 1/\|v_1 - v_2\|_2 \propto 1/\widehat{v_1, v_2}$. □

Corollary 22. *Let v_1 and v_2 be two unit data vectors and their respective SCTs be T_1 and T_2. Then*

$$\lim_{\beta \to 0} \widehat{v_1, v_2} \leq \beta \Rightarrow P(T_1 = T_2) \to 1. \tag{5.14}$$

5.3.2 ANN via SCTs

From Theorems 20 and 21, an approximate nearest neighbor scheme can be designed as described in Algorithm 2.

Algorithm 2 SCT-Hashing

Require: $\mathcal{B}$, $\mathcal{D}$,λ, $H\{$a hash table indexed by SCT$\}$

1: $\forall v \in \mathcal{D}$, compute $T = h_{(BP;\lambda)}(v)$.
2: Assign a location for the sparse code $w = \mathrm{SC}_{\mathrm{BP}}(\lambda; v)$ in the hash bucket $H(T)$ associated with T.
3: Given a query q, compute $T_q = h_{(BP;\lambda)}(q)$, and find the ANN of q

$$w^* = \underset{\forall \alpha \in H(T_q)}{\operatorname{argmin}} \|w - w_q\|_2, \tag{5.15}$$

where $w_q = \mathrm{SC}_{\mathrm{BP}}(\lambda; q)$.
4: **return** $v^* \in \mathcal{D}$ associated with w^*.

The SCT hashing scheme alone does not guarantee the retrieval of an NN candidate. Recall that the hashing is purely based on angles, and the NN recovery is based on Euclidean distances, such that two points can have equal angles but large Euclidean separation. Fortunately, when combined with Theorem 20, we can prune points that are large in the Euclidean distance, thus producing the NNs in a given ϵ-ball. In the following subsections, we will establish the connections between SCT-hashing and the LSH framework we introduced in Section 2.2.1.

5.3.3 Connections to LSH

For now, let us assume that our SCTs are one-dimensional. We will establish in Theorem 23 a connection between the angles that two data vectors subtend against each other with respect to the origin and matching their SCTs. This idea will be extended to k dimensional SCTs in Theorem 24.

Theorem 23. *Let $v_1, v_2 \in \mathbb{R}^d$ be two zero-mean unit norm data vectors and let $T_1 = h_{(BP;\lambda_1)}(v)$, $T_2 = h_{(BP;\lambda_2)}(v)$ be the respective SCTs and that $|T_1| = |T_2| = 1$, that is, the SCTs are one-dimensional. Let $\theta = \min_{\forall b_i, b_j \in \mathcal{B}, i \neq j} \widehat{b_i, b_j}$ be the smallest angle between bases in $\mathcal{B}$. If $\widehat{v_1, v_2} \leq \theta$, then $Prob(T_1 = T_2) \geq 1/2$ for some $\lambda_1, \lambda_2 > 0$.*

Proof. We have

$$P(T_1 = T_2) = 1 - P(T_1 \neq T_2). \tag{5.16}$$

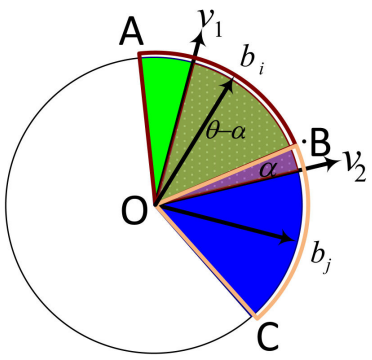


Figure 5.2: A simplified schema illustrating Theorem 23. O is the center of the unit ball, b_i and b_j are two bases under consideration. AB and BC are sectors into which if the two data points v_1 and v_2 ($\widehat{v_1, v_2} \leq \theta$) fall, then they will have the same SCT. To determine the probability of the data points that have different SCTs, we rotate a sector of size θ (represented by $v_1\widehat{Ov_2}$) from A to C; an intermediate position of this arc is shown that makes an apex angle of α in region BC and $\theta - \alpha$ in AB.

With reference to the illustrative 2D case as shown in Figure 5.2, let b_i and b_j be two adjacent basis vectors such that $\widehat{b_i, b_j} \leq \theta$ for some b_i and b_j (that is, v_1 and v_2 lie in the interior of the sector defined by v_1Ov_2). The arcs $\widehat{AB}$ and $\widehat{BC}$ represent regions around the two basis b_i and b_j respectively such that if both v_1 and v_2 fall in either of these regions, they will have the same SCT. Now, to generate the loci of all the SCTs that do not match, we can have the sector generated by v_1Ov_2 as shown in the figure rotated from A to C such that at some point, we have $\widehat{v_1, B} = \theta - \alpha$ and $\widehat{B, v_2} = \alpha$, for some $\alpha \leq \theta$. Then,

$$P(T_1 \neq T_2) = Prob(v_1 \in S(\theta - \alpha))Prob(v_2 \in S\alpha))+$$
$$Prob(v_2 \in S(\theta - \alpha))Prob(v_1 \in S(\alpha)) \quad (5.17)$$
$$= 2\frac{S_A(\theta - \alpha)S_A(\alpha)}{S_A(\theta)^2}, \quad (5.18)$$

where $S(\alpha)$ is the surface of a hypershpercial cap with an apex angle of α and $S_A(\alpha)$ represents the surface area of the unit ball.

Now, as area is an increasing function of the apex angle and since the total area of the sub-cap that we are interested in is $S_A(\theta)$, we can rewrite (5.18) as:

$$f(\alpha) = \frac{S_A(\alpha)\left(S_A(\theta) - S_A(\alpha)\right)}{S_A(\theta)^2}. \quad (5.19)$$

From [Li, 2011], we have

$$S_A(\phi) = \frac{1}{2} A_n I_{\sin^2(\phi)}(\frac{d-1}{2}, \frac{1}{2}), \tag{5.20}$$

where ϕ is the apex angle of the cone, A_n is the surface area of an n-dimensional unit ball and $I_x(a,b)$ is the regularized incomplete beta function given by:

$$I_x(a,b) = \frac{1}{C} \int_0^x t^{a-1}(1-t)^{b-1} dt \text{ for a constant } C. \tag{5.21}$$

To obtain an upper bound to (5.19), we can maximize the right-hand side of it with respect to α. Substituting (5.20) in (5.19), applying the *generalized Leibniz rule* for integrating under a differential sign and equating to zero, we get

$$S_A(\alpha) = \frac{S_A(\theta)}{2} \tag{5.22}$$

which implies $\alpha = \theta/2$. Thus we have:

$$P(T_1 \neq T_2) \leq 2\frac{S_A(\theta/2)^2}{S_A(\theta)^2} = 1/2, \tag{5.23}$$

which implies $P(T_1 = T_2) \geq 1/2$. □

Now, let us consider the general case of matching k dimensional SCTs.

Theorem 24. *Let $v_1, v_2 \in \mathbb{R}^d$ be two zero-mean unit norm data vectors that are k-sparse in a dictionary $\mathcal{B} \in \mathbb{R}^{d \times n}$ ($n \gg d$). If $T_i = h_{(BP;\lambda_i)}(v_i)$ for $i \in \{1,2\}$ be the respective k dimensional SCTs, then for $1 \leq k \leq d$, we have $P(T_1 = T_2) \geq 1/2$, if*

$$\widehat{v_1, v_2} \leq \theta, \text{ such that } S_A(\theta) = \frac{n\pi^{n/2}}{\Gamma(\frac{n}{2}+1)2^k\binom{n}{k}}, \tag{5.24}$$

where θ represents the apex angle of a hyperspherical cap and $S_A(\theta)$ represents the surface area of this cap. Further, we assume that n is even and $\lambda_i > 0$.

Proof. For a dictionary with n bases, we have $\binom{n}{k}$ ways to select k bases. Further, for each such selection, say $\mathcal{A} = \{s_{i1}b_{i1}, s_{i2}b_{i2}, \cdots, s_{ik}b_{ik}\}$ for $s_{ij} \in \{-1, +1\}$, we have 2^k unique equiangular directions that can be generated. Now, let us assume that instead of an n bases dictionary $\mathcal{B}$, we have alternatively a large dictionary $\mathcal{B}'$ with $r = 2^k\binom{n}{k}$ directions we just produced. Now for the data vector v_i, a sparse coding algorithm (such as LARS) on $\mathcal{B}'$ will select the basis

corresponding to the combined subspace produced by all the bases in T_i. Substituting $\mathcal{B}'$ in Theorem 23, we have $P(T_1 = T_2) \geq 1/2$ if $\widehat{v_1, v_2} \leq \theta$. Now, θ is upper bounded by the radial angle between the centroids of adjacent partitions of the unit ball surface, which is divided into r partitions of equal surface area. Substituting the formula for the surface area of an n-ball for even n, we have the required bound. □

Combining Theorems 21 and 24 together with the LSH scheme introduced in Proposition 5 (See Section 2.2.1), the following theorem formally shows that our hashing scheme adheres to an LSH requirements based on angles.

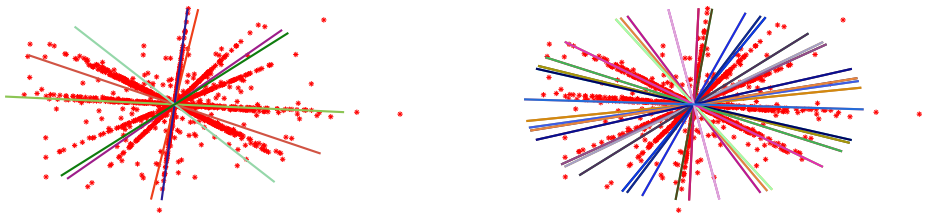
Theorem 25. *Let $v_i \in \mathbb{R}^d, i \in \{1, 2\}$ be two zero mean data points, k-sparse in a dictionary $\mathcal{B} \in \mathbb{R}^{d \times n}$. Then, if $\widehat{v_1, v_2} < \theta_1$ (as specified by Theorem 24), for SCTs $T_i = h_{(BP;\lambda_i)}(v_i)$, we have $P(T_1 = T_2) > p_1$. Further, from Theorem 21, we have $P(T_1 = T_2) < p_2$ when $\widehat{v_1, v_2} > \theta_2$ for a suitable λ_i. Since $\theta_1 < \theta_2$ and $p_2 > p_1$, we have that our scheme adheres to an LSH scheme. The (r_1, r_2)-ball constraints will be accommodated by the subsequent linear scan after hashing as proposed in Theorem 20.*

5.4 Space Partitioning

5.4.1 Data/Space Partitioning

Now that, we have a good theoretical understanding of the working of our LSH scheme, in this section we provide the reader a geometrical intuition via visualization on the effects of hashing generated by sparse coding. Figure 5.3 illustrates the partitioning introduced by the bases as the support size increases. The red points in the figure show 2D data points for which a dictionary of size 2×7 is learned. In the figure, all the bases are shown to pass through the origin. This is because we assume the data (and thus the learned dictionary) are zero-mean. A 1-sparse reconstruction of this data in the learned dictionary is essentially each basis itself (as seen from the figure), while higher support sizes introduce an increasing number of basis combinations (in equiangular directions). For example, there are 7 bases in a 1-sparse representation as shown in Figure 5.3(a), while 21 bases for 2-sparse as shown in Figure 5.3(b). An SCT represents one such line and all the data points angularly proximal to the line are mapped to one hash bucket. It is easy to see that the data points vary in radial distances. We use a linear scan or use a k-d tree to find the NN from these data subsets. Note that since we need to store only the sparse

codes associated with the data points, our storage needs are minimal. More to it, our search in a hash bucket is over extremely low dimensional spaces (corresponding to k-sparse codes), as a result we need not worry about the curse of dimensionality any more.



(a) 1-sparse (b) 2-sparse

Figure 5.3: Data space partitioning induced by sparse coding. The red points represent 2D data points and the lines denote the dictionary bases from a 2×7 learned dictionary. Figure 5.3(a) shows the directions of the dictionary bases (corresponding to a 1-sparse representation), Figure 5.3(b) shows equiangular directions corresponding to all possible 2 basis combinations from the dictionary (2-sparse).

5.4.2 Relation to Other Hashing Algorithms

It will be worth taking a closer look at how our algorithm differs from other state-of-the-art hashing algorithms. Most of the hashing algorithms in use currently (examples include Hamming Embedding [Jegou et al., 2008], spectral hashing [Weiss et al., 2009], etc.) learn linear subspaces on to which the data is projected, the sign of this projection is then utilized to a create bit vector hash code. We also use the same idea through using the basis dictionary, except that we use a large number of such linear hash functions, and dynamically choose which hash function to be used depending on the nature of the data. The power of our hashing algorithm comes fundamentally from the adaptivity introduced by this selection, through sparse coding.

5.5 Advantages of ANN via SCT

In this section, we will bring out the theoretical advantages of using sparse coding for ANN against other methods.

5.5.1 Accuracy

Traditional LSH schemes such as E2LSH [Datar et al., 2004] requires the original data descriptors to be stored in the hash bucket for resolving the hash table collisions. On the other hand, fast LSH schemes such as spectral hashing discards the original data vectors for the sake of lower memory footprint there by compromising on the accuracy of the NN. Our sparse coding algorithm is a good mix of both these extremes. The SCT representation provides a good hash code, and the active elements of the sparse coefficient vector associated with a data vector provides a compact approximate representation of the original data vector. This leads to greater accuracy as our experiments will demonstrate in later chapters.

5.5.2 Memory Footprint

Let us assume that we use a dictionary $\mathcal{B} \in \mathbb{R}^{d \times n}$, such that each dictionary atom is indexed by a unique integer. Assuming that each basis is equally likely, we need $\log n$ bits to represent a dictionary index. Further, let us assume that each data vector is projected onto a k-sparse coefficient vector (i.e. have no more than k non-zero sparse coefficients), each coefficient taking p bits. Then in total, we require $k \log n$ bits for indexing the hash table and a further kp bits to store the sparse coefficients. Thus the total space complexity is $k(p + \log n)$ bits, which is essentially $\mathcal{O}(1)$, as all the quantities we use are constants. In a real world system, if we use a 2048 bases dictionary, we will need 55 bits for hashing and 215 bits for storing the coefficients, assuming a 5-sparse code. Generally, all the dictionary atoms are not equally likely; as a result we can reduce the number of bits for hashing further (using standard compression schemes such as Huffman coding; although we do not consider this direction in this thesis).

5.5.3 Sparse Coding Complexity

Let us assume that we use a maximum of k iterations of the LARS algorithm for sparse coding a data point $v \in \mathbb{R}^d$ using a dictionary $\mathcal{B} \in \mathbb{R}^{d \times n}$. Further, assuming we use Cholesky factorization [Golub and Van Loan, 1996] and updating for computing the least angle direction in each iteration of the LARS algorithm; then each direction computation takes $\mathcal{O}(j^2)$ for the jth LARS step, leading to $\mathcal{O}(k^3)$ time for k steps. To compute the maximum correlated dictionary atom, for k steps we require $nd + (n-1)d + (n-2)d + ... + (n-k)d) = \mathcal{O}(ndk)$ computations. So total computational complexity is $\mathcal{O}(k^3 + ndk)$. As smaller codes ($k \leq 5$) are sufficient

for our algorithm, the computation is dominated by the component $\mathcal{O}(ndk)$, which is the cost of making k inner products with the entire $\mathcal{B}$. Once the SCT is constructed, hashing is $O(1)$ constant time complexity, although we might need to do linear search in the corresponding hash bucket in case there are other data points mapped to that bucket.

5.5.4 Query Complexity

Once we have mapped the query data points into a hash bucket using the SCT hashing, the next challenge is to retrieve the ANNs from data points mapped to this hash bucket. Recall that SCT achieves radial LSH, while the data points can vary significantly in magnitude along radial directions. That is, a data point v and a data point αv, for a scalar α will map to the same hash bucket. We need to use Theorem 20 to search for the ANN from the hash bucket.

The organization of the sparse codes at this stage is thus extremely important for efficient query-retrieval. In our experiments we use a simple linked list organization; as we will see later that the average hash bucket length is often found to be small. Assuming m items in a hash bucket, the query complexity is thus $\mathcal{O}(m)$ for linear search. At this point, we should like to highlight the importance of sparse codes as being extremely low dimensional. For example, typically the codes used in our experiments are of length 5-10; as a result; they are robust to the curse of dimensionality. We can use data structures such as k-d trees in a hash bucket leading to a query complexity of $\mathcal{O}(\log m)$ or even use a standard E2LSH scheme to reduce to an asymptotic complexity approaching $\mathcal{O}(1)$.

5.5.5 Hashing Efficiency

Standard LSH algorithms such as KLSH [Kulis and Grauman, 2009], E2LSH, etc. assume a fixed set of hyperplanes onto which the data is projected. These approaches use k hyperplanes to produce a k-bit hash code, such that the representational power of these hash codes is 2^k. On the other hand, in the case of SCT based approach, recall that there is a large set of basis vectors (due to the high dimensionality of the data and the overcompleteness of the dictionary) of which only a very few need to be active for sparse coding a data vector. Referring to (5.3), assume that the data vectors v are k-sparse in the dictionary and the dictionary has n bases. If each basis is equally likely to be in the active set, then we have $C = 2^k \binom{n}{k}$ unique active basis combinations possible. Sparse coding allows optimized selection of the random hyperplanes

from the set of available random hyperplanes thus allowing a greater representational power than the traditional algorithms. To see this idea in a practcal setting, let us assume we use a dictionary of size 2048, just having an active set of size 10 leads to approximately 10^{29} unique combinations. This is clear from the illustration in 5.3.

5.5.6 Hash Table Collision Frequency

Collision of the data points on to the same hash bucket is a property that decides the efficiency of the hashing algorithm. For example, if a majority of the data points are mapped to the same hash bucket, then the NN retrieval will not be significantly better than a linear search on the dataset. Let us assume that the data $\mathcal{D}$ is uniformly distributed into the hash buckets by the dictionary $\mathcal{B} \in \mathbb{R}^{d \times n}$. If the cardinality of the dataset is $|\mathcal{D}|$, then this uniform distribution implies that on an average $\frac{|\mathcal{D}|}{2^k \binom{n}{k}}$ are mapped to the same hash bucket. For moderate values of k and high values of n (as we see in our experiments), this leads to lower collision frequency and thus better hashing efficiency.

5.5.7 Scalability

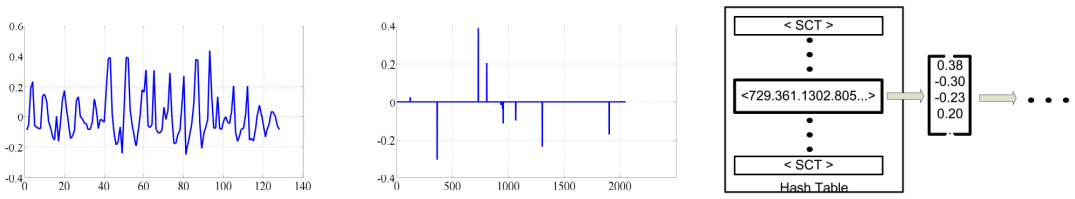
It is easy to see from the above analysis that with appropriate selection of the parameters of the model (such as the regularization value and training the dictionary to have lower coherence), the resultant SCT algorithm is scalable to large datasets.

5.6 Hashing Using SCT

Now that we understand the advantages of sparse coding, let us use it in a real-world application. This will also help bring out the issues that sparse coding brings. Following chapters will address these issues via novel variants of sparse coding and dictionary learning. But, before we proceed with our experiments, we would like recall and illustrate schematically our *Subspace Combination Tuple*.

Figure 5.4 illustrates of the idea for a typical zero-mean normalized SIFT descriptor. Each column of the dictionary is identified by its location index in the dictionary, and thus a hash key is a set of integers, which can be encoded as a bit stream. To tackle collisions in the hash buckets, the colliding descriptors are organized using an efficient data structure, such as a linked

list (or a k-d tree for efficiency).



(a) (b) (c)

Figure 5.4: (a) Normalized SIFT descriptor, (b) the sparse representation of the SIFT descriptor in (a) using a set of 2048 bases dictionary, (c) an inverted file organization, the hash key is the SCT and the hash bucket holds the active coefficients of the bases in the order of their entry in the SCT, and the descriptors are arranged in a linked list in case of bucket collisions.

5.7 Experiments

In this section, we illustrate, via a real-world experiment, how the SCT representation improves the NN retrieval performance. The application we decided is that of camera motion estimation. This is a fundamental application in computer vision; the problem is to find point correspondences between successive images from a camera moving in a static environment and estimate the motion parameters of this camera (that is, its translational and rotational velocities) from these point correspondences. Such a motion estimation problem arises in several active areas of computer vision, including 3D reconstruction, and robot navigation.

To conduct this experiment, we used a miniature RC helicopter mounted with a small camera. Successive frames from this camera are transmitted to a computer on which algorithms are run that estimates the motion parameters of the helicopter. A standard way to estimate the motion parameters is to find point correspondences between subsequent video frames; a set of such corresponding points are then used to solve a nonlinear equation (called the Fundamental Matrix equation) [Hartley and Zisserman, 2004], which is subsequently decomposed into the camera motion parameters. In mathematical terminology; suppose $X = \{x_1, x_2, \cdots, x_n\}$ represents n points in one image (each $x_i \in \mathbb{R}^3$ represents the 2D image location in homogeneous coordinates) and let $X' = \{x'_1, x'_2, \cdots, x'_n\}$ denote the corresponding n points in the successive image (again, each x'_i is in homogeneous coordinates). Then, it can be shown that there exists a

3×3 matrix F, called the Fundamental matrix, which satisfies the following condition:

$$x_i^T F x_i' = 0 \tag{5.25}$$

for each $x_i \in X$ and each $x_i' \in X'$ respectively. We can use a subset of X and X' to solve for F. Further, once we have F, it is shown in [Hartley and Zisserman, 2004] that F can be decomposed into $F = R[T]$, where R is the matrix of camera rotation and T represents the unit camera translation vector (in 3D). The notation $[T]$ refers to the skew-symmetric matrix created using the dimensions of T (See [Cherian et al., 2010] for details).

Now, a basic problem to be addressed here for reliable motion estimation is that of the selection of appropriate image point descriptors. It was found that the onboard-camera used in our experiment produced images that were often corrupted with noise from vibrations of the vehicle and radio interference. Also, other problems such as dramatic changes in the illumination of the environment due to uneven indoor lighting, white noise from the camera sensor, and extreme motion blur made motion estimation even more difficult. Thus, SIFT descriptors were found to provide the best results for the point correspondences. Since computational time is a critical factor for the real-time performance, we decided to use our SCT model to find these point correspondences. Once the correspondence problem is taken care off, we use these correspondences in the algorithm we described above (for computing the F and the subsequent matrix decomposition) to estimate the helicopter motion parameters. The accuracy of the point correspondences (using the SCT model) can thus be validated by comparing the estimated translation and rotational velocities against a ground truth motion model found to sub-pixel level accuracy using a high-end Vicon[1] camera system. This experiment is schematically illustrated in Figure 5.5.

The experiment used the SCT0 representation we introduced earlier (recall that SCT0 uses the sorted indices of the active bases in the learned dictionary). The vision algorithms were implemented partly in C++ and partly in Matlab. Every frame from the camera is matched against a frame three time steps away. The frames are of size 640x480, which is rescaled to 160x120 for reducing the motion blur and improving the speed with which the SIFT descriptors are generated. The frames are applied with additional deblurring filters and low pass filters to reduce the effects of radio interference noise. Finally they are used for finding the point correspondences using the SCT representation using a hash table. Figure 5.6 shows the estimated and actual translation velocities of the camera for two different motions, namely (i) controlled

[1] www.vicon.com

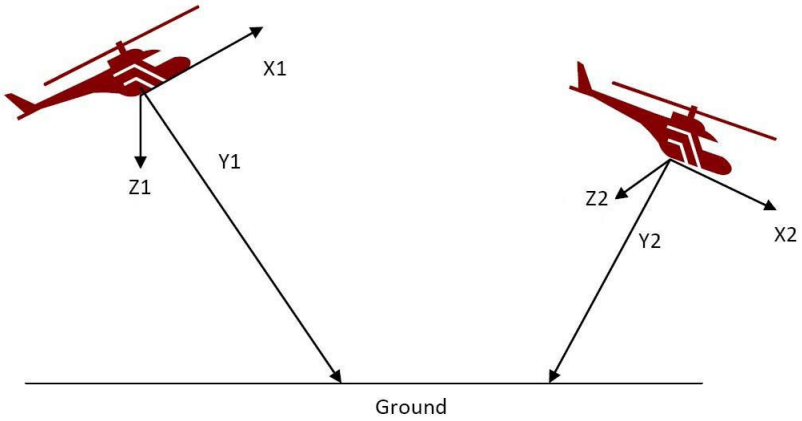


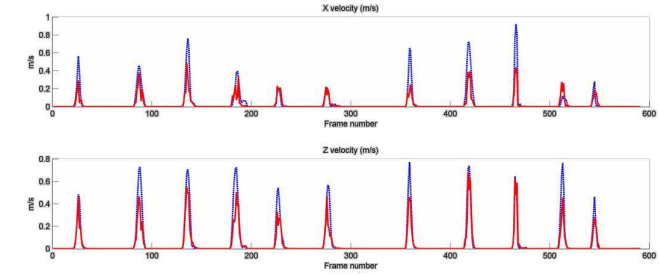
Figure 5.5: Illustrates the location (X1, Y1, Z1) and (X2, Y2, Z2) of the helicopter at two time instants along with the orientation. The altimeter reads Y1 and Y2 at the two instants.

motion in which the helicopter was moved with hand along a single motion axis, and (ii) remote controlled flight sessions. The plots in Figure 5.6 show that the estimated velocity matches with the actual velocity to a reasonable extend. A comparison of the estimated rotational velocity is provided in Figure 5.7. Table 5.1 shows the performance improvement using the SCT representation for matching compared to a k-d tree based SIFT NN retrieval. A summary of the experiment is provided in Table 5.2.

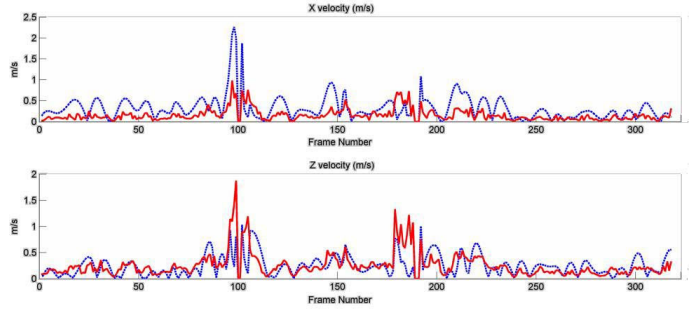
$descriptor$	Max $\#matches/frame$	Min $time$	Max $time$	$Mean$ $time$	$total$ $time$
KDTree	80	0.19	1.85	0.71	479.73
SCT	29	0.19	2.09	0.55	369.8

Table 5.1: A comparison of SIFT matching with k-d tree against the SCT method. Performance is computed on 675 frame pairs (time in seconds). Note that the described time includes the time to generate the SIFT descriptors.

It is worth noting from Table 5.1 that even though the NN speed has improved against k-d tree, the number of matches per frame pair is substantially low. This is primarily due to the perturbations that SIFT descriptors corresponding to the same keypoints undergo, that results in different SCTs. Let us understand this problem in sufficient detail next, as it forms the content for the subsequent chapters of this thesis.



(a)



(b)

Figure 5.6: It shows the absolute estimated velocity (red) and the absolute true velocity (blue dotted) in the X and Z axes respectively in a regular flight session. The x-axis is the frame number and the y-axis is in meters/seconds.

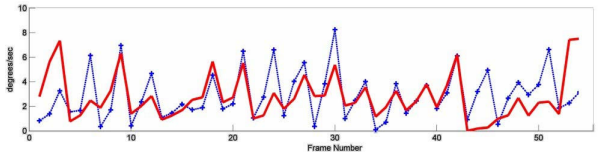


Figure 5.7: The plot shows the norm of estimated angular velocity vector $[\omega_x, \omega_y, \omega_z]^T$ in degrees against the angular velocity found from the Vicon camera system. The x-axis is the frame number and the y-axis is in degrees.

$Direction$	$Min\ abs\ Error$	$Max\ abs\ Error$	$Mean\ abs\ Error$	$Std.\ Dev.$
X (m/s)	0.0	2.62	0.21	0.07
Z (m/s)	0.0	1.71	0.15	0.03
E (deg)	0.24	6.77	2.79	4.06

Table 5.2: Results from the UAV experiment showing the error in estimation using the sparse correspondence model.

5.8 Caveats

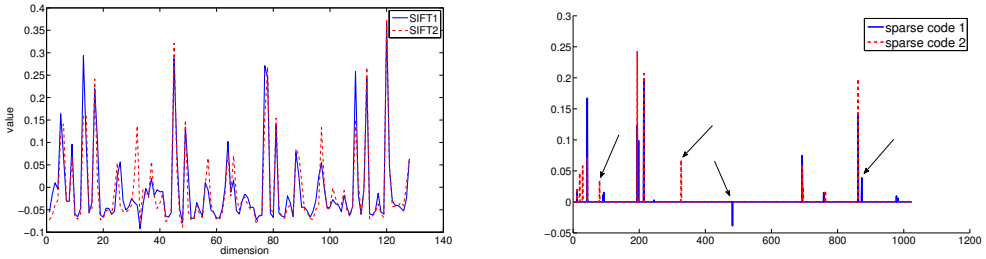
Improving NN accuracy using sparse coding via the SCT based approach described above is fundamentally based on increasing the hash collision probability. This probability is proportional to increasing the number of subspaces overlapping between a query sparse code and the database sparse codes. To this end, we will state a metric on which we will measure the performance of sparse coding based NN; *Basis Overlap* Metric. We define this metric as follows:

Definition 10. *Suppose v and q are two data vectors, then we define* Basis Overlap *(BO) between v and q for a given λ as:*

$$BO(v,q) = \frac{|h_{(BP;\lambda)}(v) \cap h_{(BP;\lambda)}(q)|}{\max\{|h_{(BP;\lambda)}(v)|, |h_{(BP;\lambda q)}|\}}, \tag{5.26}$$

where the notation $|\ |$ refers to the *set cardinality*. This metric is closely related to *Jaccard Distance* popular in statistics literature [Jaccard, 1901].

Now, let us come back to the caveats of NN retrieval using sparse coding. Although sparse coding seems like an attractive idea, it poses an important problem when working with real data. Recall that Theorems 23 and 24 make a fundamental assumption that we know the regularization constants of the sparse coding formulation for the data vectors at which they will have the same SCT. That is, these theorems assume that we have a good estimate of the regularization constant λ to sparse code both the query point and the database points, so that the hash codes generated $h_{(BP;\lambda)}(v)$ for a data point v will be mapped to its NN in the hash table. Unfortunately, finding this regularization value is not trivial in many situations. As a result the collision probability might be low. This problem is fundamental to sparse coding to be useful as a competent NN algorithm and forms the content for the next few chapters of this thesis. Figure 5.8 illustrates this problem for two neighboring SIFT descriptors and their respective sparse representations.



(a) (b)

Figure 5.8: An illustration of the problem of generating hash codes from two closely related SIFT descriptors. (a) shows two SIFT descriptors of the same world point but from two different camera views, (b) shows the sparse representation of the descriptors in (a) using a 1024 dimensional dictionary. The arrows in (b) shows places where the non-zero indices mismatch.

Our approaches to improve the performance of NN via sparse coding is to maximize the BO between database and the query vectors by adjusting the learning of the dictionary, for a fixed regularization. To this end, in Chapter 6, we analyze the statistical distribution of data points such that neighboring points belong to the same dictionary subspace. In Section 6.6, we take a robust optimization perspective to this problem. Our main idea is to estimate the maximum deviation (or perturbation) that a data point takes from its nearest neighbor, and learn the dictionary in such a way that this robustness requirement is conformed. In Chapter 7, we take a different approach to circumvent this issue using an understanding gained by the properties of the LARS regularization path.

Chapter 6

Nearest Neighbors via Robust Dictionary Learning

In the earlier chapter, we introduced the *Subspace Combination Tuple* representation of sparse coded data that helps in indexing a hash table for fast ANN retrieval. A fundamental problem when using this setup for real-world data is that noise and perturbations in the data often resulted in data points that are neighbors in the input space to occupy different subspaces in the dictionary, as a result of which SCT hashing lead to lower retrieval probability. In this chapter, we will look at this issue closely, and provide hashing algorithms that bypass this problem by learning a dictionary that is resistant to such perturbations. To this end, we propose two different schemes. In Section 6.1, we establish an analogy between the robust dictionary learning problem and the traditional signal denoising problem, where the noise follows a heterogeneous mixture of Gaussian and sparse Laplacian distributions. In Section 6.6, we take a different perspective at the robust dictionary learning problem, by looking at the perturbations as instabilities in the data itself. Ignoring any distributional assumptions on these instabilities, we approach the problem from a robust optimization perspective and provide a dictionary learning formulation that is robust to worst case perturbations. While, the methods in this chapter are towards a robust dictionary learning strategy, in Chapter 7, we put the onus on the sparse coding step and propose an efficient algorithm for ANN utilizing the properties of the solution path of the LARS-LASSO algorithm.

6.1 Robust Nearest Neighbors as Signal Denoising

In this section, we will take a signal processing approach to address the problem of robust dictionary learning, which we call *Sparse Denoising*. Our basic idea is as follows: assume that each data point is encoded by a feature descriptor $f \in \mathbb{R}^d$, and a few noisy variants of it f_i ($1 \leq i \leq k$), such that $f_i = f + \epsilon_i$, where each ϵ_i represents noise. Suppose now that f has a *sparse* representation in an overcomplete dictionary $\mathcal{B}$, i.e., there is a p-sparse vector x such that $\mathcal{B}x = f$. So, if $\mathcal{J}_f = \langle j_1, j_2, \cdots, j_p \rangle$ (where $p \ll d$) represents an SCT that we introduced in the last chapter, then $\mathcal{J}$ may be viewed as an encoding of the subspace in which f lives, and thus it can be used as a new descriptor for f. This idea has a useful consequence: if we remove ϵ_i from each of the f_i's, then all of them will have the *same* tuple descriptor. Now, since the tuple $\mathcal{J}$ can be used to index into a hash table, the NN (or rather k-NN) operation essentially reduces to a denoising operation. But to successfully use $\mathcal{J}$ for hashing, we must first model the underlying noise present in the input descriptors. Otherwise, some of the basis vectors will end up reconstructing noise, thereby corrupting the set $\mathcal{J}$, and defaulting the hashing.

In the experiments in Chapter 5, the SIFT descriptors were assumed to be noise free. This assumption is unrealistic, and a more careful analysis reveals that matching SIFT descriptors are approximately distorted Gaussian plus sparse noise. This observation is illustrated in Figure 6.1, which plots the difference between two SIFT descriptors corresponding to the same keypoint from two different images. This experiment was conducted on the SIFT descriptors computed on the dataset used in [Mikolajczyk and Schmid, 2005], which consists of images undergoing various distortions and noise corruptions. Approximate SIFT NNs were extracted using a kdtree that lies within an $\epsilon = 0.7$ ball for a given query descriptor. In Figure 6.1(b), we plot a normalized histogram of difference between these corresponding SIFT NNs. There were approximately 20K matching SIFT descriptors from all the distortion categories described in [Mikolajczyk and Schmid, 2005]. As is clear from the figure, most of the SIFT descriptors match exactly, while those inexact descriptor pairs show low distances predominantly, while the tail of the distribution shows the contribution of sparse distance differences. To show explicitly the significance of the tail of this distribution, we removed the *exact* matching components from the histogram and plotted the absoluted distances between 0.1 and 0.4 in Figure 6.1(c).

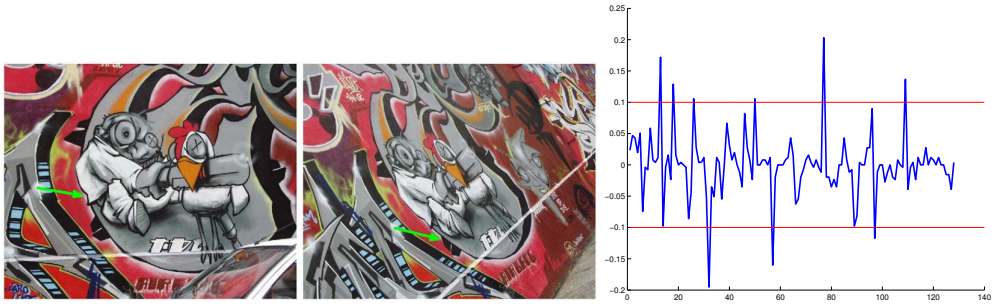
The plots indicate that while variations in most dimensions are small, some of the dimensions display high variations. These variations can be described well by a Gaussian plus sparse

(Laplace) noise model. In the following, we propose such a model for denoising SIFT descriptors by embedding the denoising task within a dictionary learning framework. But before we proceed with proposing our Laplacian denoising framework, to put our work in perspective we would like to present earlier work in using dictionary learning for image denoising applications.

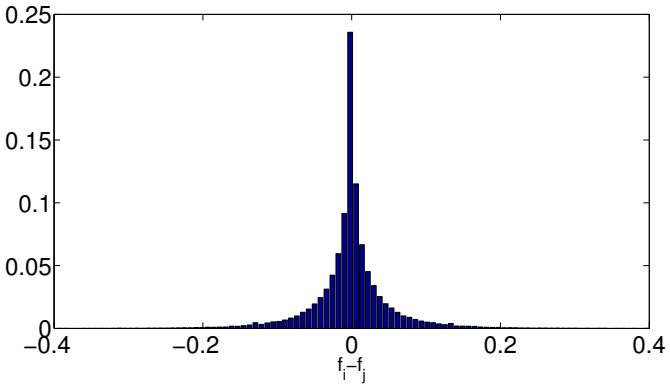
6.2 Related Work

Image and signal denoising has been one of the most important and successful applications of the dictionary learning and sparse coding framework. So providing a extensive literature review of this topic is outside the scope of this chapter. We refer the reader to the useful surveys [Thangavel et al., 2009; Motwani et al., 2004] for denoising; for dictionary learning, see [Murray and Kreutz-Delgado, 2004; Elad and Aharon, 2006; Mairal et al., 2010] and references therein. Below we recap some immediately relevant work and concepts.

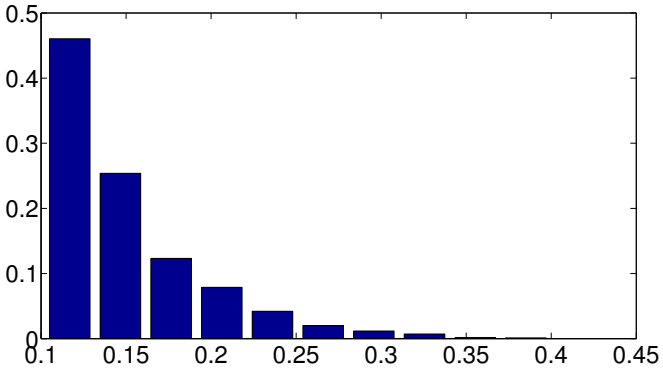
Laplacian plus white Gaussian noise models are investigated in [Selesnick, 2008], but under the restrictive assumption that the sparse basis is known. This method use underlying properties of the signal to separate it from the noise, and therefore do not apply to the signals we work with in this chapter. A paper closer in spirit to our approach is [Elad and Aharon, 2006], where an image denoising framework based on dictionary learning is suggested. But [Elad and Aharon, 2006] assumes the signal to have only zero-mean Gaussian noise, a limitation when additional Laplacian noise is present. A total variation based solution is suggested in [Le et al., 2007] for removing sparse Poisson noise from images. A successful approach via a scaled mixture of Gaussians for denoising is proposed in [Portilla et al., 2003]. Denoising in the wavelet domain is considered in [Zhang et al., 2008]. A similar Laplacian prior idea was suggested for *universal sparse coding* using dictionary learning in [Ramirez et al., 2009], with the significant difference that this paper adds the Laplacian prior on the sparse coefficients, while our objective is to describe sparsity of noise in the input space. Another paper that deals with a similar idea is [Xiao et al., 2011] that proposes a three stage algorithm based on dictionary learning for removing Gaussian and impulse noise. This algorithm first uses a median filter to remove the impulse noise from the data, later using the Gaussian denoising approach. This approach is difficult to be used in our application as it is not possible to characterize spike noise from a single data descriptor in our setting. Further, our work differs from this paper in that ours is a



(a)



(b)



(c)

Figure 6.1: Figure 6.1(a) illustrates the noise distribution of matching SIFT descriptors: (left) image pair shows two corresponding SIFT descriptors (shown in green), (right) shows a plot of the difference between the two corresponding SIFT descriptors. Figure 6.1(b) shows the histogram plots of differences between dimensions ($f_i - f_j$), for two corresponding SIFT descriptors f_i and f_j. The correspondence was defined as SIFT descriptors between subsequent pairs of frames in the [Mikolajczyk and Schmid, 2005] dataset for which the Euclidean distance was less than 0.7. Figure 6.1(c) was obtained by removing exact matches. The majority of the distances are small, but the sparse differences are statistically significant as described by the tail of the distribution.

one-step algorithm for denoising such a mixed noise signal.

6.3 Sparse Denoising

Assume that the observed signal is corrupted by independent zero-mean Gaussian and sparse Laplacian noise. Thus, denoting the true signal by the random variable X, the observed signal Y may be modeled as

$$Y = X + \epsilon_g + \epsilon_s, \tag{6.1}$$

where $\epsilon_g \sim \mathcal{N}(0, \sigma_1^2)$ is Gaussian noise, while $\epsilon_s \sim \mathcal{L}(0, \sigma_2)$ is symmetric Laplacian (sparse) noise.

For a random variable γ distributed according to a symmetric Laplacian distribution with location parameter μ and shape parameter σ, we have the following pdf:

$$\mathcal{L}(\gamma; \mu, \sigma) = \frac{\sigma}{2} \exp(-\sigma(\gamma - \mu)). \tag{6.2}$$

A sum of two random variables as in (6.1) results in the convolution of their respective probability density functions [Grinstead and Snell, 1997][Chapter 7]. A convolution of the Normal distribution with the Laplace distribution leads to the Normal-Laplace distribution [Reed, 2006] which has the following density function: for a parameter ξ, and for the underlying distributions $\mathcal{N}(0, \sigma_1)$ and $\mathcal{L}(0, \sigma_2)$ (symmetric Laplacian), we have:

$$p(\xi) = \Phi(\frac{\xi}{\sigma_1}) - \frac{\exp(-\frac{\xi^2}{\sigma_1^2})}{2} \left\{ \left[(1 - \Phi(\xi_1)) \exp(\xi_1^2) \right] - \left[(1 - \Phi(\xi_2)) \exp(\xi_2^2) \right] \right\} \tag{6.3}$$

where

$$\xi_1 = \frac{\xi}{\sigma_1} - \sigma_1 \sigma_2 \tag{6.4}$$

$$\xi_2 = \frac{\xi}{\sigma_1} + \sigma_1 \sigma_2 \tag{6.5}$$

$$\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} \exp(-t^2/2) dt. \tag{6.6}$$

It is not hard to see that $\Phi(z)$ is the cdf of $\mathcal{N}(0, 1)$, for which there is no closed form, making this route of denoising difficult to formulate.

To make our problem computationally tractable, we propose a relaxation of (6.1) by introducing an auxillary variable $Z = X + \epsilon_g$, with which we obtain the *sparse-noise* model:

$$Y = Z + \epsilon_s \qquad Z = X + \epsilon_g. \tag{6.7}$$

Figure 6.2 illustrates this idea schematically. The true signal X is assumed to be described by a sparse linear combination of an underlying dictionary, so that an instance x of the true-signal X satisfies

$$x \approx \mathcal{B}c, \tag{6.8}$$

where $\mathcal{B}$ denotes the underlying dictionary and c a sparse vector.

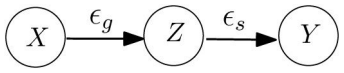


Figure 6.2: Modified noise model, where we introduce an auxiliary state variable Z that encapsulates the Gaussian noise ϵ_g, thus providing a tractable denoising model. X is the true data, while Z is X with Gaussian noise and Y is Z with sparse Laplacian noise.

Going by our approach described in earlier chapters, our aim here is to recover a dictionary $\mathcal{B}$, given just the set of observations $\{y_1, \ldots, y_N\}$, which we collect as the columns of the *observation matrix* Y. Assuming our sparse-noise model (6.7) and the linear relation (6.8), the dictionary learning task may be cast as

$$\begin{aligned} \min \;\; \psi(C, X, Z, \mathcal{B}) :=& \frac{1}{2}\sum\nolimits_{i=1}^{N} \|x_i - \mathcal{B}c_i\|_2^2 + \gamma_i\|c_i\|_1 + \lambda_i\|y_i - z_i\|_1 + \frac{1}{2}\beta_i\|z_i - x_i\|_2^2, \\ & \text{subject to} \quad C \in \mathcal{C}, X \in \mathcal{X}, Z \in \mathcal{Z}, \mathcal{B} \in D, \end{aligned} \tag{6.9}$$

where λ_i, β_i, and γ_i are (known) scalar parameters; $\mathcal{C}$, $\mathcal{X}$, $\mathcal{Z}$, and D are optional convex sets modeling additional constraints that may be desirable. The first two terms of ψ model the sparse relation (6.8); the third term models the sparse noise between y_i and z_i, while the last term models the Gaussian noise between z_i and x_i.

One way to optimize (6.9) is via block coordinate-descent, where one cycles through the blocks C, X, Z, and D. While such block steps might be easy, they can be computationally demanding and require book-keeping of the data points for the dictionary update. A potentially less demanding alternative is the subgradient (SG) method [Bertsekas et al., 1999], which proceeds as follows: Let $\theta = (C, X, Z, \mathcal{B})$ be the parameters, and $\Omega = \mathcal{C} \times \mathcal{X} \times \mathcal{Z} \times D$ the constraint-set; then SG iterates

$$\theta^{t+1} = P_\Omega(\theta^t - \alpha_t g^t), \tag{6.10}$$

where $g^t \in \partial\psi(\theta^t)$ is a subgradient, and $\alpha_t > 0$ is a stepsize. The bulk of the computation in (6.10) lies in computing g^t. This computation becomes unattractive when the number of

observations N, grows large. Since in our denoising application, N is usually large ($N \sim 10^5$), we need a better alternative. A practical, scalable, and effective alternative is developed below.

6.4 Online Dictionary Learning

Since our ultimate goal is to recover a robust dictionary $\mathcal{B}$, let us recast (6.9) in a more suitable form. Ignoring constraints for the moment, by separating entities varying with i from the parts that remain constant, we may rewrite (6.9) as the nested minimization[1]

$$\min_{\mathcal{B}} \quad \Phi(\mathcal{B}) = \sum\nolimits_{i=1}^{N} \phi_i(u_i; \mathcal{B}), \tag{6.11}$$

where $u_i = (c_i, x_i, z_i)$, and ϕ_i is defined as the minimum

$$\phi_i(u_i; \mathcal{B}) := \min_{u_i} \big(\frac{1}{2}\|x_i - \mathcal{B}c_i\|_2^2 + \gamma_i\|c_i\|_1 + \lambda_i\|y_i - z_i\|_1 + \frac{1}{2}\beta_i\|z_i - x_i\|_2^2 \big). \tag{6.12}$$

Now, we propose to replace iteration (6.10) by the following *stochastic-gradient* iteration at step t:

$$\mathcal{B}^{t+1} = P_D(\mathcal{B}^t - \alpha_t \nabla_{\mathcal{B}} \phi_k(u_k; \mathcal{B}^t)), \tag{6.13}$$

where $1 \le k \le N$ is some index, and α_t is a diminishing step-size. That is, α_t should satisfy $\lim_t \alpha_t = 0$ and $\lim_t \sum_j \alpha_{j=1}^t = \infty$ (e.g., $\alpha_t \propto 1/t$). The set D represents the constraints $\|b_j\|_2 \le 1$, on each column b_j of $\mathcal{B}$. Under suitable assumptions, one can show that $\Phi(\mathcal{B}^t) \to \Phi(\mathcal{B}^*)$, where $\mathcal{B}^*$ is a stationary-point [Bottou, 1998].

Remark: The basic stochastic-gradient iteration (6.13) may be augmented by incorporating curvature information and iterating

$$\mathcal{B}^{t+1} = P_{\mathcal{B}}(\mathcal{B}^t - \alpha_t S^t \nabla_{\mathcal{B}} \phi_k(u_k; \mathcal{B}^t)),$$

where S^t is an appropriate positive-definite matrix [Bottou, 1998].

The only remaining details are on how to solve (6.12). This computation can be split into three sub-iterations (with $s = 0, 1, \ldots$):

$$c_i^{s+1} = \operatorname{argmin}_c \quad \frac{1}{2}\|x_i^s - \mathcal{B}c\|_2^2 + \gamma_i\|c\|_1, \tag{6.14a}$$

$$x_i^{s+1} = \operatorname{argmin}_x \quad \frac{1}{2}\|x - \mathcal{B}c_i^{s+1}\|_2^2 + \frac{1}{2}\beta_i\|z_i^s - x\|_2^2, \tag{6.14b}$$

$$z_i^{s+1} = \operatorname{argmin}_z \quad \lambda_i\|z - y_i\|_1 + \frac{1}{2}\beta_i\|z - x_i^{s+1}\|_2^2. \tag{6.14c}$$

[1] A similar reformulation, for a simpler problem was also used by [Mairal et al., 2010], though the ultimate algorithm used was more complicated than ours.

Assuming for simplicity that there are no additional constraints on the variables, these subproblems can be solved as follows:

$$c_i^{s+1} = \text{LASSO}(x_i^s, \mathcal{B}, \gamma_i) \tag{6.15a}$$

$$x_i^{s+1} = (1+\beta_i)^{-1}(\beta_i z_i^s + \mathcal{B}c_i^{s+1}) \tag{6.15b}$$

$$z_i^{s+1} = y_i + \text{sgn}(x_i^{s+1} - y_i) \cdot (|x_i^{s+1} - y_i| - \lambda_i/\beta_i)^+, \tag{6.15c}$$

where $\cdot$ denotes elementwise multiplication. Theoretically, to obtain the optimal values (c_i^*, x_i^*, z_i^*) we must iterate (6.14a)–(6.14c) to convergence, but we iterate these only a few times for efficiency. Combining the above details we obtain our new algorithm: Sparse Online Learning of Dictionaries (SOLD), described in Algorithm 3. The convergence analysis of our algorithm is similar to that for the online dictionary learning given in [Mairal et al., 2010].

Algorithm 3 Sparse Online Learning of Dictionaries (SOLD)

Require: $\mathcal{D}\{\text{dataset}\}, \alpha, \beta, \gamma, \lambda$

$\mathcal{B}_0 \in \mathbb{R}^{d\times n}$ {initialize dictionary}, T {number of iterations}
for $t = 1, 2, \cdots, T$ **do**
  Draw y_i from $\mathcal{D}$
  $x_i^0 \leftarrow 0, c_i^0 \leftarrow 0$
  for $j = 1, 2, \cdots, s$ **do**
    $z_i^j \leftarrow y_i + \max\left(|x_i^{j-1} - y_i| - \frac{\lambda}{\beta}, 0\right).sgn(x_i^{j-1} - y_i)$
    $x_i^j \leftarrow \frac{1}{1+\beta}\left(\beta z_i^j + \mathcal{B}_{t-1}c_i^{j-1}\right)$
    $c_i^j \leftarrow LASSO(x_i^j, \mathcal{B}_{t-1}, \gamma)$
  end for
  {dictionary update step}
  $\nabla_{\mathcal{B}} \leftarrow \mathcal{B}_{t-1}c_i^s c_i^{sT} - x_i^s c_i^{sT}$
  $\mathcal{B}_t \leftarrow P_{\mathcal{B}}(\mathcal{B}_t - \alpha\nabla_{\mathcal{B}})$
end for
return $\mathcal{B}_T$

6.5 Experiments and Results

We present several experimental results to demonstrate the effectiveness of SOLD. Our first set of results is in a controlled setting with synthetic data, while our second batch of experiments assesses feature matching performance on a real SIFT dataset. All the experiments compare

the performance of SOLD to the LASSO-based [Cherian et al., 2010] and Gaussian denoising solutions [Elad and Aharon, 2006].

6.5.1 Simulations

For our controlled setting, we experiment with 10-dimensional vectors drawn from a *known* sparse basis, and distorted with both Gaussian and Laplacian noise. We generate 50,000 samples of such vectors, and use them to learn a 10×20 dictionary. We then test the dictionary by using (6.15) to compute a relevant nonzero basis. A *perfect match* happens when all the nonzero indices in the sparsified query and the database descriptors match exactly. Table 6.1 shows the nearest neighbors performance of our algorithm on a test-set with 100 data vectors. The table shows that for data satisfying the proposed noise model, our algorithm significantly outperforms competing approaches in correctly matching the data vectors.

Denoising Method	# Perfect matches	Avg. basis overlap
Lasso-based [Cherian et al., 2010]	4	52%
Gaussian [Elad and Aharon, 2006]	7	52%
SOLD (our approach)	**38**	**61**%

Table 6.1: Matching accuracy of NN on a test set of 100 vectors (10-dimensional); the learned dictionary had 20 basis vectors.

6.5.2 Experiments on SIFT Data

To show that SOLD is effective beyond mere simulated data, we now experiment with actual SIFT data. We base our algorithm's effectiveness on its ability to sparsify the SIFT descriptors and on the quality of NN search enabled by the sparse coefficients. We use the benchmark dataset of [Mikolajczyk and Schmid, 2005] for both training and testing; each of the SIFT descriptors was normalized to have zero mean and unit variance.

Dictionary Learning: We use 500K SIFT descriptors, each of dimension 128, to learn an overcomplete dictionary. Since it is difficult practically to simultaneously store all the descriptors in RAM, our online method SOLD is particularly suited for this problem.To accelerate SOLD further, we use mini-batches of size 10 while executing the stochastic gradient descent iterations.

The optimal dictionary size is obtained via cross-validation. Specifically, we learn dictionaries with varying number of bases, and then test their recall performance on a test dataset of 1000 descriptors. The recall values are shown in Figure 6.5.2; the maximum recall is obtained on a dictionary with 1024 vectors, so we use this dictionary for the subsequent experiments. The plot also indicates that increasing the dictionary size does not help improve NN accuracy, perhaps because the basis vectors become increasingly coherent.

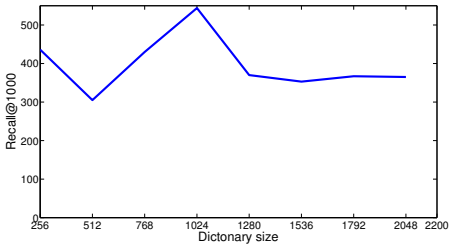


Figure 6.3: Recall with varying dictionary size (on 1000 data points).

Signal-to-Noise Ratio: We now look at the accuracy of our algorithm in reconstructing a SIFT descriptor as the noise-power is varied. For a *true* descriptor x we obtain an approximation $\mathcal{B}c$ via SOLD, which we compare by using the relative reconstruction error:

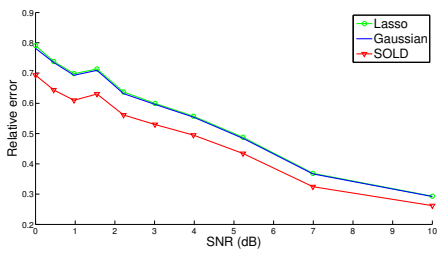
$$\epsilon_R := \|x - \mathcal{B}c\|_2 / \|x\|_2. \tag{6.16}$$

Figures 6.4(a), 6.4(b), 6.4(c) demonstrate SOLD's performance for varying noise power when only sparse noise is present and Figures 6.4(d), 6.4(e) show the performance when both sparse and Gaussian noise are present. We compare the reconstruction error values ϵ_R of SOLD against LASSO-based and Gaussian denoising [Elad and Aharon, 2006]. The figures show that compared to the other two methods, our sparse denoising framework achieves the lowest reconstruction error.

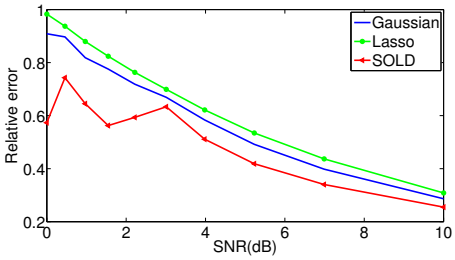
6.5.3 Nearest Neighbors Matching

Finally, we present the NN matching accuracies achieved by SOLD on real SIFT descriptors. To test SOLD we use a SIFT benchmark image dataset,[2] which consists of categories of image

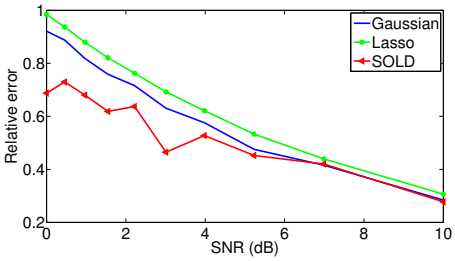
[2] http://www.robots.ox.ac.uk/∼vgg/research/affine/index.html



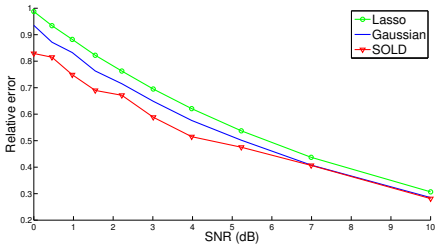
(a) ($\mathcal{L}(0, 0.1)$, No Gaussian)



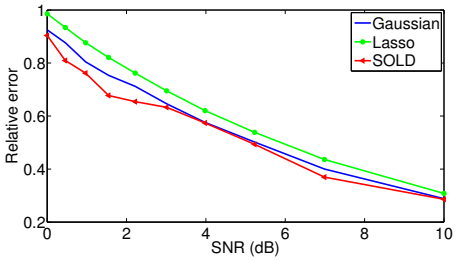
(b) ($\mathcal{L}(0, 0.2)$, No Gaussian)



(c) ($\mathcal{L}(0, 0.3)$, No Gaussian)



(d) ($\mathcal{L}(0, 0.1), \mathcal{N}(0, 0.05)$)



(e) ($\mathcal{L}(0, 0.1)$,$\mathcal{N}(0, 0.1)$)

Figure 6.4: Figures show the reconstruction error against various noise levels and varying SNR.

pairs (of the same scene) under various transformations. First we generate descriptors from this dataset for a random collection of image pairs from a specific distortion category. Then, we sparsify these descriptors using a dictionary of size 1024. Finally, we use a hash table to match the tuples. The ground truth is obtained by computing matching pairs using the Best-Bin-First (BBF) algorithm [Lowe, 2004] on the collection of generated descriptors, from which we randomly choose a set of 1000 correct pairs to form the baseline. We measure performance by counting the number of descriptor pairs for which the output of the candidate algorithms matches with the ground truth. The result of this evaluation is shown in Table 6.2, where we compare SOLD against Lasso-based and Gaussian denoising methods. The results indicate the superiority of SOLD.

Denoising Method	# Perfect matches	Avg. basis overlap
Lasso-based [Cherian et al., 2010]	208	58%
Gaussian [Elad and Aharon, 2006]	252	71%
SOLD (our algorithm)	**342**	66%

Table 6.2: Recall of nearest neighbors over SIFT descriptors against a baseline of 1000 correct matches. The regularization parameters of the methods were fixed for equal reconstruction error.

6.6 Nearest Neighbors via Robust Optimization

The last section studied a probabilistic approach to learning dictionaries that are robust to data perturbations. An obvious drawback to such a scheme is that in practice it is often difficult to characterize the distribution of such perturbations, or derive tractable density functions even when we have a good idea about the nature of these perturbations. A standard way to avoid difficulties in dealing with data uncertainties is to resort to robust optimization principles; the main idea is to explicitly model the worst case data uncertainty.

In the robust optimization literature, the formulation we derived in Section 6.1 comes under the class of *chance constrained* problems. Such problems can be generically written in our context as follows: given a data point v, we seek an associated sparse code w such that:

$$\min_{w,t} \left\{ t : Prob_{\theta \sim P} \left(\phi(w; \theta, v) \leq t \right) \geq 1 - \epsilon \right\}, \tag{6.17}$$

where ϕ is an appropriate loss function, θ encapsulates the parameters of the model including the dictionary $\mathcal{B}$, P represents the distribution of the data (such as the Gaussian and Laplacian that we discussed in Section 6.1). We assume $\epsilon \approx 0$ as the tolerance. When the distribution of P is unknown, we have an *ambiguous chance constrained* problem [Ben-Tal et al., 2009], which takes the following variant of (6.17):

$$\min_{w,t} \left\{ t : Prob_{\theta \sim P} \left(\phi(w; \theta, v) \leq t \right) \geq 1 - \epsilon, \forall P \in \mathcal{U} \right\}, \tag{6.18}$$

where now the distribution P is not well-defined, but what is known is that it belongs to a set $\mathcal{U}$ of all valid distributions. Such a scheme provides a rich setup for inference. Unfortunately, care must be taken to make sure that the robust model remains tractable. In the rest of this chapter, we will look at the robust counterparts of chance constrained problems for our case of sparse coding. Our approach is basically to model the allowed uncertainties that a given data point might undergo, subsequently incorporating the worst case uncertainty in our sparse coding formulation so that our hash codes are *immune* to these perturbations. To this end, we will first propose a robust variant of the LASSO formulation. This will help bring out the computational issues associated. Later we will derive a framework that remains tractable and at the same time makes the sparse codes robust to noise. Experiments are provided demonstrating the effectiveness of our strategy. To put our contributions into perspective, below we summarize previous work that is directly related to robust optimization and dictionary learning.

6.7 Related Work

Robust optimization (RO) [Ben-Tal and Nemirovski, 1999] is a well-established branch of non-linear programming that has of late been gaining importance in machine learning [Caraminis et al., 2011]. The normal DL problem has been viewed from certain robust perspectives previously, but these are different from the formal notions of robustness propounded by RO. For example, in [Bar and Sapiro, 2010], the image sparse coding problem is made robust to image rotations and scaling through learning a dictionary in the log-polar space. In [Ramirez et al., 2009], a robust formulation of the dictionary learning problem is suggested in the context of outlier detection by using the Lorentian norm instead of the L1 norm. This paper also proposes constraints on the dictionary atoms for better incoherence and thereby improve the robustness of the reconstruction. In contrast, we ground our Robust Dictionary Learning (RDL) approach using the notion of robustness from RO, wherein the solution sought must be invariant to certain bounded uncertainty in the data.

6.8 Robust Dictionary Learning

Before proceeding to introduce our algorithms, let us fix the notations for this section, along with recalling the context of robustness in the dictionary learning setting. Mathematically, suppose $\mathcal{D} = \{v_1, \ldots, v_m\}$ (each $v_i \in \mathbb{R}^d$) is the set of input vectors for which we have learned a dictionary $\mathcal{B} \in \mathbb{R}^{d \times n}$ (where we will use b_i to denote the ith column of $\mathcal{B}$) and sparse vectors $w_1, \ldots, w_m$ by solving the LASSO problem (5.3). For an arbitrary $v \in \mathcal{D}$, let $J(v) := \{j_1, \ldots, j_k\}$ be the corresponding set of indices such that $w_{j_l} \neq 0$ for $1 \leq l \leq k$ and $v \approx \mathcal{B}w$. As we saw in earlier chapters, the set $J(v)$ may be viewed as a hash-code (SCT) for v. Indeed, we can build a data structure that stores $\mathcal{B}$ along with the vectors $\mathcal{D}$ hashed according to $\{J(v_1), \ldots, J(v_m)\}$. Now, suppose a new (test or query) point v' is presented. We first obtain the corresponding sparse representation $v' \approx \mathcal{B}w'$ by solving

$$w' = \text{argmin}_w \quad \frac{1}{2}\|v' - \mathcal{B}w\|_2^2 + \lambda\|w\|_1, \tag{6.19}$$

to obtain the hashkey $J(v')$, which we can rapidly look up in the hash table. The nearest data points are retrieved by scanning through (linearly or via a further data structure) the points associated with the key $J(v')$. Though the above NN scheme seems attractive, it could be brittle in the presence of noise. In symbols, suppose $v \approx \mathcal{B}w$ and $\bar{v} = (1 + \gamma)v$, for some

small perturbation $\gamma \neq 0$. For facilitating NN retrieval, we desire $J(v) = J(\bar{v})$; but since $\mathcal{B}$ is overcomplete there is *no guarantee* that $w = (1+\gamma)\bar{w}$.

6.8.1 Robust Formulation

Let there be a set of nominal data vectors $\bar{\mathcal{D}} = \{\bar{v}_1, \ldots, \bar{v}_m\}$ that is sparse in an appropriate dictionary $\mathcal{B}$. Let $\mathcal{U}(\bar{v})$ model *uncertainty* for the point $\bar{v}$, i.e., it is some set that models perturbations in $\bar{v}$. To be robust against the perturbations, RDL seeks a dictionary so that all points $v \in \mathcal{U}(\bar{v})$ have the same hash codes: $J(v) = J(\bar{v})$. A common approach [Caraminis et al., 2011] is to seek an optimal solution that is immune to the worst-case uncertainty. Thus, we consider its *robust* counterpart:

$$\begin{aligned} \min_{\mathcal{B}, w_1, \ldots, w_m} \quad & \sum_{i=1}^{m} \big(\frac{1}{2}\|\bar{v}_i - \mathcal{B}w_i\|_2^2 + \lambda\|w_i\|_1\big) \\ \text{s.t.} \quad & \|b_j\|_2 \leq 1, \text{ for } j = 1, 2, \ldots, n \\ & v_i \in \mathcal{U}(\bar{v}_i), \text{ for } i = 1, 2, \ldots, m. \end{aligned} \tag{6.20}$$

Some practical choices for $\mathcal{U}$ are [Ben-Tal and Nemirovski, 1999]:

- *Cardinality:* $\mathcal{U}(\bar{v}) := \{v : |\text{supp}(v) \setminus \text{supp}(\bar{v})| \leq \delta\}$
- *Ellipsoidal:* $\mathcal{U}(\bar{v}) := \{v : v = \bar{v} - Pu,\ \|u\|_2 \leq 1,\ P \succ 0\}$,
- *Norm-ball:* $\mathcal{U}(\bar{v}) := \{v : \|v - \bar{v}\|_p \leq \delta\}$ for $p \geq 1$.

Of these choices, the cardinality constraint does not adhere by our notion of hashing and thus we do not consider it further in this thesis. Let us take a closer look at the other two possibilities.

6.8.2 Ellipsoidal Uncertainty

Assuming that the data vectors belong to uncertainty sets around their respective nominal data vectors, we can expand (6.20) to the following formulation:

$$\begin{aligned} & \min_{\mathcal{B}, W} \sum_{i=1}^{m} \big(\frac{1}{2}\|\bar{v}_i - \mathcal{B}w_i\|_2^2 + \lambda\|w_i\|_1\big) \\ \text{where} \quad & (v_i - \bar{v}_i)^T P\,(v_i - \bar{v}_i) \leq 1 \text{ for } i = 1, 2, \ldots, m \\ \text{and} \quad & \|b_j\|_2 \leq 1, \text{ for } j = 1, 2, \ldots, n, \end{aligned} \tag{6.21}$$

where P is an invertible matrix which is assumed to be known and $W = \{w_1, \ldots, w_m\}$. Problem 6.21 is a *Quadratically Constrainted Quadratic Program* (QCQP), which is in general

known to be NP-hard and thus our formulation might seem to be of little value. To this end, we propose an alternative view of the robustness formulation in which we seek to find worst case perturbations in the uncertainty sets that make the data vectors immune to any perturbations. The main idea being to add to the data point v, a direction of worst case perturbation expected from the dataset such that the data uncertainties become inconsequential while sparse coding. This idea is captured by the following formulation: for a data point v, we compute the *robustified* data vector $\hat{v}$ as follows:

$$\hat{v} := v + Pu^* \text{ where } u^* = \max_{\|u\|_2 \leq 1} \|v + Pu\|_2^2. \tag{6.22}$$

Now, assuming that this robustified data vector $\hat{v}$ has an associated sparse dictionary, we get our new RDL formulation:

$$\begin{aligned} &\min_{\mathcal{B},W} \sum\nolimits_{i=1}^{m} \big(\frac{1}{2}\|v_i + Pu_i^* - \mathcal{B}w_i\|_2^2 + \lambda\|w_i\|_1\big) \\ \text{where} \quad &u_i^* := \max_{\|u\|_2 \leq 1} \frac{1}{2}\|v_i + Pu\|_2^2 \\ \text{and} \quad &\|b_j\|_2 \leq 1, \text{ for } j = 1, 2, \ldots, n. \end{aligned} \tag{6.23}$$

The formulation (6.23) is our core RDL problem; but before we go into potential algorithms for solving it, we would like to explore the case of *Norm-ball* uncertainty sets we described earlier.

6.8.3 Norm-ball Uncertainty

Following the derivation in (6.23), it can be shown that the norm-ball constraint for an ℓ_2 norm leads to the following formulation:

$$\hat{v} := v + su^* \text{ where } u^* := \max_{u} \ \|v + su\|_2^2, \ s.t. \ \|u\|_2 <= 1, \tag{6.24}$$

for a scalar s, the solution of which is trivial: $u = v/\|v\|_2$. Unfortunately, this setup did not capture the required robustness for SIFT descriptors in our experiments. Other uncertainty models (eg. polygonal uncertainty) lead to complicated optimization problems; *ellipsoidal uncertainty* offers a tractable tradeoff, and thus let us move on to deriving efficient algorithms for solving the RDL formulation given in (6.23).

6.9 Algorithms for RDL

The basic difference between the traditional dictionary learning problem (5.5) and the robust formulation (6.23) is in computing u^* which is the direction of the worst case perturbation. Since u is dependent only on the data points v, we can solve for it independently of $\mathcal{B}$ and W. Once u is computed for each $\hat{v}$, (6.23) boils down to (6.19) where the data points are now the robustified vectors $\hat{v} = v + Pu^*$. To this end, the primary optimization problem that we need to tackle is in efficiently solving for u; avoiding the subscripts for simplicity, the formulation in (6.23) on u^* can be rewritten as:

$$\max_{\|u\|_2 \leq 1} \quad \frac{1}{2}\|v + Pu\|_2^2, \tag{6.25}$$

Upon expanding, and replacing maximization by minimization (6.25) becomes:

$$f(u) := \min_{\|u\|_2 \leq 1} \quad -\frac{1}{2}u^T P^T P u - u^T P v - \frac{1}{2}v^T v, \tag{6.26}$$

At first sight (6.26) might look bad, because the quadratic term is $-P^T P$, which is negative-definite instead of positive-definite. That is, (6.26) is a nonconvex quadratic program. Fortunately, because there is just one constraint, (6.26) still remains tractable. In fact, as is well-known, for this particular type of nonconvex problems, strong-duality holds thanks to the so-called S-lemma [see e.g.[Boyd and Vandenberghe, 2004]]. Problem (6.26) is even more special: it is a trust-region subproblem, although a non-trivial one because of the negative-definite matrix $-P^T P$. Though, we can use a generic trust-region solver like the LSTRS (Large Scale Trust Region Solver) method [Rojas et al., 2008] for this problem, we show below that this problem can be solved more efficiently.

6.9.1 Efficient Implementation via Newton Descent

Using a multiplier γ, the Lagrangian of (6.26) can be written as:

$$L(\gamma, u) := -\frac{1}{2}u^T P^T P u - u^T P v - \frac{1}{2}v^T v + \gamma\left(u^T u - 1\right).$$

Setting $\frac{\partial L}{\partial u} = 0$, we have: $u = \left(P^T P - 2\gamma\mathcal{I}\right)^{-1} Pv$, where $\mathcal{I}$ is the $d \times d$ identity matrix. Let $P^T P = U\Sigma U^T$ for an orthonormal U and diagonal Σ, and substituting for u in the constraint, we have the following root finding problem in the scalar γ:

$$v^T U^T \Sigma \left(\Sigma - 2\gamma\mathcal{I}\right)^{-2} \Sigma U v = 1.$$

Using the substitution $q = \Sigma U v$ and using the Hadamard product $\hat{q} = (q \circ q)$ then leads to a simple vector equation:

$$\sum_{i=1}^{d} \frac{\hat{q}_i}{(\sigma_i - 2\gamma)^2} = 1. \tag{6.27}$$

where $\hat{q}_i$ is the ith component of $\hat{q}$ and σ_i is the ith eigenvalue of Σ. As is clear, (6.27) is convex in γ and can be efficiently solved by the *Newton-Raphson* method for a suitable initialization. For a reasonably well-conditioned matrix $P^T P$, (in our implementation, we initialize the iterations with $\gamma = -\max(diag(\Sigma)^2)$), on an average, 3–6 times speedup was observed against the LSTRS method. Figure 6.5 shows the average speedup produced by our algorithm compared to the LSTRS method for an increasing dimensionality of the P matrix and simulated data.

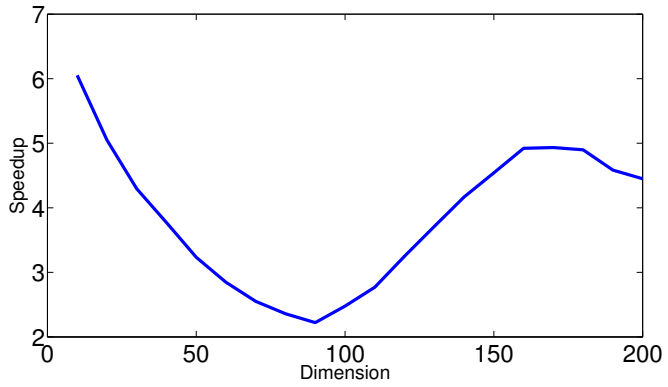


Figure 6.5: A plot of the speedup produced by the Newton Descent solution of RDL problem against the LSTRS solver. The x-axis is the dimensionality of the data uncertainty matrix.

6.9.2 Computing the Uncertainty Matrix

The parameter P of our formulation plays a crucial role in quantifying uncertainty and thus care must be taken in learning this parameter. Towards this end, we propose a supervised approach by setting P to the *Lowner-Jones* uncertainty ellipsoid computed on matching data pairs. That is: suppose we have a training set of matching data pairs $\{(v_i, v_i'), i = 1, 2, \cdots, N\}$, where we assume $v_i = \bar{v}$ is the nominal data point and $v_i' = \bar{v} + \epsilon_i$ for some noise ϵ_i. We compute the difference $\epsilon_i = v_i - v_i', i = 1, 2, \cdots, N$. An uncertainty elliposid can be defined as an ellipsoid of *minimum volume* enclosing all the ϵ_i's. If P represents the parameters of this ellipsoid and if

c is its centroid, then

$$\min_{P \succ 0, c} \; -\log det(P)$$
$$\text{s. t. } (\epsilon_i - c)^T P (\epsilon_i - c) \leq 1, \tag{6.28}$$

for $i = 1, \cdots, N$. The problem (6.28) can be solved via the well-known *Khachiyan* algorithm. Since we work with zero-mean data, c is generally seen to be very close to the origin and thus can be neglected.

6.10 Experiments and Results

In this section, experiments are presented to evaluate the effectiveness of RDL. First, we will recall the metrics against which the performance is measured and later compare RDL against the state-of-the-art methods.

Basis Overlap: We use the same definition of the BO metric as we introduced in (5.26). When the BO is 100%, we will refer to it as the *Perfect Basis Overlap*, which means these data descriptors will adorn the same SCTs.

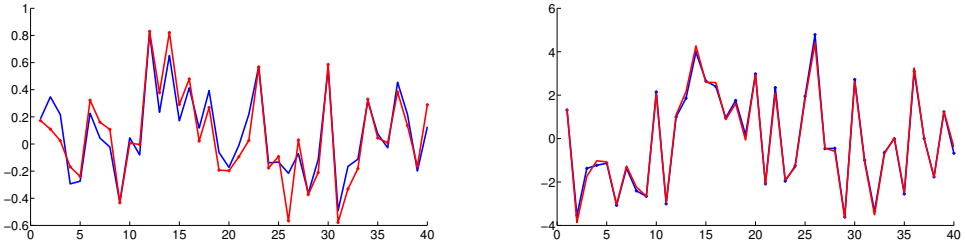
Accuracy: Due to unavailability of the ground truth for our datasets, the baselines were decided by the standard distance measure (Euclidean distance) via a linear scan. For the datasets, we create a database and a query set of q items. For each query item i, k ground truth neighbors (G_i^k) were found using the linear scan, followed by k nearest neighbors (A_i^k) are retrieved using the respective algorithm. We define

$$Accuracy = \frac{1}{q} \sum_i \frac{|G_i^k \cap A_i^k|}{|G_i^k|}. \tag{6.29}$$

Simulations: In this section, we will first illustrate using simulated data, how incorporating the worst case uncertainty increases the collision probability of the sparse codes. Recall that, in our robust formulation we map a given data point to a new vector $\hat{v} = v + Pu$ along the direction of maximal divergence u, which we later sparse code via the robust dictionary $\mathcal{B}$. To evaluate our idea and understand how robustness is achieved, we created a small dataset of 40D vectors generated by a known 40×120 dictionary. The data vectors were later corrupted by Gaussian

and sparse Laplacian noise to produce noisy data pairs. Figure 6.6 illustrates two matching data pairs (Figure 6.6(a)) and their robustified counterparts (Figure 6.6(b)). As is clear, the magnitudes of the sparse codes are now different, while their dimensions match almost exactly, showing that any SCT created from these robustified descriptors will be the same.

To further understand quantitatively the effectiveness of RDL, we applied it to ANN retrieval. To this end, we created 50K descriptor pairs using the same simulated setting, 40K of these descriptors were used to learn a new dictionary (on the robust data pairs). One-NN retrieval was tried on the rest of the descriptors. In Figure 6.7(a), we plot the average size of the active support for varying LASSO regularizations. The main motivation for this exercise is to show the superiority in robustness achieved by RDL against an otherwise robustness achieved by changing the LASSO regularization (which is a topic we will discuss in Chapter 7). We also plot the BO and Perfect BO using the ordinary LASSO and RDL formulations in Figure 6.7(b). The plots clearly show the superiority of our RDL formulation against a non-robust setting.

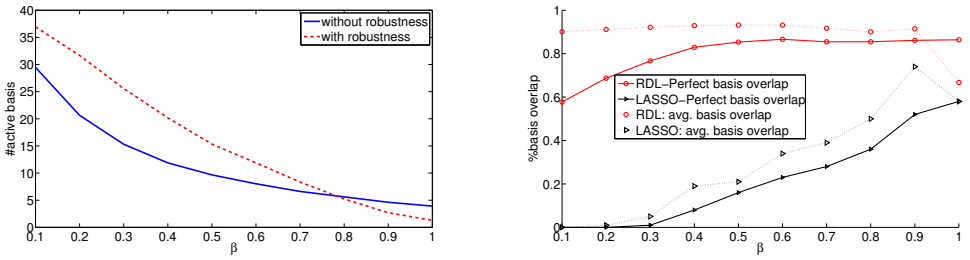


(a) (b)

Figure 6.6: Simulation results: (a) A 40D vector and its nearest neighbor in the Euclidean space, (b) The robustified counterparts of the noisy vectors in (a). As is clear from the plot, robust formulation brings the data points belonging to the uncertainty set closer in the new mapping.

6.10.1 Real Data Experiments

Dictionary Size: To choose the right size of the dictionary for the NN operation, we trained multiple dictionaries with number of bases varying from 256 to 4096 at steps of 256 and measured the NN performance on a small test set of 10K SIFT descriptors. Interestingly, we found that the NN performance decreased after a dictionary size of 2048, which we speculate is due to the increase in the coherence of dictionary atoms leading to ambiguous sparse codes. Thus,



(a) (b)

Figure 6.7: Simulation results: (a) Number of active non-zero dictionary indices against an increasing regularization, (b) Average performance of NN against increasing regularization. As is clear, using RDL formulation, we have more number of non-zero indices as well as increased basis overlap.

we used a dictionary of size 128×2048 in our later experiments. We used a regularization value $\lambda = 0.2$ in RDL. On an average 7 dictionary atoms were activated in sparse coding of the robustified descriptors.

Uncertainty Matrix: To decide the uncertainty ellipsoid matrix, we randomly selected a set of 20K points from the dataset and computed their one-nearest neighbor using a linear scan with Euclidean distance. Later, the difference of *matching* descriptors were used to find the ellipsoid using the algorithm mentioned before. Even though this step is computationally expensive, it has to be done only once.

Nearest Neighbor Performance: The basic motivation for RDL is to make the NN robust to different noise and image distortion models. We chose to compare our method with (i) KDT (Best-Bin-First based on KD-Trees), (ii) Kernelized LSH [Kulis and Grauman, 2009], (iii) LASSO based NN (DL), and (iv) Product Quantization (PQ) method [Jegou et al., 2011]. The latter technique was recently shown to outperform other well-known methods like spectral hashing and Hamming embedding. As for dataset, we used the standard SIFT benchmark dataset[3] which consists of eight image categories. Each category had six images each distorted in a certain way: (i) *BARK*, with images undergoing magnification, (ii) *BIKES*, with fogged images, (iii) *LEUVEN*, with illumination differences, (iv) *TREES*, with Gaussian blur, (v) *BOAT*,

[3] www.robots.ox.ac.uk/ vgg/research/affine/index.html

with magnification and rotation and (vi) *GRAPHIC*, with 3D camera transformation, (vii) *UBC*, with JPEG compression noise and (viii) *WALL*, with changes in view angle. Sample images from each of these categories are shown in Figure 6.8.

SIFT descriptors were extracted from each of the images in a category. Image pairs were chosen from each category and NN is applied on the descriptors from these pairs, where the descriptors from the first image is used as the database and the other as the query. We used the following approximate NN strategy to ensure an ϵ-neighborhood. We first computed the NNs for a given query point using linear scan. Suppose d_{ls} denotes the distance of the NN for a query point q to its closest neighbor in the database. If d_q is the distance of the closest neighbor found by an algorithm, then we say the NN is correct if $\frac{d_{ls}}{d_q} > \epsilon$ (we used $\epsilon = 0.9$ for all the algorithms). The results are shown in Figure 6.9. As is clear, RDL is seen to outperform all the other methods.

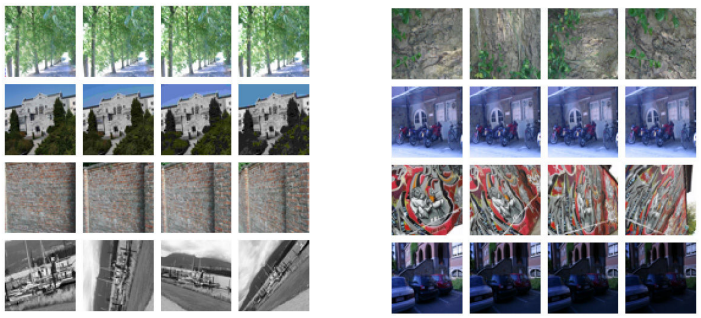


Figure 6.8: Dataset with various types of noise used for performance evaluation of RDL. *Left block:* (a) Gaussian blurred (TREES), (b) JPEG compression introduced noise (UBC) (c) viewpoint change (WALL), (d) affine transformation (BOAT). *Right block:* (e) image magnification (BARK), (f) blur (BIKES), (g) 3D transformation (GRAPHIC), and (h) illumination (LEUVEN).

Computational Cost: Our algorithms were implemented mainly in MATLAB. On a Pentium 4 machine with 3GHz CPU and 3GB RAM, it took on an average 9ms (in MATLAB) for the finding the robust directions (u) and less than 100μs for sparse coding a SIFT vector using the SPAMS toolbox[4] , followed by less than 1ms for querying a hash table of 1M SIFT descriptors. This performance seems comparable to those reported in [Jegou et al., 2011].

[4] http://www.di.ens.fr/willow/SPAMS/

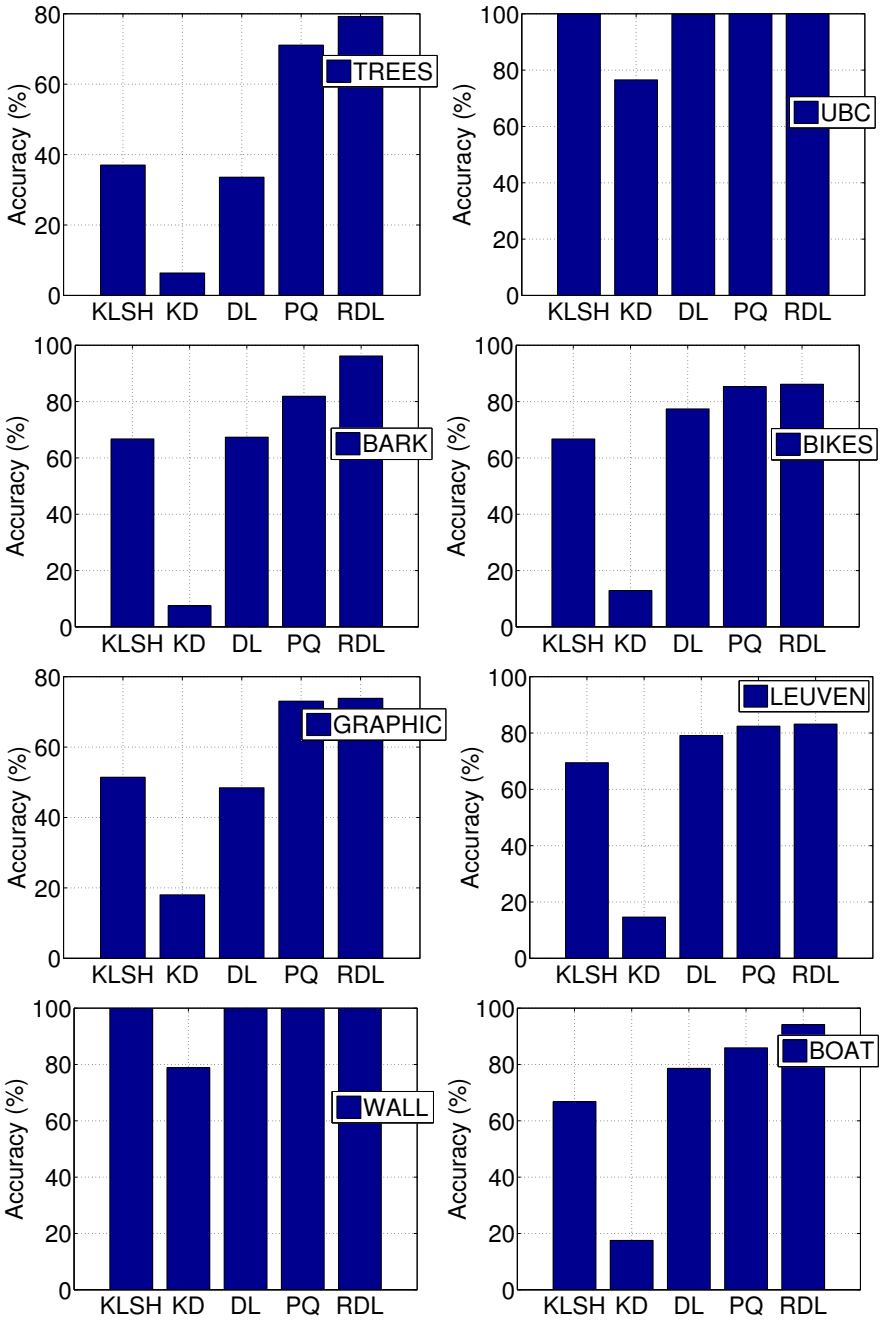


Figure 6.9: One nearset neighbor retrieval accuracy for the eight distortion categories using the RDL formulation, compared against state-of-the-art methods.

Chapter 7

Nearest Neighbors via Robust Sparse Coding

In previous chapters, we introduced sparse coding based approximate nearest neighbor retrieval and posed the problems that we need to tackle to get such an approach to work in practice. In the last chapter, our focus was in exploring ideas to make the dictionaries robust to data uncertainties. In this chapter, we will focus on a different robustness idea based on the following question: assuming a dictionary (learned using the traditional dictionary learning objective), is it possible to generate robust sparse codes? Our approach to address this question involves analyzing the effects of regularization on the LASSO formulation. This understanding will lead us to an alternative definition of our hash code, which is surprisingly simple, more accurate and easier to compute (compared to the approaches in the last chapter), at the same time this new approach is scalable even to web-scale datasets.

7.1 Robust Sparse Coding

In this chapter, we will first explicitly state a few issues that need to be circumvented when using the SCT0 representation that we introduced in Chapter 5. This will precede proposing novel variants of the SCT that will provide robust and efficient NN retrieval. In the following we will assume that we use the LARS algorithm for sparse coding, due to the advantages (such as data space partitioning we alluded to in Section 5.2.2) it offers for NN retrieval. Let us see the issues with SCT0 in detail.

1. Our first issue originates because of a property of the LARS algorithm. Specifically, due to the overcompleteness of the dictionary, the basis vectors in an active support can actually decrease in magnitude over the regularization path and may get dropped. Such a dropping of the basis produces a vacancy in the SCT tuple. Since SCT hashing is an exact string matching algorithm, dropping of a value from the string will fail the hashing. More intuitively, we would expect larger SCT sizes produce smaller data partitions. But when some active basis gets dropped, this leads to uninterpretable/unreliable data partitioning.

2. A second issue is associated with the size of the data partitions. When using smaller regularizations (which implies low sparsity), the subsequent data space partitions can be small (recall Theorem 21, which shows this fact). As a result, even the slightest perturbation in the data can make the perturbed data point to change the partitions (that is, to have different active sets). This will lead to producing different SCTs, failing the hashing algorithm.

3. A third problem (which of algorithmic in nature) is when the hash bucket for a given SCT is empty. We need to have a heuristic that decides which hash bucket may be looked at next to guarantee an ANN retrieval.

In the following sections, we will address all these issues using a robust sparse coding approach. That is, we will assume a dictionary learned using the traditional dictionary learning objective (given in (5.5)), but we will use a variant of the sparse coding objective that will produce robust SCTs. Let us consider the first problem that we posed above, in detail next.

7.2 Basis Drop

As we discussed in the review of the LARS algorithm in Section 5.2.2, at times the descent direction of the LARS will be against the direction of one of the active basis, i.e. if w_i^k ($\neq 0$) is the ith coordinate of the coefficient vector at the kth LARS step and if γ is the step size, then the descent at this step will follow:

$$w_i^{k+1} = w_i^k + \gamma \widehat{b_{\mathcal{A}}^i} \tag{7.1}$$

where $\widehat{b_{\mathcal{A}}^i}$ is the ith coordinate of the direction that is equiangular to all the bases in the active set $\mathcal{A}$. Since the dictionary bases are not assumed to be orthogonal, w_i^{k+1} crosses zero at $\gamma = -w_i^k / \widehat{b_{\mathcal{A}}^i}$. At this stage, LARS drops the active basis b_i from the active set $\mathcal{A}$ temporarily and adds it back in the next step (refer [Efron et al., 2004] for details). By definition, our SCT representation cannot take care of the possibility of dropping bases and thus the hash code generated defaults NN retrieval. This problem can be avoided if we make sure that the cardinality of the active support will never produce a linearly dependent set. Such a notion can be formalized by introducing the concept of *spark* of a dictionary.

Definition 11 (Spark [Elad, 2010])**.** *Spark of a given matrix $\mathcal{B}$ is defined as the smallest number of columns from $\mathcal{B}$ that are linearly dependent. That is,*

$$spark(\mathcal{B}) = \min_{\mathcal{B}x=0} \|x\|_0. \tag{7.2}$$

Theorem 26. *Using the notation we introduced in Chapter 5, let $T = h_{(BP;\lambda)}(v)$ be the SCT associated with a data point v. Further, let $\mathcal{B}$ be the associated dictionary, and $spark(\mathcal{B}) = k$. Then, there will not be any* basis drop *in LARS if $|T| < k$.*

Proof. The proof is directly implied by the definition of the matrix spark. □

Although the above idea seems attractive theoretically, computing the matrix spark is NP-hard. Fortunately, we can at least estimate a lower bound to the spark using the *coherence* of the dictionary as stated in the following theorem:

Theorem 27. *Suppose $\mu(\mathcal{B})$ is the coherence of a dictionary $\mathcal{B}$, then we have:*

$$spark(\mathcal{B}) \geq 1 + \frac{1}{\mu(\mathcal{B})}. \tag{7.3}$$

Proof. See [Elad, 2010][Lemma 2.1] for the proof. □

Unfortunately, a lower bound might not be sufficient for the success of SCT. Recall that the efficiency of hashing is proportional to the cardinality of the SCTs; thus longer SCTs are preferred. As a result, we decided to estimate the spark empirically.

We did the following experimental study of the regularization paths of the LARS solution on a part of our SIFT dataset consisting of approximately 100K SIFT descriptors. We computed the regularization paths of these descriptors by increasing the support set size from 1 to L, for $L = \{5, 10, 15, 20\}$. For example, for L=5, we computed LARS 5 times on a SIFT descriptor, each time incrementing the allowed number of supports by one (L is also the number of iterations of the LARS algorithm); the sign of each of the active basis in the set was then checked to see if any of the coefficients changed sign. Figure 7.1 shows the result of this experiment. The x-axis is the active set size (L) and y-axis plots the percentage of the dataset that had a zero-crossing at least once. We used a dictionary of size 2048 learned from 1M SIFT descriptors for this experiment and the regularization constant λ was fixed at 0.5.

As is clear from the plot, the number of zero-crossings increase as we allow the activated basis atoms to live longer in the LARS process. For $L = 5$, we observed that less than 0.2% of the dataset produced zero-crossings, while this increased to approximately 50% for $L = 20$. This result suggests that if we restrict the support set size to less than 5, the effect of the zero-crossing problem is negligible, and practically can be ignored. Based on this observation, we will assume that LARS will not encounter any *basis drop*. Formally,

Condition 1. *For a data vector v, let w^k be the k-sparse vector found by LARS at step k and let w_i^k denote the ith dimension of w^k. Then, if $w_i^k \neq 0$, implies $w_i^{k+1} \neq 0$.*

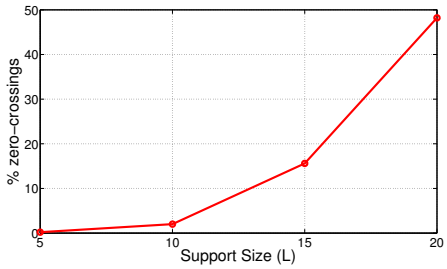


Figure 7.1: A plot of the percentage of 100K SIFT descriptors that had atleast one zero-crossing in its sparse regularization path against the allowed support set size.

7.3 Data Perturbations

Next, let us consider the second problem (that we described in the introduction of this chapter) in detail. For data points v_1 and v_2, assume that $T_1 = h_{(BP;\lambda)}(v_1)$ and $T_2 = h_{(BP;\lambda)}(v_2)$ are the respective SCTs. Assume that $v_1 = v + \epsilon_1$ and $v_2 = v + \epsilon_2$, where ϵ_1, ϵ_2 represent random noise and v is the underlying noise-free data vector. Suppose, we apply (5.3) to v_1 and v_2 using a dictionary $\mathcal{B}$ and a regularization λ, due to noise, there is no guarantee that $T_1 = T_2$. This is because some of the atoms in the dictionary might be used to represent the noise. This fails our SCT based hash table indexing.

The basic motivation for our approach to overcome this difficulty comes from ideas in wavelet based multiscale analysis, in which a given signal is approximated to various error levels by restricting the choice of the wavelet scales used. That is, the signal is approximated poorly in coarse scales and the error residual is decreased when using finer and finer scales. Analogously, when using sparsity, smaller SCTs (or larger sparsity) correspond to larger data partitions and larger data partitions implies more space to accommodate data perturbations, leading to better robustness. Such a scheme of adjusting the scale of sparse coding can be achieved by varying the regularization value of the LASSO formulation. Due to this, we will call our algorithm *Multi-Regularization Sparse Coding* (MRSC). To set the stage for our discussion on MRSC, the following section reviews the concept of regularization and builds up a few theoretical results that will come useful when putting forth the main arguments of this chapter.

7.3.1 Regularization

Our idea is to build a connection between the regularization values and sparsity of a given data vector as produced by LARS. This will help us create hash codes (SCT) of different lengths for a given data vector using multiple values of the regularizer. The following results directly follow from Algorithm 1 and Condition 1 that we discussed in Chapter 5.

Proposition 28. *With reference to the LASSO formulation* (5.4) *(*SC_{CBP}*) and assuming Condition 1, let* v *be k-sparse in* $\mathcal{B}$ *and assume that* w^j *be the sparse vector found by LARS at the* j*th step (*$0 < j \leq k - 1$*). Then,*

I *The cardinality of the support of* w *increases in each LARS step, i.e.* $\|w^j\|_0 = j$ *and* $\|w^{j+1}\|_0 > \|w^j\|_0$.

II *Increasing the reconstruction error bound (δ) decreases sparsity, i.e. there exist bounds $\delta, \delta', \delta' < \delta$, so that if w_δ and $w_{\delta'}$ are the sparse codes generated using bounds δ and δ' respectively by LARS, then $\|w_\delta\|_0 = j$ and $\|w_{\delta'}\|_0 = j + 1$.*

Proof. Considering each case,

I This follows directly from Algorithm 1, if Condition 1 is imposed.

II Let $\|C^j\|_2$ be the minimizer of (5.3) for a bound δ and assuming Condition 1. Given that v is k-sparse and $j \leq k - 1$, it is clear that there should exist a residual C such that $\|C\|_2 < \|C^j\|_2$. Now, given the convexity of the formulation and the steepest gradient descent at each LARS step, we can achieve $\|C\|_2$ only by continuing the loop in Algorithm 1. This implies that we need to decrease the reconstruction error bound to $\gamma < \delta$. Now, given that LASSO regularization path is continuous and piecewise linear (refer [Mairal and Yu, 2012][Lemma 2]), there should exist a $\gamma \leq \delta' < \delta$ such that $\|w_{\delta'}\|_0 = j + 1$. □

When the Coherence $\mu(\mathcal{B}) = 1$, there will be multiple equivalent LARS regularization paths (which deters its usefulness to NN retrieval). The following proposition formally states this fact.

Proposition 29. *Assuming Condition 1, for a data vector v, let $\mathcal{A} = \{b_{i1}, b_{i2}, \cdots, b_{ik}\}$ be a well-ordered set of basis vectors $b_{ij} \in \mathcal{B}$ selected by LARS, with the ordering $b_s < b_t$, if $s < t$ for LARS steps s and t. Then the sequence $\mathcal{A}$ is* unique *iff $\mu(\mathcal{B}) < 1$.*

Proof. It is easy to see that if $\mu(\mathcal{B}) = 1$, step 4 of Algorithm 1 will select multiple equally correlated bases. □

Lemma 30. *If $b_i \in \mathcal{B}$ is the most correlated basis to a data vector v, then b_i will be selected in the first step of LARS.*

Proof. This is true by the definition of LARS algorithm. □

Using the above analysis, we are now ready to present two alternative definitions of the SCT that will prepare us for a robust sparse code.

Definition 12 (Subspace Combination Tuple: SCT1)**.** *Let a data vector v and a dictionary $\mathcal{B}$ be given and that $w = \mathrm{SC}_{BP}(\lambda; v)$. A tuple of integers $h(w) = \{i_1, i_2, \cdots, i_k\}$ is defined as an SCT1, if $i_1, i_2, \cdots, i_k$ are the unique identifiers of the bases and are arranged in the order in which the associated bases are* activated *by the LARS algorithm.*

Intuitively, it is easy to see that SCT1 implicitly captures the regularization path of the data vector in the given dictionary, as produced by the LARS algorithm. Unfortunately, there are two technical difficulties our SCT1 definition introduces. Firstly, our definition strongly ties the problem of NN retrieval to a specific sparse coding algorithm (LARS in our case). Suppose we have a more efficient LASSO solver than LARS, but which does not provide a regularization path explicitly (such as the In-Crowd algorithm [Gill et al., 2011] or the Feature Sign Search [Lee et al., 2005]). We would like to still use the hashing scheme with such solvers. Secondly, existing LARS implementations might not return the LARS regularization path, unless modified suitably (for example, if one uses binary code libraries such as the SPAMS toolbox [Mairal et al., 2010] for sparse coding without the intention to modify the code). To circumvent these technical difficulties, we propose an alternative but slightly less efficient approximation to SCT1, which achieves the same effect in practice.

Definition 13 (Subspace Combination Tuple: SCT2)**.** *Let a data vector v and a dictionary $\mathcal{B}$ be given and that $w = \mathrm{SC}_{BP}(\lambda; v)$ be the k sparse code associated with v. Let $h(w) = \{i_1, i_2, \cdots, i_k\}$ be unique identifiers corresponding to active bases $b_{i1}, b_{i2}, \cdots, b_{ik}$ with associated coefficients $w_{i1}, w_{i2}, \cdots, w_{ik}$ respectively. Then $h(w)$ is called an SCT2 if $|w_{i1}| \geq |w_{i2}| \geq \cdots |w_{ik}|$. That is, the basis identifiers are arranged in* monotonically decreasing order *of their absolute sparse coefficients.*

Remark 31. *If the choice of our active set cardinality adheres to Theorem 26, then both SCT1 and SCT2 will be equivalent. Unfortunately, since we use an empirical estimate of this cardinality, there is still a chance that the magnitudes of the associated sparse coefficients reduce along the regularization path. To overcome this issue, we need to take all possible permutations of the indices in SCT2 and query the respective hash buckets. This can in fact be done in parallel, later sorting the fetched neighbors for the nearest ones. This approach might not scale when the size of the support set is large (fortunately, as we will see later, we work with smaller support sets).*

The following Lemma will be useful in our future analysis.

Lemma 32. *Let $w = \mathrm{SC}_{CBP}(\delta; v)$ be the sparse code associated with a data vector v using a dictionary $\mathcal{B}$ and reconstruction error bound δ. Then $w = 0$, if $\delta \geq \|v\|_2$.*

Proof. Recall that for LARS steps k and $k+1$, we have the following relation with regard to the reconstruction errors (see Proposition 28).

$$\|v - \mathcal{B}w^{k+1}\|_2 < \|v - \mathcal{B}w^{k}\|_2. \tag{7.4}$$

Now we know that LARS proceeds its loop until the error bound $\|v - \mathcal{B}w\|_2 \leq \delta$ is satisfied. From (7.4), it is clear that every step of LARS will generate smaller errors. So the maximum error that is possible is when $w = 0$, which corresponds to $\|v\|_2$. Thus, for any $\delta > \|v\|_2$, the error bound condition will be satisfied and LARS will produce $w = 0$. □

Recall that in Proposition 28 we showed that by adjusting the LASSO regularization (or the bound t or δ), we can control the sparsity of the solution, i.e. the length of the SCT hash code. The next theorem connects the regularization path of the LASSO solution produced by LARS, with matching SCT codes. This will lead us to develop the Multi-Regularization Sparse Coding for generating robust SCTs.

Theorem 33. *Assume v_1, v_2 are two zero-mean unit norm data vectors, and $\mathcal{B}$ is a given dictionary. Further, let μ is the coherence of $\mathcal{B}$, ($\mu(\mathcal{B}) < 1$). Suppose $T_i = h_{(CBP;\delta)}(v_i)$ for $i \in \{1, 2\}$ are the two SCTs (SCT1) found by LARS for a reconstruction error bound δ. If $T_1 \neq T_2$, then there exists a δ', where*

$$\delta < \delta' \leq \max\left(\|v_1\|_2, \|v_2\|_2\right)$$

such that the new SCTs $T_i' = h_{(CBP;\delta')}(v_i)$ will satisfy,

I. $T_1' \subseteq T_1$ *and* $T_2' \subseteq T_2$

II. $T_1' = T_2'$

iff there exists a $b_i \in \mathcal{B}$ such that $|v^T b_i| < \sqrt{\frac{1}{2}(1+\mu)}$, $\forall v \in \{v_1, v_2\}$.

Proof. Considering each case:

I. This is a direct result from the continuity properties of the LASSO as implied by Proposition 28, when Condition 1 is satisfied.

II. Let us define a θ such that $cos\theta = \mu$ so that $\sqrt{\frac{1}{2}(1+\mu)} = cos\frac{\theta}{2}$. To prove the *if* part: Suppose $\exists b_i \in \mathcal{B}$ such that $|b_i^T v| < \sqrt{\frac{1}{2}(1+\mu)}$, then $\widehat{b_i, v} < \frac{\theta}{2}$. This condition means that b_i is the most correlated dictionary atom to v and thus will be selected by LARS in the first step. If not, there should exist another basis b_j such that this condition is satisfied. In which case, $b_i^T b_j < \mu$, which contradicts the condition that the coherence of the dictionary is μ. Now, using the definition of SCT1, combined with Proposition 28 and Lemma 32, we have that $\exists \delta'$ such that $\delta < \delta' < max(\|v_1\|_2, \|v_2\|_2)$ which satisfies $T_1 = T_2$.

To prove the *only if* part: Assume that there exists a $0 \leq \delta < max\left(\|v_1\|_2, \|v_2\|_2\right)$ so that $T_1 = T_2$. Let $T_1 = T_2 = \{i1, i2, \cdots, ik\}$ be the respective SCT. Then by the definition of SCT1, we have that $i1$ is the most correlated basis to the data v and thus using Lemma 30 will be selected at the first step of LARS, from which the result follows.

□

Theorem 34. *Let $\theta = \max_{i,j} \widehat{b_i, b_j}$ for $b_i, b_j \in \mathcal{B}$. If $\widehat{v_1, v_2} > \theta$, then $T_1 \neq T_2$ with probability one.*

Based on the above analysis, now we are ready to present our robust sparse coding algorithm.

7.3.2 Multi-Regularization Sparse Coding

We have from Theorem 33 that, if the SCT representations for two data vectors that make an angle less than $\theta/2$ with the same basis do not coincide at one error bound, then there should exist a larger bound at which the two SCTs will accord. This implies that if we use multiple SCT codes corresponding to a set of bounds, then at least one of them will match up with a query SCT. This is exactly the idea in the Multi-Regularization Sparse Coding algorithm we propose below. There is one key difference though; instead of running the LARS algorithm in multiple passes for each bound, we make a single pass, keeping track of the index of active basis support at each LARS iteration, incrementally building the SCT. This is theoretically equivalent to running the algorithm for different bounds. Using this idea, we can create multiple SCTs for a sequence of support sizes: $\mathcal{L} := \{L_{min}, L_{min}+1, L_{min}+2 \cdots, L_{max}\}$. With a slight abuse

of our notation, we will denote a k dimensional SCT using our dual LASSO formulation as $h_{(CBP;k)}(v)$. Using this notation, for each data vector v, we define the *hash code set* $H_{CBP,\mathcal{L}}(v)$ as follows:

$$H_{CBP,\mathcal{L}}(v) := \{h_{(CBP;L_{min})}(v), h_{(CBP;L_{min}+1)}(v), \cdots, h_{(CBP;L_{max})}(v)\}. \tag{7.5}$$

Algorithms 4 and 5 show the essential steps of our MRSC scheme.

Algorithm 4 MRSC Hashing

Require: $\mathcal{B}$, $\mathcal{D}$,
1: Initialize hash table $\mathcal{H}$
2: **for** $v_i \in \mathcal{D}$ **do**
3: **for** $\ell \Leftarrow \{L_{min}, \cdots, L_{max}\}$ **do**
4: $w \Leftarrow \mathrm{SC}_{\mathrm{CBP}}(\ell; v_i)$
5: $T \Leftarrow h_{(CBP;\ell)}(v_i)$
6: $\mathcal{H}(T) \Leftarrow \mathcal{H}(T) \cup w\{$ assign w to a hash bucket associated with $T\}$
7: **end for**
8: **end for**
9: **return** $\mathcal{H}$

Algorithm 5 MRSC Query

Require: $\mathcal{B}$, $\mathcal{H}$, v_q, $\mathrm{NN}_q = \{\}$
1: Initialize $\mathrm{NN}_q \Leftarrow 0$
2: **for** $\ell \Leftarrow \{L_{max}, \cdots, L_{min}\}$ **do**
3: $w_q \Leftarrow \mathrm{SC}_{\mathrm{CBP}}(\ell; v_q)$
4: $T \Leftarrow h_{(CBP;\ell)}(v_q)$
5: **if** $\mathcal{H}(T) \neq \{\}$ **then**
6: $w^* \Leftarrow \mathrm{argmin}_{w \in \mathcal{H}(T)} \|w - w_q\|_2$
7: $\mathrm{NN}_q \Leftarrow v^*$ associated with w^*
8: **return** NN_q
9: **end if**
10: **end for**
11: **return** NN_q

The *MRSC Hashing* algorithm works as follows: for every data point, it computes the L_{max} sparse coefficients via the LARS algorithm. The coefficients are sorted and their indices used to create the SCT (SCT2 as defined previously). This SCT is hashed into the inverted file system after compression into a bit stream. With this hash code is stored the sparse coefficients

associated. We do the hashing $(L_{max} - L_{min} + 1)$ times into the same hash table, each time producing an SCT with dimension one more than the previous step, beginning from L_{min}. To query the NN for a given data point, we use the *MRSC Query* algorithm, which works the same way as the previous algorithm, except that the SCT is queried in the reverse order, that is, we query the SCT with L_{max} indices first, and if we encounter an empty bucket, then the indices are removed one at a time, until a sufficient number of NNs are retrieved from the dataset.

7.4 Empty Hash Bucket

The third problem that we need to consider when using sparse coding for NN is to answer the question of 'where to look for next' if we find a hash bucket to be empty. This is partially explained in the previous section through using multiple sparse codes. The question still to be answered is where to query if the database point and the query point make angles with each other that is greater than the θ of Theorem 34. In the following subsections, we introduce *Cyclic SCTs* for this purpose.

7.4.1 Cyclic Subspace Combination Tuples

It often happens that the database point and the query point are noise corrupted versions of an underlying true data point. Let v be a noisy data point and let $T = \langle i1, i2, \cdots, ik \rangle$ be the SCT associated with T. Assume $j^* = \text{argmax}_{j \in T} \widehat{b_{ij}, v}$ denotes the maximum angle that the data point makes with any basis in the active support. Then, as long as for a noise component ϵ such that $v' = v + \epsilon$, $\widehat{v', b_{ij^*}} \leq \theta$, $T \cap T' \neq \{\}$, where T' is the SCT for v'. A schematic illustration of this idea is suggested in Figure 7.2. A data point A and a query point B, are assumed to be noise corrupted versions of each other such that the noise displaces the spatial location of A to B. Note that point A has $SCT_A = \langle j, k \rangle$, while point B has a $SCT_B = \langle k, i \rangle$.

Motivated by this idea, we propose a *cyclic SCT*, in which, instead of generating a single MRSC, we do a cyclic rotation of the MRSC hash codes, making each of the basis as being the first entry in the respective SCT cycle, so that the MRSC algorithm queries neighboring subspaces for a given query rather than constraining only the subspace majored by the most correlated basis to the query point. If the length of SCTs are short, such that computing all permutations of the indices in an SCT is not computationally costly, then we can in fact look at every permutation of the MRSC hash code. This may provide a faster search of the subspaces.

Further, such a heuristic can be implemented in parallel for increased speed up. After retrieving the neighbors from each of the SCTs in the cycle (or permutations), the NNs are decided by sorting the candidate neighbors. Referring back to Algorithm 5, Cyclic MSRC will have an additional loop at line#5, where the T is rotated (or permuted).

7.5 Algorithm Analysis

Dictionary Learning Complexity Let us assume that we use the K-SVD algorithm to learning a dictionary $\mathcal{B} \in \mathbb{R}^{d \times n}$ using a training set $\mathcal{D}$. If we assume that each of the data vector $v \in \mathcal{D}$ is k-sparse in the dictionary being learned, then the time taken per iteration for DL is roughly $|\mathcal{D}|(k^2 + 2d)n$ FLOPS [Rubinstein et al., 2008].

Sparse Coding Complexity Let us assume that we use a maximum of k iterations of the LARS algorithm for sparse coding a data point $v \in \mathbb{R}^d$ using a dictionary $\mathcal{B} \in \mathbb{R}^{d \times n}$. Using Cholesky updates, it can be shown that the equiangular directions can be computed in j^2 FLOPS for the j^{th} LARS step, leading to $\mathcal{O}(k^3)$ computational time for k steps. For computing k maximally correlated dictionary atoms (one in each LARS step) we need ndk FLOPS. Thus, the total computational complexity is $\mathcal{O}(k^3 + ndk)$. As smaller codes ($k \leq 5$) are sufficient for generating the hash codes, the computation is dominated by the component $\mathcal{O}(ndk)$, which is the cost of making k inner products with the entire $\mathcal{B}$.

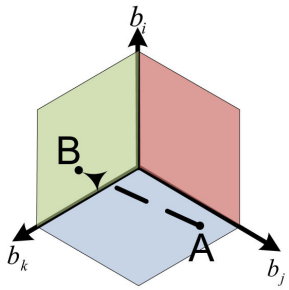


Figure 7.2: Illustration of the cyclic MSRC algorithm: assuming an orthogonal subspace for clarity of presentation, a data point denoted by A inhabiting a subspace spanned by bases b_j and b_k gets displaced due to noise to a spatial location B spanned by the subspaces b_i and b_k.

Space complexity For k-sparse data points, we will need k integers to represent the hash codes corresponding to the indices of the dictionary and k floating point numbers to store the coefficients in memory. Assuming each basis is equally likely[1], we will need $k\log(n)$ bits for hashing. Suppose each floating point number is stored in s bits, then we require a total of $k(\log(n)+s)$ bits for storage per data point. To give an idea of our requirements, for the SIFT dataset in our experiments, we use 2048 basis. Suppose we use OSTs of length 5, we will need 55 hash bits for our OST scheme and 160 bits to store 32 bit floating point numbers.

Query Complexity Assuming the data $\mathcal{D}$ is uniformly distributed into the hash buckets by the dictionary $\mathcal{B} \in \mathbb{R}^{d\times n}$, and we use linear search to resolve hash table conflicts. Then, for an OST of length L_{max}, the best query time Q_{time}^{best} is bounded by

$$Q_{time}^{best} = \frac{|\mathcal{D}|}{2^{L_{max}}\binom{n}{L_{max}}L_{max}!}. \tag{7.6}$$

Now, let us assume that we do not find an NN with an L_{max} size OST. Then, we need to continue the search using an OST of length $L_{max}-1$, and so on. Suppose we have a minimum OST length of L_{min}. Then, it can shown that the worst case query complexity is upper bounded by:

$$Q_{time}^{worst} = \frac{|\mathcal{D}|}{2^{L_{min}}n!}\left[(\Delta+1)(n-L_{min})!\right], \text{ where } \Delta = L_{max}-L_{min}. \tag{7.7}$$

Note that, for smaller OST lengths, we will have more hash bucket collisions, slowing down the query. In such a case, we can use a kd-tree to organize the data for faster access. As we store only low dimensional sparse codes associated with the data, we do not need to worry about the curse of dimensionality.

7.6 Experiments

In this section, we detail our experimental setup; the datasets used and the performance metrics against which the methods are evaluated. Our algorithms were implemented in C++ for speed, with MATLAB interfaces for ease of data handling. All the algorithms, except the one on the retrieval speed for an increasing database size, used a 2GHz 64bit machine with 4GB RAM.

[1] This assumption is strictly *not* true. It is shown in [Ramirez et al., 2009] that the probability of each basis getting activated is not uniform. In such a case, we can use an adaptive encoding scheme (using techniques such as Huffman codes) to reduce the number of hash bits even further.

7.6.1 Datasets

SIFT Dataset: Our experiments were primarily based on the SIFT descriptors created from the INRIA Holidays and the INRIA Copydays datasets[2]. We used approximately 400M SIFT descriptors created from 200K images from these datasets.

Spin Image Dataset: 3D shape recognition and retrieval has recently been gaining a lot of attention due to the introduction of cheap 3D sensors. Since these sensors produce millions of 3D points per video frame, retrieving corresponding points across frames for object recognition is a challenging operation. One of the most popular 3D point cloud descriptors is the *spin image* [Johnson, 1997]. Each spin image is a 2D histogram of the changes in the 3D surface structure of the point cloud, computed in cylindrical coordinates, around a given point. We used a 20×20 histogram, thus forming spin vectors of 400D for each 3D point in the cloud. The descriptors were generated from the public SHREC dataset[3] [Boyer et al., 2011], consisting of 50 object classes, of which we used 10 classes for computing the spin images, amounting to a total of 1M descriptors.

7.6.2 Precision/Recall@K

We use the following definition of Precision@K and Recall@K for the performance analysis. Given a dataset $\mathcal{D}$ and a query set $\mathcal{Q}$, we define:

$$Recall@K = \frac{1}{|\mathcal{Q}|} \sum_{\forall q \in \mathcal{Q}} \mathrm{NN}_K^{\mathcal{A}}(q, \mathcal{D}) \cap \mathrm{NN}_1^{gt}(q, \mathcal{D}) \tag{7.8}$$

where $\mathrm{NN}_K^{\mathcal{A}}$ retrieves the K nearest neighbors of a query point q from $\mathcal{D}$ using the respective algorithm $\mathcal{A}$ and $\mathrm{NN}_1^{gt}(q, \mathcal{D})$ is the ground truth nearest neighbor computed using linear scan. This definition of recall has been used in many papers such as [Jegou et al., 2011] as it provides a sense of approximately how many NNs have to be retrieved by a given algorithm so that there is a good chance that the true NN is retrieved. Similarly, we define Precision@K as follows:

$$Precision@K = \frac{1}{|\mathcal{Q}|} \sum_{\forall q \in \mathcal{Q}} \mathrm{NN}_1^{\mathcal{A}}(q, \mathcal{D}) \cap \mathrm{NN}_K^{gt}(q, \mathcal{D}) \tag{7.9}$$

[2] http://lear.inrialpes.fr/ jegou/data.php
[3] http://www.itl.nist.gov/iad/vug/sharp/contest/2011

where NN_K^{gt} retrieves the K nearest neighbors of a query point q from $\mathcal{D}$ using linear scan and $\mathrm{NN}_1^{\mathcal{A}}$ is the nearest neighbor found by the algorithm $\mathcal{A}$.

7.6.3 Dictionary Learning

Learning the overcomplete basis dictionary is the first step in sparse coding. Deciding the number of bases is a critical component with regard to the accuracy and speed of sparse coding and NN retrieval. A dictionary with a large number of bases could increase the coherence between the atoms such that the sparse codes become too sensitive to noise or can even become ambiguous (coherence when unity). Too few dictionary atoms might result in less expressive SCTs. Thus, in this chapter we approach the problem from a data driven perspective and use cross-validation against recall@1 to decide the optimum dictionary size.

Towards this end, we randomly chose 1M SIFT descriptors from the dataset. The dictionary learning was performed using the SPAMS toolbox [Mairal et al., 2010] for an increasing number of bases. Typically, SIFT descriptors have integer dimensions that pose significant problems with the standard dictionary learning algorithms, especially the ill-conditioning of the matrices involved. Thus we scaled down all the descriptors by dividing them by 255, and later normalizing each descriptor to have zero mean. Figure 7.3 shows the recall@1 performance on these normalized descriptors for varying dictionary sizes on a small database of 10K descriptors and a query size of 1K. The best recall was observed for a dictionary of size 2048 for the SIFT descriptors and thus we use this dictionary for subsequent experiments in this chapter. Sparse coding a SIFT descriptor using a 2048 bases dictionary took approximately 100μs on an average. We fixed the LASSO regularization to 0.5 for dictionary learning. As for the spin image dataset, we followed suit and learned a dictionary of size 400×1600 using 100K descriptors from the dataset.

7.6.4 Active Support Size

Another important parameter that needs to be estimated in our algorithm is the size of the active support. As was seen in Section 7.2, an active set size less than 10 provides robustness against problems such as the basis drop in the LARS regularization path. To estimate the recall performance of NN against various support sizes, we did the following cross-validation experiment. On a subset of our SIFT dataset consisting of 1M SIFT descriptors, we computed the average

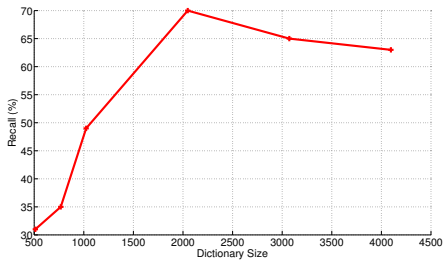


Figure 7.3: A plot of the recall performance of the SIFT descriptors for varying sizes of the dictionary. The dataset was of size 10K sparse-SIFT descriptors and the query had 1K descriptors. The dictionaries were learned on 1M SIFT descriptors.

recall@1 using 1K SIFT descriptors for two cases: (i) using the SCT generated hash codes and (ii) using the MRSC algorithm for the active support size varying from 2 to L (recall that the latter experiment is required in case an empty hash bucket is encountered). Figure 7.4(a) and Figure 7.4(b) plot both these cases respectively. Figures 7.4(c) and 7.4(d) plot the average query times for both the cases.

As is clear from the plots, an active set size in the range of 4–7 provides the best recall and the best query time. The exact match of the query SCT with the ground truth happens less than 30% of the time and thus the recall as shown in Figure 7.4(a) is low. On the other hand, using multiple regularizations helps address the issue of empty hash buckets and thus improving the recall as the size of the support increases, as shown in Figure 7.4(b). Although the query time goes down as the size of the hash code increases as shown in Figure 7.4(c), it increases after a certain point (L=5) in the MRSC algorithm. As suggested from the above plots, we decided to choose the L value in the range of 2–5 in this chapter. A point to note here is that the recall goes low when L=2. This is because, there will be too many hash collisions and the number of coefficients compared against to resolve the collision is too small.

7.6.5 Experimental Results

In this subsection, we compare the recall performance of the MRSC algorithm (cyclic MRSC) against the state-of-the-art techniques. We chose six other popular NN techniques to compare against: (i) Product Quantization (PQ) algorithm [Jegou et al., 2011], (ii) KD-Trees (KDT), (iii) Euclidean Locality Sensitive Hashing (E2LSH) based on [Datar et al., 2004], (iv) Kernelized LSH (KLSH) [Kulis and Grauman, 2009], (v) Spectral Hashing (SH) [Weiss et al., 2009], and

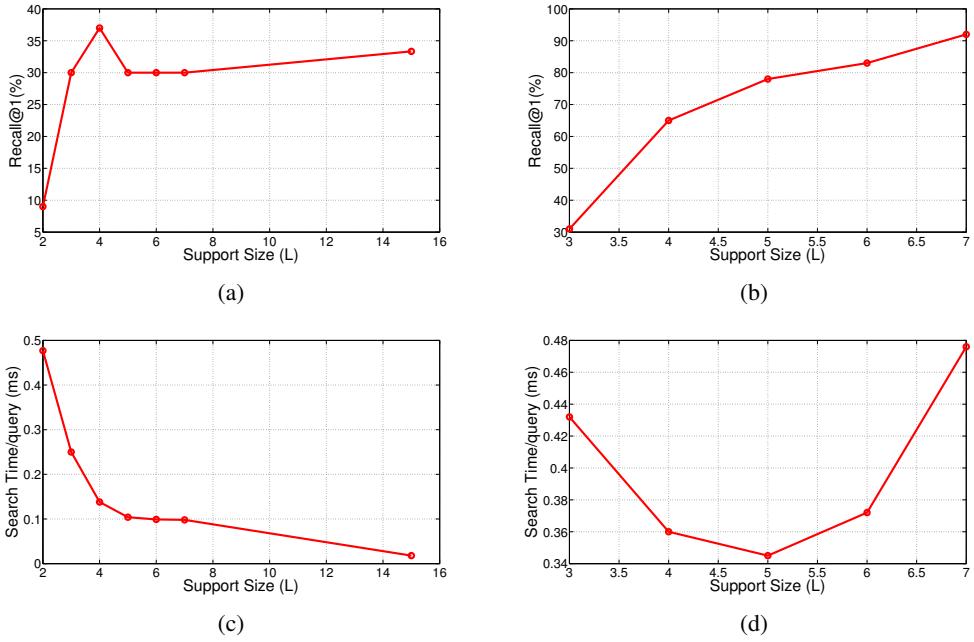


Figure 7.4: (a) Recall for an increasing support size, (b) Recall using the MRSC algorithm for the support size varying from 2 to L (L being varied), (c) Query time against increasing support size, (d) Query time for the MRSC algorithm for support size varying from 2 to L. The experiment used 1M SIFT descriptors and the performance averaged over 1K query descriptors.

(vi) Shift Invariant Kernel Hashing (SIKH) [Raginsky and Lazebnik, 2009]. Spectral hashing used 40 bit hash codes as it was observed that higher number of bits reduced the recall performance (this is expected as SH internally uses a PCA step and the sensitivity of the higher order bits increases as the length of the hash code increases since these higher order bits correspond to smaller eigenvalues). We used 40 bits for SIKH as well. Other algorithms used the default parameters as provided by their implementations. For KLSH, we used the RBF kernel and used 1K descriptors randomly selected from the dataset for learning the kernel. As for the PQ algorithm, we used the IVFADC variant as it seemed to show the best performance both in speed and accuracy.

Recall on SIFT

We used approximately 2M SIFT descriptors as the database and 1K query points. The average recall performance after 10 trials for an increasing number of retrieved points is shown

in Figure 7.5(a). As is clear, the MRSC algorithm outperforms all other algorithms in recall, especially for recall@1. Although, SIKH performs very close to MRSC, the performance of other algorithms especially the KLSH was found to be very low. We found that the selection of the training set for building the kernel had a major impact on the performance of this algorithm. Thus, we show the best performance of KLSH that we observed from 10 different kernel initializations.

Recall on Spin Images

We used approximately 1M spin images (400D) as the database and 1K descriptors, excluded from the dataset, as the query points. Since the KDT algorithm needs to store all the data points in memory, we did a dimensionality reduction of these descriptors to 40D descriptors through a PCA projection as suggested in [Johnson, 1997]. The average recall performance after 10 trials is shown in Figure 7.5(b) for an increasing number of retrieved points. As we saw with the SIFT dataset, MRSC seems superior to all the other algorithms in the quality of retrieved points.

7.7 Image Search Qualitative Results

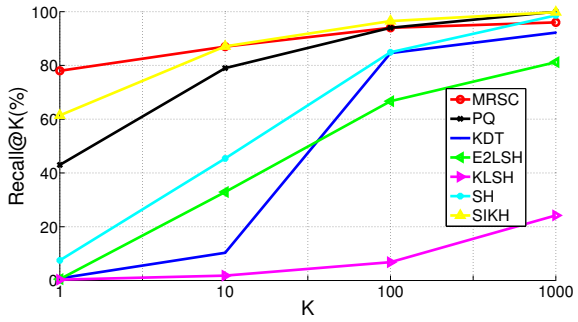
7.7.1 Notre Dame Dataset

As explained earlier, SIFT feature correspondence is an essential ingredient in applications such as Photosynth[4] that creates realistic photo walk-throughs in virtual environments. Given a query image, the idea is to extract all the images in the database that contain the SIFT descriptors associated with the query.

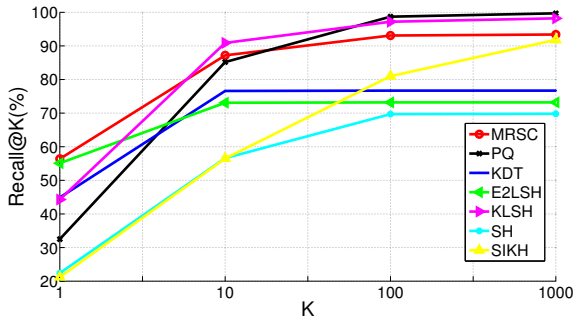
For this experiment, we used the Notre Dame dataset[5]. This dataset consists of 1500 high resolution photos downloaded from the Flickr website. Figure 7.6 shows a few sample images from this dataset. SIFT descriptors extracted from the images formed the database on which the MRSC algorithm to serve image queries. We used a *maximum voting* strategy to find the NN images to the query. That is, NN was decided by those images that had the maximum number of SIFT NN matches with respect to the SIFT descriptors from the query image. There were approximately 6M SIFT descriptors created from this dataset, from which we randomly picked 300K descriptors as the SIFT database for computational reasons. Figure 7.7 shows a

[4] http://photosynth.net/

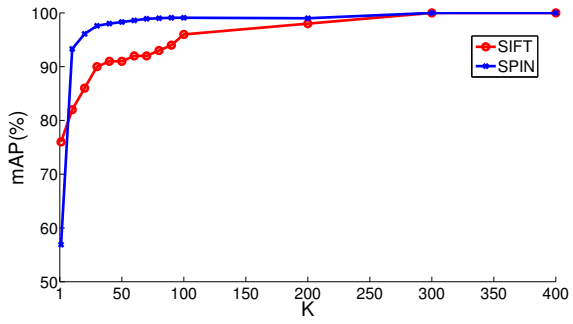
[5] http://phototour.cs.washington.edu/datasets/



(a) SIFT



(b) Spin Images



(c) Precision

Figure 7.5: Recall@K for SIFT (Figures 7.5(a) and 7.5(b)) compared against the state-of-the-art methods respectively. The comparisons were carried out on 2M points from the SIFT and 1M spin image datasets respectively, and the recall averaged over a query size of 1K descriptors averaged over 10 trials. Figure 7.5(c) plots the mean average precision for the SIFT and spin image datasets.

few qualitative results from the experiment. The first image in each row of Figure 7.7 shows the query image, while rest of the images are the first four NNs returned by the MRSC algorithm.

Figure 7.6: Sample images from the Notre Dame dataset.

7.7.2 Perceptual Image Search

Next, we show an application of MRSC algorithm for the task of retrieving perceptually similar images. Perceptual search is an abstract and subjective idea. Here, we define *perceptual image search* as images in the database that belong to the same category as the query image with respect to the color distribution, object shapes, appearances, and spatial organization of the image. For this application, we tried a slightly different approach to dictionary learning (compared to the SIFT based retrieval we discussed in the last section). Instead of learning dictionaries on feature descriptors, we learned the dictionary on the image as a whole. That is, each image (along with the three color channels) was vectorized into a long vector, on which large dimensional dictionaries are then learned. The basic motivation for this experiment was two fold: (i) we wanted to understand if such direct vectorization and dictionary learning automatically learned interesting features from the dataset, and (ii) we wanted to test our sparse coding based NN retrieval on high dimensional datasets.

For this experiment, we use the tiny images dataset which contains approximately 80M images. Each image in this dataset is of size 32×32. Unfortunately, we found that working

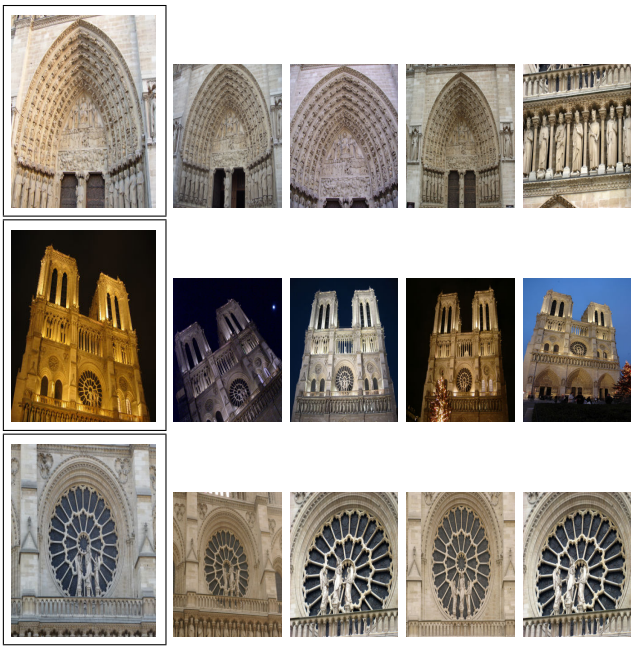


Figure 7.7: Real-world image search application. A query image (in black box) and retrieved nearest neighbors from the NotreDame 1500 image dataset is shown.

directly on vectorized tiny images was impractical for the dictionary learning set (direct vectorization results in each data point to be 3072D, and learning a dictionary that is overcomplete seemed computationally difficult). Thus, for the dictionary learning to be computationally easier, we resized each image by half, vectorized them creating 256 dimensional vectors for each color dimension, which are later combined to form a 768 dimensional feature vector representing each image. We used 1M arbitrary images from this dataset to learn a dictionary of size 768×3072. Later, we selected an arbitrary 10M images from the dataset as the database, sparse coded each data point in this database and applied the MRSC algorithm for a query set of size 1K. The statistics of the experiment are shown in Table 7.1. A few query results are shown in Figure 7.8. The first item on the left of each image block shows the query image, followed by the first column showing the first four NNs returned by the MRSC algorithm, and the second column showing the results returned by a cosine similarity baseline. From the qualitative images it seems that the MRSC retrieves perceptually similar images, although validating this observation quantitatively is hard (as the baseline itself may be subjective).

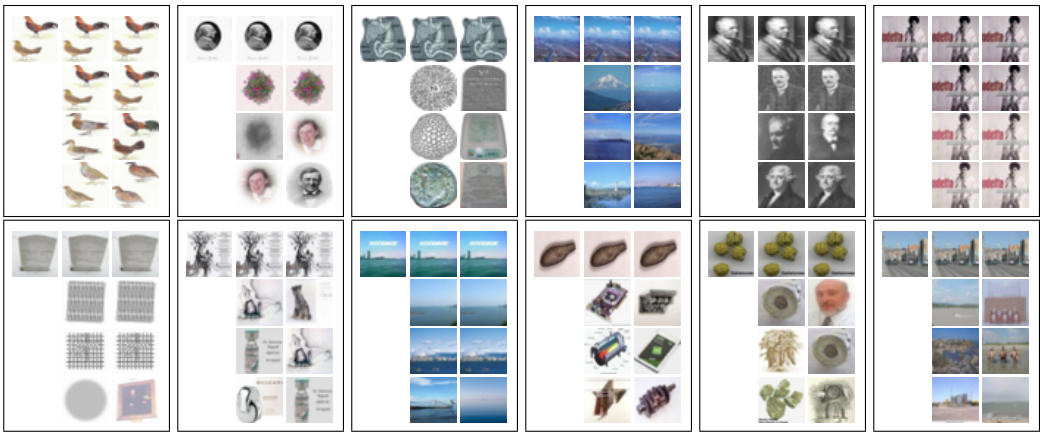


Figure 7.8: Perceptual similarity search on the tiny images dataset. For each block above (inside the box), the query image is shown (first column), followed by the top four results returned by the MRSC algorithm (second column), and those returned by a linear scan (third column).

Statistic	*Value*
Avg. hash table collusions	201.89
Max. items in hash bucket	5437 ($< 1\%$ of dataset)
Mean # active basis used	4.3
Avg. Search time/query	0.04 ms

Table 7.1: Statistics of the hash table used on the Tiny images dataset (10M color images each resized to 16×16 sparse coded using a dictionary of size 768×3072.

7.8 Webscale Datasets

7.8.1 Retrieval Speed

Scalability to large datasets is an important property that any NN algorithm should posses if targeted to work at webscale. In this subsection, we show NN retrieval performance of the MRSC (cyclic) against an increasing database size. For this experiment we used the 400M SIFT dataset mentioned previously. The dataset was gradually incremented from 1M to 400M, each time 1K descriptors were randomly chosen from the dataset (and thus excluded from the database) and recall@1 computed. Figure 7.9(a) plots the average search time per query for our algorithm compared against the PQ algorithm. Since the optimized implementation of the PQ algorithm is licensed, we do the comparison against the results reported in [Jegou et al., 2011] using the same hardware platform as suggested by the authors. Clearly, as seen from the plot,

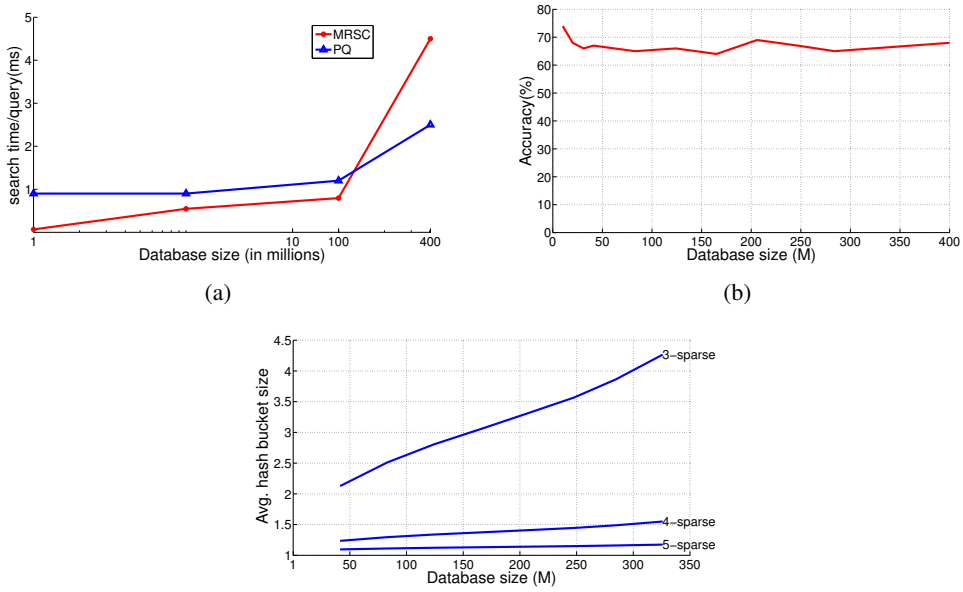


Figure 7.9: 7.9(a) plots the search time (in ms) per query for increasing database size, 7.9(b) plots the recall@1 for increasing database size, and 7.8 plots the average hash bucket size against increasing database size. All the experiments were conducted on SIFT dataset.

our implementation rivals the state-of-the-art in search speed.

7.8.2 Recall Performance

Next, we compared the recall@1 performance of MRSC against an increasing database size. As increasing the database leads to increased hash table collisions, the purpose of this experiment was to analyze how good the approximate sparse Euclidean distance computation was. We used the same 400M SIFT dataset and did the recall@1 for 1K descriptors by gradually increasing database size as was done in the last experiment. Figure 7.9(b) shows the result. Even though there is a slight drop in the recall as the database size increases, the subsequent drop is minimal and on an average maintains a recall almost close to the recall@1 performance as seen in Figure 7.5(a). This is explained by the Figure 7.8 which shows that as the database size increases the new points fall onto new hashbuckets such that the relative average increase in the size of the existing buckets is minimal. The 2-sparse plots are omitted from the Figure 7.8 as they grow at a faster rate than the other support sizes. For example, for the 2-sparse case, we have an average bucket size of 27.36 for 40M SIFT descriptors, while this increases to 143.3 for 400M

descriptors.

7.8.3 Precision

Finally, we compare the Precision@K for the SIFT and spin image datasets. For this experiment, we used the same setup as the one for the recall (as discussed in Section 7.6.5), except that instead of comparing K NNs found by the algorithm against one ground truth, we compared the first NN found by the algorithm against K nearest neighbors from the dataset as found using Euclidean linear scans. Figure 7.5(c) plots the mean average precision (mAP) computed on 1K SIFT and spin image datasets averaged over 10 trials. As is clear from the plot, the quality of the NNs retrieved by the cyclic MRSC algorithm lies in the top 50 true nearest neighbors for more than 90% of the queries.

7.8.4 Robustness to Distortions

SIFT descriptors are commonly employed for finding feature correspondences across images in applications such as motion estimation, 3D reconstruction, etc. The images used to compute the correspondences are generally distorted in multiple ways and thus robust computation of the matching descriptors is essential for the task. In this section, we explicitly evaluate the robustness of the MRSC algorithm to compute SIFT correspondences across images undergoing various commonly seen distortions. For this experiment, we use the SIFT benchmark dataset[6], consisting of eight categories of images; each category containing six images exhibiting a specific type of distortion. The distortion categories are: (i) *BARK*, with images undergoing magnification, (ii) *BIKES*, with fogged images, (iii) *LEUVEN*, with illumination differences, (iv) *TREES*, with Gaussian blur, (v) *BOAT*, with rotation, (vi) *GRAPHIC*, with 3D camera transformation, (vii) *UBC*, granularity introduced by JPEG compression, and (viii) *WALL*, with affine transformations.

SIFT descriptors were computed for each image in a category. The descriptors from the first image in each category were used as the query set, while the database at step i was formed by the descriptors from image i, with i varying from 2 to 6. The recall@1 performance was computed against a ground truth using Euclidean linear scan. As there was too much of distortion in the images for $i = 6$ in each category, we did a slight relaxation of the nearest neighbor criteria

[6] www.robots.ox.ac.uk/ vgg/research/affine/index.html

such that we declared a candidate point to be a neighbor if it was inside an ϵ-neighborhood of the true neighbor, where we used $\epsilon = 1.12$ (or 90% neighbor). We also pruned out ground truth correspondences that were greater than a threshold in the Euclidean distance, as there was a high chance of them being false positives. We compared the performance against three other state-of-the-art algorithms in NN retrieval for the task: KLSH, KDT and PQ, with the same analysis configurations. The result of this experiment is shown in Figure 7.10 with the recall averaged over all the queries in a category. The plots clearly show the robustness of cyclic MRSC against other approaches.

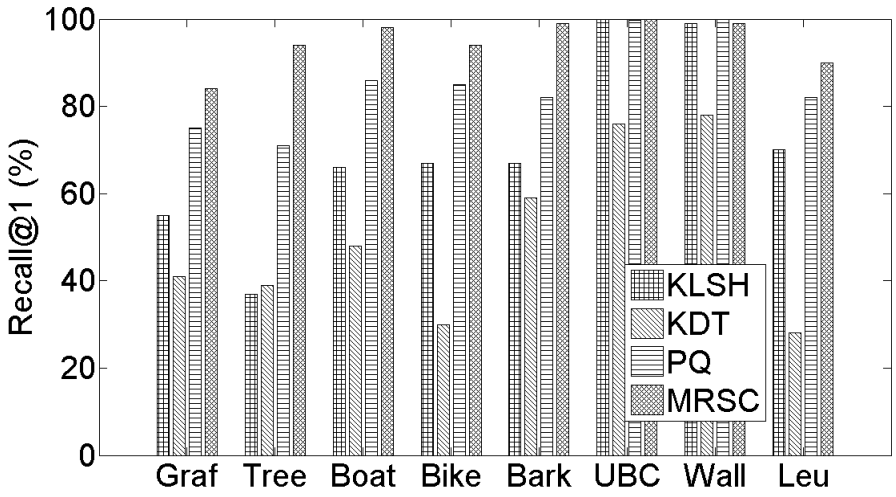


Figure 7.10: Recall@1 for SIFT descriptors undergoing various image distortions.

Part III

Sparse Coding for Covariance Valued Data

Chapter 8

Nearest Neighbors on Covariances via Sparse Coding

The first part of this thesis looked at the problem of nearest neighbor retrieval on covariance valued data. Our algorithms were primarily based on an n-ary tree data structure (such as the metric tree) and fast ANN retrieval used well-established heuristics such as the best-bin-first. In the second part of this thesis, we looked at hashing algorithms on vector valued data by extending the dictionary learning and sparse coding framework. Our algorithm showed superior performance against well-known algorithms both in retrieval speed and accuracy. Assuming the hash functions are chosen appropriately, hashing provides constant time data access compared to the logarithmic retrieval times of the n-ary data structures. An obvious question that arises here is if our sparse code based hashing framework can be extended for hashing covariances. In this chapter, we will look at ways to answer this question and propose our *Generalized Dictionary Learning* (GDL) framework. The basic idea is to approximate the covariances as the sum of rank-one positive semi-definite matrices. To this end, we will propose a new rank-one dictionary learning formulation and efficient algorithms for solving it. Our subspace combination tuple coding scheme is extended to deal with sparse rank-one approximations of the covariances. Experiments show the efficacy of our algorithm against other competitive ANN algorithms on covariances.

8.1 Related Work

Our GDL framework is based on the rank-one approximations to the covariance valued data descriptors. Such a scheme has been tried in several contexts before. In this section, we will review some of the works that are closely related to ours. In [Shen et al., 2008; S. et al., 2010], the standard Mahalanobis type metric learning framework is extended by approximating the positive definite Mahalanobis inverse covariance matrix as a linear combination of rank-one positive semi-defnite matrices. The approach and application of this chapter is different in that they have just one covariance matrix to appoximate, while we deal with a dataset of covariances. A paper that is closer in spirit to our application and approach is [Sivalingam et al., 2010]. The paper deals with covariance valued data and proposes a positive definite dictionary learning framework that sparse codes the data covariances. Since the problem is difficult to optimize, the paper suggests reducing the problem to the well-known MAXDET formulation that enables reformulating the problem as a Semi-Definite Program (SDP) for which efficient solvers exist. On the negative side, the SDP formulation poses significant implementation issues when working with larger datasets or with high dimensional covariances. Finally, we note that in the compressed sensing community, optimizing over matrix-valued data by minimizing the trace norm (nuclear norm) [Cai et al., 2008], [Liu and Vandenberghe, 2009], [Candes and Plan, 2010] 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 this dictionary, as elucidated in Section 8.3 below.

8.2 Covariances and Rank-one Approximations

In this section, we systematically extend the framework of DL to the space of SPD matrices. Let $s \in \mathbb{R}^d$ be some zero mean data vector [1] from the dataset. Further, assume $\mathcal{B} \in \mathbb{R}^{d \times n}$ is a sparsifying dictionary on these feature vectors, such that $\mathcal{B} = (b_1, b_2, \cdots, b_n)$ where $b_i \in \mathbb{R}^d$. Then, going by the principles of dictionary learning and sparse coding, if s is sparse in $\mathcal{B}$, then

[1] Assume that we know the global mean of the distribution of feature vectors.

we can find the sparse approximation of s in $\mathcal{B}$ by minimizing,

$$\underset{c\in\mathbb{R}^d}{\operatorname{argmin}} \|c\|_1 \text{ such that } s = \mathcal{B}c = \sum_{i=1}^{n} c_i b_i \tag{8.1}$$

where c is the vector of sparse coefficients. Now, we see that

$$ss^T = \left(\sum_{i=1}^{n} c_i b_i\right)\left(\sum_{i=1}^{n} c_i {b_i}^T\right) \tag{8.2}$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j^T b_i b_j^T \tag{8.3}$$

$$= \sum_{i=1}^{n^2} z_i \mathcal{D}_i \tag{8.4}$$

where each $\mathcal{D}_i = b_i b_i^T$ and z_i's are scalar coefficients. Note that, since s is sparse in $\mathcal{B}$, so will be vector $z \in \mathbb{R}^{n^2}$.

Going further, a covariance matrix S created from a set of r zero-mean data vectors s_i $(i = 1, 2, \cdots, r)$ can be written as

$$S = \frac{1}{r-1}\sum_{j=1}^{r} s_j s_j^T \tag{8.5}$$

$$= \frac{1}{r-1}\left[(\sum_{i=1}^{n^2} z_i\mathcal{D}_i)_1 + \cdots + (\sum_{i=1}^{n^2} z_i\mathcal{D}_i)_r\right] = \sum_{i=1}^{n^2} \mathcal{D}_i w_i \tag{8.6}$$

where the notation $(.)_i$ stands for the outer product of the ith data point we showed in (8.4) and w's are the sums of the z_i's inside each outer product. In general, w which is the sum of rank-one outer products, need not be sparse, but we can impose constraints on the dictionary learning over $\mathcal{B}$ such that w is sparse, which paves the way for the *Generalized Dictionary Learning* framework.

8.3 Generalized Dictionary Learning

We assume that we have input matrices $S_i \in \mathcal{S}_{++}^p$, $1 \le i \le m$. From the intuitions we gained from (8.6), we can now rewrite the sparse coding formulation in (5.3) (LASSO) as:

$$S_i \approx \mathcal{D}C_i, \quad \text{and} \quad C_i \text{ is sparse}, \quad \text{for } 1 \le i \le m. \tag{8.7}$$

Based on (8.7), we propose to solve *Generalized Dictionary Learning* (GDL) by casting it as the penalized optimization problem

$$\min_{C_1,\ldots,C_m,D} \quad \frac{1}{2}\sum_{i=1}^{m} \|S - \mathcal{D}C_i\|_{\mathrm{F}}^2 + \sum_{i=1}^{m} \beta_i \mathrm{sp}(C_i), \tag{8.8}$$

where $\beta_i > 0$ are scalar, 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 $\|C\|_{\mathrm{tr}}$.

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 35. *If $A \succeq 0$, and $B \in \mathbb{R}^{p\times n}$ is any matrix, then $BAB^T \succeq 0$.*

Theorem (35) suggests that we should encode $\mathcal{D}$ by the following bilinear map

$$\mathcal{D}C := \mathcal{B}C\mathcal{B}^T, \quad \text{for the dictionary } \mathcal{B} \text{ as suggested by (8.6)} \tag{8.9}$$

and additionally *restrict* to $C \succeq 0$. Notice that, viewed as a linear map (8.9) can be written as (vec is an operator that just stacks columns of its argument):

$$\mathrm{vec}(\mathcal{B}C\mathcal{B}^T) = (\mathcal{B}\otimes\mathcal{B})\,\mathrm{vec}(C), \tag{8.10}$$

where the operator $\mathcal{D}$ is encoded (isomorphically) by the product $\mathcal{B}\otimes\mathcal{B}$. The matrix notation in (8.9) looks simpler. In addition, analyzing the storage and computational requirements for the two models; it can be easily shown that (8.10) takes $mp^2 + p^2q^2 + mq$ storage space for m covariance matrices each of dimension p, while (8.9) takes $mp^2 + pq + mq$. Computationally, the second formulation takes $O(p^2q)$ iterations for solving the LASSO, while it is just $O(pq)$ using the first formulation. Motivated by this analysis, we chose to use (8.9).

There are two fundamental choices for modeling the matrix C:

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

In this chapter we focus on the first choice, which is equivalent to modeling input SPD matrices as weighted sums of rank-1 matrices, because then

$$S \approx \mathcal{B}C\mathcal{B}^T = \sum_{i=1}^{n} c_i b_i b_i^T, \quad \text{where } c_i = C_{ii}. \tag{8.11}$$

Although diagonal C might appear to be a simple choice, it is quite powerful as equation (8.11) suggests; more importantly, this choice is crucial for developing efficient GDL algorithms.

8.3.1 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 $\text{sp}(C) = \|C\|_1$ for enforcing sparsity. Then, using representation (8.11), the GDL formulation (8.8) becomes

$$\min_{C_1,\ldots,C_N \geq 0, \mathcal{B}} \quad \frac{1}{2} \sum\nolimits_{i=1}^{m} \|S_i - \mathcal{B} C_i \mathcal{B}^T\|_{\text{F}}^2 + \sum\nolimits_{i=1}^{m} \beta_i \|C_i\|_1, \tag{8.12}$$

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 (8.12) is also nonconvex, which makes it extremely unlikely to obtain a globally optimal solution. Here, we will follow the standard approach of alternating minimization towards a local minimum. Since it is often observed that the number of input data points m is large, the alternating step over the Cs can easily become computationally prohibitive. Taking cue from the recent work in dictionary learning [Elad and Aharon, 2006; Mairal et al., 2009a], 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 $\mathcal{B}$. A practical choice is to require $\|b_j\|_2 \leq 1$ for each column of matrix $\mathcal{B}$. We denote these requirements by the feasible set $\mathcal{B}$. To run a stochastic-gradient procedure, we break up the processing into K "mini-batches." Then, we rewrite the GDL (8.12) over these mini-batches as

$$\min_{\mathcal{B} \in \mathcal{B}} \quad \Phi(\mathcal{B}) := \sum\nolimits_{b=1}^{K} \phi_b(\mathcal{B}), \tag{8.13}$$

where ϕ_b denotes the objective function for batch b. Let k_b be the size of batch b ($1 \leq b \leq K$) 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

$$\phi_b(\mathcal{B}) := \min_{C_{j(1)},\ldots,C_{j(k)} \geq 0} \quad \frac{1}{2} \sum\nolimits_{i=1}^{k_b} \|S_{j(i)} - \mathcal{B} C_{j(i)} \mathcal{B}^T\|_{\text{F}}^2 + \beta_{j(i)} \|C_{j(i)}\|_1. \tag{8.14}$$

Our stochastic-gradient algorithm then performs the iteration

$$\mathcal{B}_{t+1} = \Pi_{\mathcal{B}}(\mathcal{B}_t - \eta_t \nabla_{\mathcal{B}} \phi_{b(t)}(\mathcal{B}_t)), \quad b(t) \in [1..K],\ t = 0, 1, \ldots, \tag{8.15}$$

where $\Pi_{\mathcal{B}}$ denotes orthogonal projection onto $\mathcal{B}$. Assuming that ϕ_b is determined by a unique solution to (8.14), it can be shown that the gradient $\nabla_{\mathcal{B}}\phi_{b(t)}$ is well defined. Specifically, let $(C^*_{j(1)}, \ldots, C^*_{j(k)})$ be the argmin of (8.14). Then, some matrix differential calculus shows that (let $b \equiv b(t)$)

$$\nabla_{\mathcal{B}}\phi_b(\mathcal{B}) = 2\sum\nolimits_{i=1}^{k_b}\Big(\mathcal{B}C^*_{j(i)}\mathcal{B}^T - S_{j(i)}\Big)\mathcal{B}C^*_{j(i)}. \tag{8.16}$$

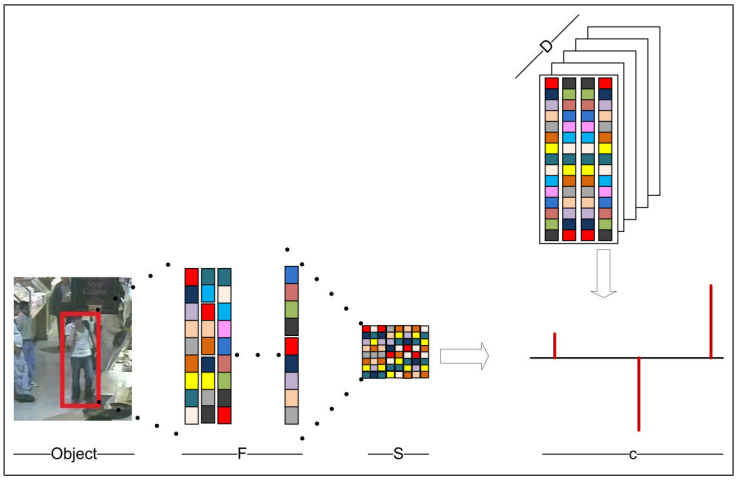


Figure 8.1: Sparse coding for covariances: From the object one extracts base-features $F_1, \ldots, F_k$. These, then yield the covariance feature $S = \sum_i (F_i - \mu)(F_i - \mu)^T$ where μ is the mean feature vector, which has a sparse coding C in the dictionary $\mathcal{B}$, i.e., $S \approx \mathcal{D}C$.

Sparse Coding: Computing ϕ_b

All that remains to specify is how to compute ϕ_b, i.e., how to solve (8.14). First, notice that (8.14) is just a sum of k_b independent problems, so without loss of generality we need to consider only a subproblem of the form

$$\min_{C\geq 0} \quad f(C) := \frac{1}{2}\|S - \mathcal{B}C\mathcal{B}^T\|_{\mathrm{F}}^2 + \beta\|C\|_1, \tag{8.17}$$

where $\beta > 0$, and C is restricted to be a diagonal matrix. Since $C \geq 0$, problem (8.17) further simplifies to

$$\min_{C\geq 0} \quad f(C) := \frac{1}{2}\|S - \mathcal{B}C\mathcal{B}^T\|_{\mathrm{F}}^2 + \beta\,\mathrm{Tr}(C), \tag{8.18}$$

which is nothing but a regularized Non-Negative Least-Squares (NNLS) problem. There exist a variety of solvers for NNLS, for example, LBFGS-B [Liu and Nocedal, 1989], or Spectral

Algorithm 6 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 (8.18).
    end for
    Update dictionary, $\mathcal{B}_{t+1} = \Pi_{\mathcal{B}}(\mathcal{B}_t - \eta_t \nabla_{\mathcal{B}} \phi_{b(t)}(\mathcal{B}_t))$
    $t \leftarrow t + 1$.
  end while
  return $(\mathcal{B}, C_1, C_2, \ldots,)$.

Projected-Gradient (SPG) [Birgin et al., 2000]. We prefer to use the latter, as it is not only simple, but also exhibits excellent empirical performance. Ignoring other details, at its core SPG performs the following iteration

$$C_{k+1} = [C_k - \alpha_{\text{BB}} \nabla f(C_t)]^+, \tag{8.19}$$

where $[\cdot]^+ = \max(0, \cdot)$ denotes orthogonal projection onto the nonnegative orthant, and α_{BB} denotes the famous *Barzilai-Borwein* stepsize [Barzilai and Borwein, 1988]:

$$\alpha_{\text{BB}} := \frac{\|C_k - C_{k-1}\|_2^2}{\langle C_k - C_{k-1}, \nabla f(C_k) - \nabla f(C_{k-1}) \rangle}. \tag{8.20}$$

The final crucial detail for implementing (8.19) is that C is diagonal, so iteration (8.19) should update only the diagonal part of C; the rest of the matrix can be assumed to be zero. Also, the gradient $\nabla f(C)$ is given by the diagonal matrix

$$\nabla f(C) = \text{Diag}\big(\mathcal{B}^T(\mathcal{B}C\mathcal{B}^T - S)\mathcal{B}\big) \tag{8.21}$$

Algorithm 6 assembles all the above details into pseudocode for online GDL. Figure 8.1 provides a high level schema of the GDL algorithm.

8.4 Nearest Neighbors via GDL

Motivated by the SCT representation, we propose a similar indexing representation for covariance matrices with respect to their sparsity against the dictionary.

Definition 14. Subspace Combination Tuple (SCT0)
Let $S \in \mathcal{S}_{++}^{d}$ be an input SPD matrix, $\mathcal{B} \in \mathbb{R}^{d \times n}$ an overcomplete dictionary, and $u_i,\ i \in \{1, \cdots, n\}$ a unique identifier for the ith column of $\mathcal{B}$. If C is a diagonal matrix, such that $C = diag(c_1, c_2, \cdots, c_n)$ is the coefficient matrix corresponding to the sparse representation of S as per (8.18)*, then a tuple $h(S) = \{i1, i2, \cdots, ik\}$ is defined as a* Subspace Combination Tuple *(SCT), if $c_{ij} \neq 0, \forall ij \in \{1, 2, \cdots, n\}$ and $\{i1, i2, \cdots, ik\}$ is a strictly increasing sequence.*

Note that, this definition is in accordance with the SCT0 definition that we introduced in Section 5.3. This representation of input SPD matrices as tuples opens up the possibility of using hash tables for fast locality sensitive hashing as we saw in the earlier chapters. To tackle collisions in the hash buckets, the colliding input matrices are organized as 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 descriptor (query SPD matrix), we solve (8.18) to first obtain its sparse coefficient matrix. From this matrix, 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 (C.1) to find the best matches (the bucket can also be organized for faster than linear scans, if desired).

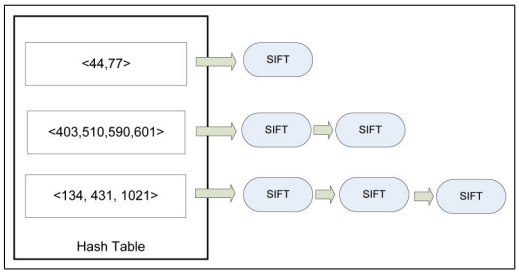


Figure 8.2: An illustration of the hashing idea. The input covariance matrix is sparse coded and the non-zero active coefficients are formed into a tuple, which is then used for indexing into a hash table. To resolve hash table collisions, the colliding covariances are arranged in a suitable data structure. The covariance matrices are denoted as S_i (for random values of i) in the figure.

8.5 Experiments and Preliminary Results

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) 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 [Phillips et al., 1998, 2000]. 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 [Pang et al., 2008], 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 [Dana et al., 1999] 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

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

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 8.3.

Figure 8.3: Sample image from LabelMe object dataset (top), FERET face appearances (mid) and Brodatz texture database (bot).

8.5.1 Algorithms Compared

We briefly review below the techniques we compared our method against. We consider the following methods for comparison: (i) Log-Euclidean Embedding followed by vectorization and application of state-of-the-art LSH techniques (we use Hamming (HAM) and L2LSH hash functions), (ii) mere vectorization of the upper (or lower) triangular part of the covariance matrix, and (iii) Kernelized LSH (KLSH) using the geodesic pseudo-kernel.

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

Vectorization: We chose this experiment to demonstrate that a mere vectorization of the covariance matrix (without projecting it using the log-Euclidean mapping), does not lead to a good hashing (VEC).

Kernelized LSH: A recently proposed, sophisticated technique built on LSH, is Kernelized LSH (KLSH) [Kulis and Grauman, 2009]. 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 [Forstner and Moonen, 1999]. This kernel K has the following form: For two SPD matrices X and Y,

$$K(X, Y) := e^{-\gamma \|\log(\lambda(X,Y))\|_2^2}, \tag{8.22}$$

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

Metric Tree: We also compare GDL against another popular non-hashing based NN: Metric tree (MT). The covariance dataset is first arranged into a binary metric tree data structure [Ciaccia et al., 1997] based on the geodesic distance. Later, query descriptor traverses the tree based on its metric distance to the children of a given node.

8.5.2 Dictionary Learning

Through a cross-validation approach, we decided on the dictionary sizes for the various datasets as follows: 7×28 for the object dataset, 40×160 for the faces dataset, and 5×50 for the texture dataset.

8.5.3 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 8.1 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 is low, which is exactly as desired.

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

Table 8.1: Percentage of the database searched by GDL to find the nearest neighbor.

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 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 $\epsilon = 0.75$. Figure 8.4 shows the accuracy of NN for each of the datasets, where accuracy is defined as:

$$\text{Accuracy} := \frac{\#\text{correct matches}}{\#\text{query size}}. \tag{8.23}$$

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. 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.

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

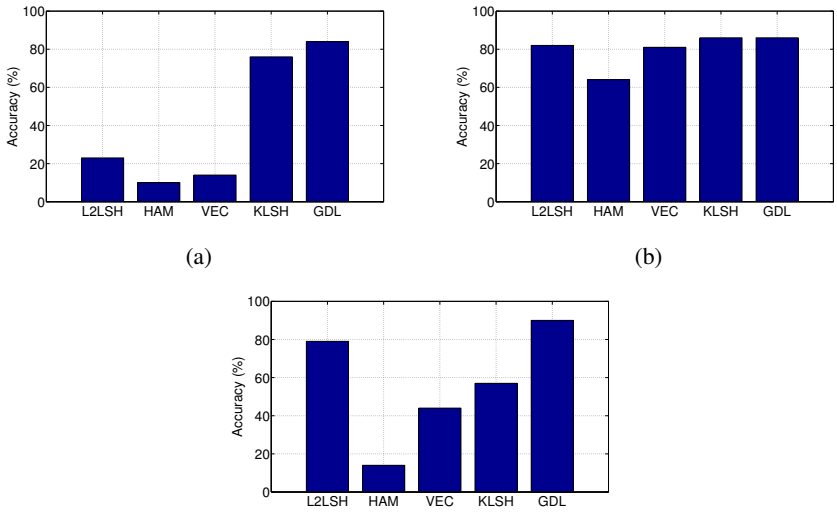


Figure 8.4: Plots of the $accuracy$ of NN retrieval of GDL compared to various techniques; (a) objects dataset, (b) faces dataset, and (c) texture dataset.

Chapter 9

Conclusions and Future Work

9.1 Summary

In this thesis, we addressed the problem of similarity search in visual datasets. We saw that similarity search is an important algorithmic component of several fundamental algorithms in computer vision and machine learning. We looked at this search problem for data represented in two different mathematical forms: (i) data as covariance matrices, and (ii) data as high-dimensional vectors. A natural question that might arise with respect to these data descriptors that we have not addressed yet is; *Which data type is useful and when it is useful?*. Although this question is subjective and entirely depends on the application at hand, we would like to point out some properties of the data that might help in easing this decision.

Covariance valued data is useful in situations where the discriminating or recognizing properties of the data are based on feature correlations rather than on the frequency of occurrences of certain patterns (as is the case with feature histograms). A typical application which might help understand this concept is that of action recognition. Actions such as walking, running, etc. produce a unique profile of optical flow vectors [Guo et al., 2010]. Although the flows are in a variety of directions, there is a certain pattern to these flow vectors for a unique action. Covariance from optical flow features (such as the correlations between the flow directions, divergence, vorticity, etc. of the flow vectors) are shown to recognize actions more robustly compared to histogram based approaches. On the other hand, suppose that all the flow vectors are uni-directional (for example, in an action such as pushing a car). In this case, there might not be any variance in the flow vectors, as a result a covariance might be rank-deficient. For this

action, a histogram based descriptor on optical flow (such as the HoG-HoF descriptor [Laptev et al., 2008]) might provide better results.

The contributions of this thesis benefit applications that have decided on one of the two data types and demands efficient similarity search on that data type for its performance. Such applications are plenty in computer vision, pattern recognition, robotics or data mining. We demonstrated both theoretically and empirically the challenges to face when computing similarities in both the above data spaces. Our focus was on developing algorithms for improving the speed of retrieval, at the same time performing at par or better than contemporary algorithms. In the following, we list explicitly the contributions that we made in this thesis.

For covariance valued data,

- We proposed a novel similarity measure; the Jensen-Bregman LogDet Divergence for efficient similarity computations, specifically when the application uses covariances that are high dimensional. We showed that JBLD computations are superior to existing metrics when the data dimensions are large or when there is a need to compute gradients. We also showed the connections that JBLD enjoys with other metrics, also demonstrating other desirable theoretical properties.

- We addressed the issue of applying JBLD to large datasets, we developed a metric tree on JBLD, a prime component of this framework is a K-means clustering algorithm on our measure. We provided concrete analysis of the convergence of our clustering algorithm. Later, we demonstrated the empirical performance of JBLD on a variety of computer vision datasets.

- We enabled offline computations on JBLD for fast comparisons, we provided an approximate measure (AJBLD), which is a lower bound to JBLD. This measure takes less storage, and is potentially faster than JBLD or other metrics on covariances, although we need to sacrifice on accuracy of retrieval.

- A key component in several similarity search applications is data clustering. In this thesis, we developed a Dirichlet Process Mixture Model (DPMM) framework for unsupervised clustering on covariances. This is an important component for covariance based algorithms that work on real-world data (such as video surveillance) for which the complexity of the data increases quickly. We derived DPMM models for three potential probability

measures: (i) LogDet divergence, (ii) log-Euclidean distance, and (iii) Frobenius distance. Our empirical results demonstrated that DPMM based on the LogDet measure show the best performance for unsupervised clustering on covariance valued data.

The second part of this thesis looked at the problem of similarity search on high dimensional vector valued data. Although similarity search problem on this data type have been investigated thoroughly over the past few decades, we approached this problem from a unique perspective; using dictionary learning and sparse coding. Traditionally, sparse coding using overcomplete dictionaries is well-known to be sensitive to data noise and uncertainties. We proposed several algorithms to circumvent these issues, providing state-of-the-art results. Below, let us summarize our contributions.

- We first showed theoretical connections between sparse coding and the principles of locality sensitive hashing. Later, we derived a simple hashing scheme using the active support of sparse codes generated by a learned dictionary. This helped us understand the difficulties in adopting sparse coding for the NN retrieval problem. Our approaches to overcome these difficulties are two fold: (i) Robust Dictionary Learning (RDL), and (ii) Robust Sparse Coding (RSC).

    1. RDL looks at learning the dictionary basis so that they are aligned in directions which may nullify the data perturbations: To this end, we proposed two further algorithms:

        - A statistical denoising model, in which the data descriptors were treated as discrete signals corrupted by noise of a particular type. We analyzed the noise model for matching SIFT descriptors and developed a Gaussian-Laplacian denoising model. Later, we derived an online dictionary learning algorithm for solving this model using stochastic gradient descent.
        - The denoising model assumes that we know the data noise model, which is difficult in practice. To circumvent this fundamental issue, we took refuge in robust optimization principles and proposed a dictionary learning algorithm that aligns the basis vectors in directions of worst case data perturbations. This model was seen to produce better results than the statistical model.

    2. RSC utilizes the idea of sparse coding in steps (as done in algorithms such as LARS)

so that sparsity at a higher regularization produces the most robust codes, while sacrificing the code length (which is important for diversity). We proposed an algorithm: Multi-Regularization Sparse Coding (MRSC) that uses this idea and showed that it can achieve superior retrieval performance and scalability.

Combining the first two parts of thesis into a comprehensive framework is the content of the third part of this thesis. Towards this end, we developed a framework: Generalized Dictionary Learning (GDL), for sparse coding covariances using dictionary learning. In GDL, we approximated each covariance matrix as a sparse linear combination of rank-one dictionary atoms, the active dictionary indices are then used for hashing. We formulated a novel objective for this problem, that is computationally and storage efficient, at the same time empirically producing stable sparse coding of covariances. We provided experimental results on several covariance datasets that showed that our algorithm performs consistently and more accurately than the state-of-the-art.

9.2 Future Work

In this section, we will elucidate several directions along which the works in this thesis can be extended. Let us consider each data type and the algorithms we derived for them separately.

9.2.1 Covariance Valued Data

Although, we discussed several of the properties of covariances and their usefulness in computer vision applications, there are a few issues that these data types bring along, which might need to be addressed. We will review some of these problems and pose some directions of investigations.

JBLD for DTMRI

Symmetric positive definite matrices are fundamental data objects in DTMRI. The JBLD measure that we developed in this thesis was predominantly derived for mainstream computer vision applications. DTMRI applications require several specific properties that a measure on covariances need to satisfy. For example, DTMRI images are often high resolution images (each pixel of which is a 3×3 symmetric positive definite matrix). This necessitates applications using

these images to use sub-sampling and reconstruction algorithms for their efficient storage and retrieval. Such reconstruction algorithms need interpolation schemes on the relevant metrics. Extending the JBLD work to address such problems as interpolation, regularization, etc. are important topics that need to be looked into in future.

Semi-Supervised DPMM for Covariances

Recalling our DPMM framework on covariances for visual surveillance, one important issue (other than the computational expense for comparing these matrices) is that, covariances may be sensitive to outliers and feature perturbations. For example, it is well-known that a single large outlier can skew the covariance matrix. As a result, these matrices may lead to low accuracies for retrieval. This is important from a surveillance application perspective when an object is tracked between cameras. Specular reflections from object surfaces can skew the appearances, resulting in erroneous tracking. One way to circumvent this difficulty is to provide supervisor feedback to the unsupervised clustering offered by DPMM. For example, the supervisor can provide inputs constraining two dissimilar appearances to be clustered together. Such supervision cannot be taken care of in the current DPMM methodology, demanding further investigations.

Metric Learning on Covariances

Covariance descriptors use a heuristic combination of feature vectors collected from an image region and assumes that this heuristic combination works for the concerned application. It might seem more appealing if we have a mechanism to learn the effective weights or contributions of each of these features to the overall application performance. This will give us an intuition into why a particular combination of features works well and what should not be used. Such an idea sounds like the traditional metric learning idea for vector valued data [Davis et al., 2007] in which one learns a Mahalanobis type of metric (from a training set) that weighs the contribution of each dimension in a feature vector for improved performance. Extending such a framework to covariance data type might be worthwhile. Unfortunately, such an approach is difficult when compared to the Mahalanobis type mechanism, as metric learning on covariances need to consider the Riemannian manifold structure of these matrices as well.

LSH on Covariances

An important NN technique that we did not completely address in this thesis is that of establishing a locality sensitive hashing framework on covariances. Although there are extensions of the standard LSH schemes to covariances (such as first approximating these matrices using log-Euclidean projections (we saw this in Chapter 2) and later use the standard LSH framework), an LSH framework that produces similar hash codes for covariances closer in the original Riemannian metric has not been addressed so far. This is an important direction that needs to be considered.

Other Metrics on Covariances

Finally, symmetric positive definite matrices are important objects not only in computer vision and machine learning, but they have an important role to play in quantum information theory as *density matrices* [Fano, 1957]. These density matrices are useful in describing the state of a quantum system. A natural divergence in this setting is the *von* Neumann entropy defined as $S = -\operatorname{Tr}(X \log X)$ for $X \in \mathcal{S}_{++}$. Note that this is a convex function, just like the $-\log|\,|$ function we used in JBLD. It would be an interesting exercise to check if using von Neumann entropy in a symmetric Bregman divergence lead to any useful measure.

9.2.2 Vector Valued Data

Next, let us consider a few extensions to the sparse coding algorithms for better NN retrieval on vector valued data.

Efficient Dictionary Learning and Sparse Coding

One of the most important issues with of our NN scheme for vector valued data is the need to sparse code and learn the dictionary. When the data dimensionality is very large (more than 2000D), dictionary learning is seen to become impractical. Similarly, when using large over-complete dictionaries, sparse coding a given data vector is often seen to suffer (especially when we seek a large support set). These problems may constrain the usefulness of our approaches.

A potential direction of investigation is that of *bulk sparse coding* in which data descriptors are first clustered according to their similarities, and later sparse coded together, with the

assumption that these descriptors might have an overlapping regularization path. Such a mechanism could speed up the sparse coding stage. A similar strategy can be used for large scale dictionary learning as well. There have been a few initial investigations in these directions as can be seen in papers such as [Xiang et al., 2011] and [Jain et al., 2011].

Semi-Supervised Dictionary Learning

Although we investigated several algorithms for making the NN via sparse coding robust to data uncertainties, the DL strategy remained completely unsupervised. An alternative direction to achieve robustness is to make the DL step learn the dictionaries according to user specified constraints on the data points. For example, a subset of the ground truth labels can be used to learn uncertainty models for the data points in a semi-supervised or active learning way. This is a direction that needs to be looked at in the future.

References

M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *Transactions on signal processing*, 54(11):4311, 2006.

D. Alexander, C. Pierpaoli, P. Basser, and J. Gee. Spatial transformations of diffusion tensor magnetic resonance images. *Transactions on Medical Imaging*, 20(11):1131–1139, 2002.

T.W. Anderson. *An Introduction to Multivariate Statistical Analysis*. Wiley New York, 1958.

A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Symposium on Foundations of Computer Science*, pages 459–468, 2006.

V. Arsigny, P. Fillard, X. Pennec, and N. Ayache. Log-Euclidean metrics for fast and simple calculus on diffusion tensors. *Magnetic Resonance in Medicine*, 56(2):411–421, 2006.

V. Arsigny, P. Fillard, X. Pennec, and N. Ayache. Geometric means in a novel vector space structure on symmetric positive-definite matrices. *Matrix Analysis and Applications*, 29(1): 328, 2008.

S. Arya, D.M. Mount, N.S. Netanyahu, R. Silverman, and A.Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM*, 45(6): 891–923, 1998.

A. Banerjee, D. Boley, and S. Acharyya. Symmetrized Bregman divergences and metrics. *The Learning Workshop*, 2009.

A. Banerjee, S. Merugu, I. Dhillon, and J. Ghosh. Clustering with Bregman divergences. *Journal of Machine Learning Research*, 6:1705–1749, 2005.

L. Bar and G. Sapiro. Hierarchical dictionary learning for invariant classification. In *International Conference on Acoustics Speech and Signal Processing*, 2010.

J. Barzilai and J.M. Borwein. Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, 8(1):141, 1988.

H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

M. Beal, Z. Ghahramani, and C. Rasmussen. The infinite hidden Markov model. *Advanced in Neural Information Processing Systems*, pages 577–584, 2002.

Jeffrey S. Beis and David G. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Computer Vision and Pattern Recognition*, pages 1000–1006, 1997.

A. Ben-Tal, L. El Ghaoui, and A.S. Nemirovskiĭ. *Robust optimization*. Princeton University Press, 2009.

A. Ben-Tal and A. Nemirovski. Robust solutions of uncertain linear programs. *Operations Research Letters*, 25(1):1–14, 1999.

D. Bertsekas, W. Hager, and O. Mangasarian. *Nonlinear programming*. Athena Scientific, Belmont, Massachusetts, 1999.

K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is nearest neighbor meaningful? *Database Theory*, pages 217–235, 1999.

R. Bhatia. *Matrix analysis*, volume 169. Springer Verlag, 1997.

R. Bhatia. *Positive definite matrices*. Princeton Univ Press, 2007.

D. Bini and B. Iannazzo. Computing the Karcher mean of symmetric positive definite matrices. *Linear Algebra and its Applications*, 2011.

E.G. Birgin, J.M. Martínez, and M. Raydan. Non-monotone spectral projected gradient methods on convex sets. *SIAM Journal on Optimization*, 10(4):1196–1211, 2000.

D. Blackwell and J. MacQueen. Ferguson distributions via Polya urn schemes. *Annals of Statistics*, 1(2):353–355, 1973.

C. Bohm, S. Berchtold, and D.A. Keim. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *ACM Computing Surveys*, 33(3): 322–373, 2001.

L. Bottou. Online learning and stochastic approximations. *Online learning in neural networks*, pages 9–42, 1998.

N. Bouguila and D. Ziou. A Dirichlet process mixture of generalized Dirichlet distributions for proportional data modeling. *Transactions on Neural Networks*, 21(1):107 –122, 2010.

G. Box and G. Tiao. *Bayesian inference in statistical analysis*. Wiley New York, 1992.

S.P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

E. Boyer, A. Bronstein, M. Bronstein, B. Bustos, T. Darom, R. Horaud, I. Hotz, Y. Keller, J. Keustermans, A. Kovnatsky, et al. SHREC 2011: Robust feature detection and description benchmark. *Arxiv preprint arXiv:1102.4258*, 2011.

L. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7(3):200–217, 1967.

S. Brin. Near neighbor search in large metric spaces. In *International Conference on Very Large Data Bases*, 1995.

T. Brox, M. Rousson, R. Deriche, and J. Weickert. Unsupervised segmentation incorporating colour, texture, and motion. In *Computer Analysis of Images and Patterns*, pages 353–360. Springer, 2003.

J.F. Cai, E.J. Candes, and Z. Shen. A singular value thresholding algorithm for matrix completion. *Arxiv preprint arXiv:0810.3286*, 2008.

Y. Cai, V. Takala, and M. Pietikäinen. Matching groups of people by covariance descriptor. In *International Conference on Pattern Recognition*, pages 2744–2747, 2010.

E. J. Candes and D. Donoho. Curvelets: A surprisingly effective nonadaptive representation for objects with edges. Technical report, DTIC Document, 2000.

E.J. Candes. Compressive sampling. In *International Congress of Mathematics*, 2006.

E.J. Candes and Y. Plan. Matrix completion with noise. *Proceedings of the IEEE*, 98(6):925–936, 2010.

C. Caraminis, H. Xu, and S. Mannor. *Optmization for Machine Learning*, chapter Robust Optimization in Machine Learning. MIT Press, 2011.

R. Caseiro, J.F. Henriques, and J. Batista. Foreground segmentation via background modeling on Riemannian manifolds. In *International Conference on Pattern Recognition*, pages 3570–3574, 2010.

L. Cayton. Fast nearest neighbor retrieval for Bregman divergences. In *International Conference on Machine Learning*, pages 112–119, 2008.

Y. Censor and S. Zenios. *Parallel Optimization: Theory, Algorithms, and Applications*. Oxford University Press, 1997.

R. Chaudhry and Y. Ivanov. Fast approximate nearest neighbor methods for non-Euclidean manifolds with applications to human activity analysis in videos. *European Conference on Computer Vision*, pages 735–748, 2010.

G. Chechik, V. Sharma, U. Shalit, and S. Bengio. An online algorithm for large scale image similarity learning. In *Advances in Neural Processing Information Systems*, 2009.

C. Chen, M. Ryoo, and J. Aggarwal. UT-Tower Dataset: Aerial View Activity Classification Challenge. http://cvrc.ece.utexas.edu/SDHA2010/Aerial_View_Activity.html, 2010.

P. Chen. *Bregman metrics and their applications*. PhD thesis, University of Florida, 2007.

S.S. Chen, D.L. Donoho, and M.A. Saunders. Atomic decomposition by basis pursuit. *SIAM Review*, pages 129–159, 2001.

A. Cherian, J. Andersh, V. Morellas, B. Mettler, and N. Papanikolopoulos. Motion estimation of a miniature helicopter using a single onboard camera. *American Control Conference*, 2010.

A. Cherian, S. Sra, A. Banerjee, and N. Papanikolopoulos. Efficient similarity search for covariance matrices via the Jensen-Bregman logdet divergence. In *International Conference on Computer Vision*, pages 2399–2406, 2011.

M.C. Chiang, R.A. Dutton, K.M. Hayashi, O.L. Lopez, H.J. Aizenstein, A.W. Toga, J.T. Becker, and P.M. Thompson. 3D pattern of brain atrophy in HIV/AIDS visualized using tensor-based morphometry. *Neuroimage*, 34(1):44–60, 2007.

P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *International Conference on Very Large Data Bases*, pages 426–435, 1997.

K.L. Clarkson. An algorithm for approximate closest-point queries. In *Symposium on Computational Geometry*, pages 160–164, 1994.

T. Cormen. *Introduction to algorithms*. The MIT press, 2001.

N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. *European Conference on Computer Vision*, pages 428–441, 2006.

K.J. Dana, B. Van Ginneken, S.K. Nayar, and J.J. Koenderink. Reflectance and texture of real-world surfaces. *ACM Transactions on Graphics*, 18(1):1–34, 1999.

M. Datar, N. Immorlica, P. Indyk, and V.S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. *Symposium on Computational Geometry*, pages 253–262, 2004.

R. Datta, D. Joshi, J. Li, and J.Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys*, 40(2):1–60, 2008.

I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 57 (11):1413–1457, 2004.

J. Davis and I. Dhillon. Differential entropic clustering of multivariate Gaussians. *Advances in Neural Information Processing Systems*, 2007.

J. Davis, B. Kulis, P. Jain, S. Sra, and I.S. Dhillon. Information-theoretic metric learning. In *International Conference on Machine Learning*, 2007.

I. Dhillon and J. Tropp. Matrix nearness problems with Bregman divergences. *SIAM Journal of Matrix Analysis and Applications*, 29(4):1120–1146, 2007.

D. Donoho. Compressed sensing. Technical report, Department of Statistics, Stanford University, 2004.

D.L. Donoho. Compressed sensing. *Transaction on Information Theory*, 52(4):1289–1306, 2006.

I. Dryden, A. Koloydenko, and D. Zhou. Non-Euclidean statistics for covariance matrices, with applications to diffusion tensor imaging. *University of Nottingham, NG7 2RD, UK*, 2008.

M. Eaton. *Multivariate statistics: a vector space approach*. Wiley New York, 1983.

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32(2):407–451, 2004.

M. Elad. *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*. Springer Verlag, 2010.

M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *Transactions on Image Processing*, 15(12):3736–3745, 2006.

K. Engan, S.O. Aase, and J.H. Husøy. Multi-frame compression: Theory and design. *Signal Processing*, 80(10):2121–2140, 2000.

F. Porikli, and O. Tuzel. Covariance tracker. *Computer Vision and Pattern Recognition*, 2006.

U. Fano. Description of states in quantum mechanics by density matrix and operator techniques. *Reviews of Modern Physics*, 29(1):74–93, 1957.

T. Ferguson. A Bayesian analysis of some nonparametric problems. *Annals of Statistics*, 1(2): 209–230, 1973.

P. Fillard, V. Arsigny, N. Ayache, and X. Pennec. A Riemannian framework for the processing of tensor-valued images. *Deep Structure, Singularities, and Computer Vision*, pages 112–123, 2005.

D. Fink. A compendium of conjugate priors. *Tech. Report.*, 1997.

W. Forstner and B. Moonen. A metric for covariance matrices. *Qua vadis geodesia*, pages 113–128, 1999.

A. Frome, Y. Singer, F. Sha, and J. Malik. Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *International Conference on Computer Vision*, pages 1–8, 2007.

Y. Ge and J. Yu. A scene recognition algorithm based on covariance descriptor. In *Conference on Cybernetics and Intelligence Systems*, pages 838–842, 2008.

D. Gil, J. Garcia-Barnes, A. Hernandez-Sabate, and E. Marti. Manifold parametrization of the left ventricle for a statistical modeling of its complete anatomy. In *Society of Photo-Optical Instrumentation Engineers Conference Series*, volume 7623, page 3, 2010.

P.R. Gill, A. Wang, and A. Molnar. The in-crowd algorithm for fast basis pursuit denoising. *Transactions on Signal Processing*, 59(10):4595–4605, 2011.

G. Golub and C. Van Loan. *Matrix Computations*. Johns Hopkins University Press, third edition, 1996.

L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. *Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2247–2253, 2007.

C. Grinstead and J. Snell. *Introduction to probability*. Amer Mathematical Society, 1997.

M. Gromov. Monotonicity of the volume of intersection of balls. *Geometrical Aspects of Functional Analysis*, pages 1–4, 1987.

Q. Gu and J. Zhou. A similarity measure under Log-Euclidean metric for stereo matching. In *International Conference on Pattern Recognition*, pages 1–4, 2009.

K. Guo, P. Ishwar, and J. Konrad. Action recognition in video by covariance matching of silhouette tunnels. In *Brazilian Symposium in Computer Graphics and Image Processing*, pages 299–306, 2010.

Antonin Guttman. R-trees: a dynamic index structure for spatial searching. In *International Conference on Management of Data*, pages 47–57, 1984.

R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

S. Hidot and C. Saint-Jean. An Expectation-Maximization algorithm for the Wishart mixture model: Application to movement clustering. *Pattern Recognition Letters*, 31(14):2318–2324, 2010.

R. Horn and C. Johnson. *Matrix analysis*. Cambridge University Press, 1990.

P. Indyk. On approximate nearest neighbors in non-Euclidean spaces. *Symposium on Foundations of Computer Science*, pages 148–155, 1998.

P. Indyk. Stable distributions, pseudorandom generators, embeddings and data stream computation. In *Symposium on Foundations of Computer Science*, pages 189–197, 2000.

P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. *Symposium on Theory of computing*, pages 604–613, 1998.

P. Jaccard. Etude comparative de la distribution florale dans une portion des Alpes et du Jura. *Bulletin de la Socit vaudoise des Sciences Naturelles*, 37:547–579, 1901.

P. Jain, B. Kulis, I.S. Dhillon, and K. Grauman. Online metric learning and fast similarity search. *Advances in Neural Information Processing Systems*, 2008.

P. Jain, A. Tewari, and I.S. Dhillon. Orthogonal matching pursuit with replacement. *arXiv preprint arXiv:1106.2774*, 2011.

V. Jain and E. Learned-Miller. FDDB: a benchmark for face detection in unconstrained settings. Technical Report UM-CS-2010-009, University of Massachusetts, Amherst, 2010.

H. Jegou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. *European Conference on Computer Vision*, pages 304–317, 2008.

H Jegou, M Douze, and C Schmid. Product quantization for nearest neighbor search. *Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011.

A.E. Johnson. *Spin-images: A representation for 3D surface matching*. PhD thesis, Carnegie Mellon University, 1997.

R. Johnson and D. Wichern. *Applied multivariate statistical analysis*. Prentice Hall, 1998.

S. Kak. Quantum information and entropy. *International Journal of Theoretical Physics*, 46(4): 860–876, 2007.

K.Dana, B. Van-Ginneken, S. Nayar, and J. Koenderink. Reflectance and texture of real world surfaces. *ACM Transactions on Graphics*, 18(1):1–34, 1999.

S. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky. An interior-point method for large-scale l1-regularized least squares. *Selected Topics in Signal Processing*, 1(4):606–617, 2007.

J.M. Kleinberg. Two algorithms for nearest-neighbor search in high dimensions. *Symposium on Theory of computing*, pages 599–608, 1997.

D.E. Knuth. *The art of computer programming. Vol. 3, Sorting and Searching*. Addison-Wesley Reading, MA, 1973.

B. Kulis and K. Grauman. Kernelized locality-sensitive hashing for scalable image search. In *International Conference on Computer Vision*, 2009.

B. Kulis, M. Sustik, and I. Dhillon. Low-rank Kernel Learning with Bregman Matrix Divergences. *Journal of Machine Learning Research*, 10:341–376, 2009.

I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *Computer Vision and Pattern Recognition*, pages 1–8, 2008.

J. Lawson and Y. Lim. The geometric mean, matrices, metrics, and more. *The American Mathematical Monthly*, 108(9):797–812, 2001.

T. Le, R. Chartrand, and T.J. Asaki. A variational approach to reconstructing images corrupted by Poisson noise. *Journal of Mathematics of Imaging and Vision*, 27(3):257–263, 2007.

H. Lee, A. Battle, R. Rajat, and A. Ng. Efficient sparse coding algorithms. *Advances in Neural Processing Information Systems*, 2005.

A.D. Leow, I. Yanovsky, N. Parikshak, X. Hua, S. Lee, A.W. Toga, C.R. Jack Jr, M.A. Bernstein, P.J. Britson, J.L. Gunter, et al. Alzheimer's disease neuroimaging initiative: a one-year follow up study using tensor-based morphometry correlating degenerative rates, biomarkers and cognition. *Neuroimage*, 45(3):645–655, 2009.

N. Lepore, C. Brun, Y. Chou, M. Chiang, R. Dutton, K. Hayashi, E. Luders, O.L. Lopez, H. Aizenstein, A. Toga, et al. Generalized tensor-based morphometry of HIV/AIDS using multivariate statistics on deformation tensors. *Transactions on Medical Imaging*, 27(1): 129–141, 2007.

S. Li. Concise formulas for the area and volume of a hyperspherical cap. *Asian Journal of Mathematics and Statistics*, 4:66–70, 2011.

X. Li, W. Hu, Z. Zhang, X. Zhang, M. Zhu, and J. Cheng. Visual tracking via incremental Log-Euclidean Riemannian subspace learning. In *Computer Vision and Pattern Recognition*, 2008.

D.C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

T. Liu, A.W. Moore, A. Gray, and K. Yang. An investigation of practical approximate nearest neighbor algorithms. *Advances in neural information processing systems*, 17:825–832, 2004.

W. Liu, J. Wang, S. Kumar, and S. Chang. Hashing with graphs. In *International Conference on Machine Learning*, 2011.

Z. Liu and L. Vandenberghe. Interior-point method for nuclear norm approximation with application to system identification. *SIAM Journal on Matrix Analysis and Applications*, 31(3): 1235–1256, 2009.

D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

M. Moakher, and P. Batchelor. Symmetric positive-definite matrices: From geometry to applications and visualization. *Springer Berlin Heidelberg*, 2006.

B. Ma and Y. Wu. Covariance matching for PDE-based contour tracking. In *International Conference on Image and Graphics*, pages 720–725, 2011.

E. Maggio, E. Piccardo, C. Regazzoni, and A. Cavallaro. Particle PHD filtering for multi-target visual tracking. In *International Conference on Acoustics, Speech and Signal Processing*, 2007.

J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. *International Conference on Machine Learning*, 2009a.

J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11:19–60, 2010.

J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Supervised dictionary learning. *Advances in Neural Processing Information Systems*, 2009b.

J. Mairal, M. Elad, and G. Sapiro. Sparse representation for color image restoration. *Transactions on Image Processing*, 17(1):53–69, 2008a.

J. Mairal, G. Sapiro, and M. Elad. Learning multiscale sparse representations for image and video restoration. *SIAM Multiscale Modeling and Simulation*, 7:214–241, 2008b.

J. Mairal and Bin Yu. Complexity analysis of the LASSO regularization path. In *arXiv:1205.0079v1 [stat.ML]*, 2012.

D.M. Malioutov, M. Cetin, and A.S. Willsky. Homotopy continuation for sparse signal representation. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 733–736, 2005.

S.G. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *Transactions on Signal Processing*, 41(12):3397–3415, 1993.

K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *Transactions on Pattern Analysis and Machine Intelligence*, 27:1615–1630, 2005.

H. Min, N. Papanikolopoulos, C. Smith, and V. Morellas. Feature-based covariance matching for a moving target in multi-robot following. In *Mediterranean Conference on Control and Automation*, 2011.

M. Moakher and P. Batchelor. Symmetric positive-definite matrices: from geometry to applications and visualization. *Visualization and Processing of Tensor Fields, Berlin*, 2006.

T. Moeslund, A. Hilton, and V. Kruger. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104(2-3):90–126, 2006.

G. Mori, S. Belongie, and J. Malik. Shape contexts enable efficient retrieval of similar shapes. In *Computer Vision and Pattern Recognition*, pages 723–730, 2001.

M.C. Motwani, M.C. Gadiya, R.C. Motwani, and F.C. Harris Jr. Survey of image denoising techniques. In *Proceedings of GSPx*, pages 27–30, 2004.

R. Muirhead. *Aspects of multivariate statistical theory*. Wiley Online Library, 1982.

K. Murphy, A. Torralba, D. Eaton, and W. Freeman. Object detection and localization using local and global features. *Toward Category-Level Object Recognition*, pages 382–400, 2006.

J.F. Murray and K. Kreutz-Delgado. Sparse image coding using learned overcomplete dictionaries. *Machine Learning for Signal Processing*, pages 579–588, 2004.

T. Myrvoll and F. Soong. On divergence based clustering of normal distributions and its application to HMM adaptation. In *European Conference on Speech Communication and Technology*, page 1517, 2003.

N. Neal. Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, pages 249–265, 2000.

J. Neyman and E. Pearson. On the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London*, pages 289–337, 1933.

F. Nielsen and S. Boltz. The Burbea-Rao and Bhattacharyya centroids. *Transactions on Information Theory*, 57(8):5455–5466, 2011.

F. Nielsen and R. Nock. On the centroids of symmetrized Bregman divergences. *Arxiv preprint arXiv:0711.3242*, 2007.

F. Nielsen and R. Nock. Jensen-Bregman Voronoi diagrams and centroidal tessellations. In *International Symposium on Voronoi Diagrams in Science and Engineering*, pages 56–65, 2010.

O. Tuzel, F. Porikli, and P. Meer. Covariance Tracking using model update based on Lie Algebra. *Computer Vision and Pattern Recognition*, 2006.

O. Tuzel, F.Porikli, and P. Meer. Region covariance: A fast descriptor for detection and classification. *European Conference on Computer Vision*, 2006.

B.A. Olshausen and D.J. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1. *Vision Research*, 37(23):3311–3325, 1997.

S. Omohundro. *Five balltree construction algorithms*. International Computer Science Institute, 1989.

M.R. Osborne, B. Presnell, and B.A. Turlach. On the LASSO and its dual. *Journal of Computational and Graphical Statistics*, 9(2):319–337, 2000.

Y. Pang, Y. Yuan, and X. Li. Gabor-based region covariance matrices for face recognition. *Transactions on Circuits and Systems for Video Technology*, 18(7):989–993, 2008.

X. Pennec, P. Fillard, and N. Ayache. A Riemannian framework for tensor computing. *International Journal of Computer Vision*, 66(1):41–66, 2006.

J.M. Peyrat, M. Sermesant, X. Pennec, H. Delingette, C. Xu, E. McVeigh, and N. Ayache. Statistical comparison of cardiac fibre architectures. *Functional Imaging and Modeling of the Heart*, pages 413–423, 2007.

P.J. Phillips, H. Moon, S.A. Rizvi, and P.J. Rauss. The FERET evaluation methodology for face-recognition algorithms. *Transactions on Pattern Analysis and Machine Intelligence*, 22 (10):1090–1104, 2000.

P.J. Phillips, H. Wechsler, J. Huang, and P.J. Rauss. The FERET database and evaluation procedure for face-recognition algorithms. *Image and Vision Computing*, 16(5):295–306, 1998.

J. Pitman and J. Picard. *Combinatorial stochastic processes*. Springer, Berlin, 2006.

F. Porikli and T. Kocak. Robust license plate detection using covariance descriptor in a neural network framework. In *International Conference on Video and Signal Based Surveillance*, pages 107–107, 2006.

J. Portilla, V. Strela, M.J. Wainwright, and E.P. Simoncelli. Image denoising using scale mixtures of Gaussians in the wavelet domain. *Transactions on Image Processing*, 12(11):1338–1351, 2003.

J. Puzicha, J.M. Buhmann, Y. Rubner, and C. Tomasi. Empirical evaluation of dissimilarity measures for color and texture. In *International Conference on Computer Vision*, pages 1165–1172, 1999.

M. Raginsky and S. Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In *Advances in Neural Information Processing Systems*, 2009.

P. Ram, D. Lee, H. Ouyang, and A. Gray. Rank-approximate nearest neighbor search: Retaining meaning and speed in high dimensions. *Advances in Neural Information Processing Systems*, 2009.

I. Ramirez, F. Lecumberry, and G. Sapiro. Universal priors for sparse modeling. In *Workshop on Computational Advances in Multi-Sensor Adaptive Processing*, pages 197–200, 2009.

W. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

N. Rasiwasia, P.J. Moreno, and N. Vasconcelos. Bridging the gap: Query by semantic example. *Transactions on Multimedia*, 9(5):923–938, 2007.

W.J. Reed. The normal-Laplace distribution and its relatives. *Advances in Distribution Theory, Order Statistics, and Inference*, pages 61–74, 2006.

J.T. Robinson. The KDB-tree: a search structure for large multidimensional dynamic indexes. *International Conference on Management of Data*, 1981.

A. Rodríguez, D.B. Dunson, and A.E. Gelfand. Nonparametric functional data analysis through Bayesian density estimation. *Biometrika*, 2007.

M. Rojas, S.A. Santos, and D.C. Sorensen. LSTRS: Matlab software for large-scale trust-region subproblems and regularization. *ACM Transactions on Mathematical Software*, 34(2): 11, 2008.

R. Rubinstein, M. Zibulevsky, and M. Elad. Efficient implementation of the K-SVD algorithm using batch orthogonal matching pursuit. *Technion*, 2008.

A. Ruta, Y. Li, M. Uxbridge, F. Porikli, S. Watanabe, H. Kage, K. Sumi, and J. Amagasaki. A new approach for in-vehicle camera traffic sign detection and recognition. In *Conference on Machine Vision Applications*, 2009.

Chunhua S., Junae K., and Lei W. Scalable large-margin Mahalanobis distance metric learning. *Neural Networks*, 21(9):1524 –1530, 2010.

R. Salakhutdinov and G. Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, 2009.

H. Samet. *Foundations of multidimensional and metric data structures*. Morgan Kaufmann, 2006.

I.W. Selesnick. The estimation of Laplace random vectors in additive white Gaussian noise. *Transactions on Signal Processing*, 56(8):3482–3496, 2008.

J. Sethuraman. A constructive definition of Dirichlet priors. *Statistica Sinica*, 4(2):639–650, 1994.

G. Shakhnarovich, P. Viola, and T. Darrell. Fast pose estimation with parameter-sensitive hashing. In *International Conference on Computer Vision*, pages 750–757, 2003.

M. Sharif, N. Ihaddadene, and C. Djeraba. Covariance matrices for crowd behaviour monitoring on the escalator exits. *Advances in Visual Computing*, pages 470–481, 2008.

C. Shen, A. Welsh, and L. Wang. PSDBoost: Matrix-generation linear programming for positive semidefinite matrices learning. *Advanced Neural Information Processing Systems*, pages 1473–1480, 2008.

Y. Shinohara, T. Masuko, and M. Akamine. Covariance clustering on Riemannian manifolds for acoustic model compression. In *International Conference on Acoustics Speech and Signal Processing*, pages 4326–4329, 2010.

R. Sivalingam, D. Boley, V. Morellas, and N. Papanikolopoulos. Tensor sparse coding for region covariances. In *European Conference on Computer vision*, pages 722–735, 2010.

R. Sivalingam, V. Morellas, D. Boley, and N. Papanikolopoulos. Metric Learning for Semi-supervised Clustering of Region Covariance Descriptors. *International Conference on Distributed Smart Cameras*, 2009.

J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *International Conference on Computer Vision*, pages 1470–1477, 2003.

N. Snavely, S.M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3D. In *ACM Transactions on Graphics*, volume 25, pages 835–846, 2006.

S. Sra. Positive definite matrices and the symmetric Stein divergence. *http://arxiv.org/abs/1110.1773*, 2011.

B. Sriperumbudur and G. Lanckriet. On the convergence of the concave-convex procedure. *Advances in Neural Information Processing Systems*, pages 1759–1767, 2009.

E. Sudderth. *Graphical models for visual object recognition and tracking*. PhD thesis, Massachusetts Institute of Technology, 2006.

E. Sudderth, A. Torralba, W. Freeman, and A. Willsky. Describing visual scenes using transformed dirichlet processes. *Advances in Neural Information Processing Systems*, 2006.

K. Thangavel, R. Manavalan, and I.L. Aroquiaraj. Removal of speckle noise from ultra sound medical image based on special filters: Comparative study. *ICGST-GVIP Journal*, 9, 2009.

P. Turaga and R. Chellappa. Nearest-neighbor search algorithms on non-Euclidean manifolds for computer vision applications. In *Indian Conference on Computer Vision, Graphics and Image Processing*, pages 282–289, 2010.

O. Tuzel, F. Porikli, and P. Meer. Human detection via classification on Riemannian manifolds. In *Computer Vision and Pattern Recognition*, pages 1–8, 2007.

J. Vogt, S. Prabhakaran, T. Fuchs, and V. Roth. The translation-invariant Wishart-Dirichlet process for clustering distance data. In *International Conference on Machine Learning*, pages 1111–1118, June 2010.

J. Wang, S. Kumar, and F. Chang. Semi-supervised hashing for scalable image retrieval. In *Computer Vision and Pattern Recognition*, 2010.

Z. Wang, B. Vemuri, Y. Chen, and T. Mareci. A constrained variational principle for direct estimation and smoothing of the diffusion tensor field from complex dwi. *Transactions on Medical Imaging*, 23(8):930–939, 2004.

Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Advances in Neural Information Processing Systems*, volume 21, pages 1753–1760, 2009.

M. West. Hyperparameter estimation in Dirichlet process mixture models. *Duke University*, 1992.

Z.J. Xiang, H. Xu, and P.J. Ramadge. Learning sparse representations of high dimensional data on large scale dictionaries. *Advances in Neural Information Processing Systems*, 2011.

Y. Xiao, T. Zeng, J. Yu, and M.K. Ng. Restoration of images corrupted by mixed gaussian-impulse noise via l1-l0 minimization. *Pattern Recognition*, 44(8):1708–1720, 2011.

E. Xing, M. Jordan, and R. Sharan. Bayesian haplotype inference via the Dirichlet process. *Journal of Computational Biology*, 14(3):267–284, 2007.

J. Yang and Y. Zhang. Alternating direction algorithms for l1-problems in compressive sensing. *SIAM Journal on Scientific Computing*, 33:250–278, 2011.

C. Ye, J. Liu, C. Chen, M. Song, and J. Bu. Speech emotion classification on a Riemannian manifold. *Advances in Multimedia Information Processing*, pages 61–69, 2008.

C. Yuan, W. Hu, X. Li, S. Maybank, and G. Luo. Human action recognition under log-Euclidean Riemannian metric. *Asian Conference on Computer Vision*, pages 343–353, 2010.

A. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Computation*, 15(4): 915–936, 2003.

Jianguo Z., M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. In *Computer Vision and Pattern Recognition Workshop*, 2006.

J. Zepeda, E. Kijak, and C. Guillemot. Approximate nearest neighbors using sparse representations. In *International Conference on Acoustics Speech and Signal Processing*, pages 2370 –2373, 2010.

B. Zhang, J. Fadili, and J. Starck. Wavelets, ridgelets, and curvelets for Poisson noise removal. *Transactions on Image Processing*, 17(7):1093–1108, 2008.

H. Zhang, A.C. Berg, M. Maire, and J. Malik. SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. In *Computer Vision and Pattern Recognition*, volume 2, pages 2126–2136, 2006.

H. Zhang, J. Yang, Y. Zhang, N.M. Nasrabadi, and T.S. Huang. Close the loop: Joint blind image restoration and recognition with sparse representation prior. In *International Conference on Computer Vision*, pages 770–777, 2011.

W. Zheng, H. Tang, Z. Lin, and T. Huang. Emotion recognition from arbitrary view facial images. *European Conference on Computer Vision*, pages 490–503, 2010.

H. Zhu, H. Zhang, J. Ibrahim, and B. Peterson. Statistical analysis of diffusion tensors in diffusion-weighted magnetic resonance imaging data. *Journal of the American Statistical Association*, 102(480):1085–1102, 2007.

Appendices

Appendix A

p-Stable Distributions

Locality sensitive hashing was originally introduced based on the so called p-stable distributions. A probability distribution is said to be *stable*, if a linear combination of random variables from this distribution will preserve the distribution upto a scale and location. Extending this notion of stability, we can define a *p-stable* distribution [Datar et al., 2004] as follows:

Definition A.1. *A distribution $S \subset \mathbb{R}$ is said to be p-stable if there exists a $p \geq 0$ such that for any set of m real numbers $a_1, a_2, \cdots, a_m$ and any set of i.i.d random variables $X_1, X_2, \cdots, X_m$ such that each $X_i \sim S$, we have the property that*

$$\sum_{i=1}^{m} a_i X_i \sim \|X_i\|_p X, \tag{A.1}$$

where $X \sim S$ and $\|.\|_p$ denotes the p norm.

It is known that such stable distributions exist for $p \in (0, 2]$. When $p = 1$, it is the Cauchy distribution, while for $p = 2$, we have the Gaussian distribution. Now we will elucidate how a p-stable distribution helps in LSH. It is clear that the summation in (A.1) can be written as a dot product $\langle a, \hat{X} \rangle$ between a vector $a \in \mathbb{R}^m$ and a set of i.i.d random variables $\hat{X} \in \mathbb{R}^m$ such that each dimension of $X_i \sim S$. Suppose, we have two data vectors $a_1, a_2 \in \mathbb{R}^m$, then it is easy to see that if we sample a set of vectors from S and project the data points onto these vectors, then the difference between these projections: $a_1^T \hat{X} - a_2^T \hat{X}$ follows the distribution $\|a_1 - a_2\|_2 X$, where $X \sim S$. That is, the difference between the projections of the points captures the Euclidean distance between them (assuming S is 2-stable), which is

essentially what we require to achieve locality sensitive hashing. Thus, we can segment the real-line (the line projection) into w separate hash buckets of equal widths, then with high probability neighboring points will fall into the same hash buckets. An obvious hash function here is $h(x) = \lfloor \frac{ax+b}{w} \rfloor$, where a represents the projection lines and b is a real-number chosen from $(0, w]$ uniformly. For more details on this algorithm and asymptotic properties of the hashing performance, see [Datar et al., 2004].

Appendix B

Predictive Distribution for Wishart-Inverse-Wishart DPMM

In this appendix section, we will derive the marginal distribution associated with the Wishart-Inverse Wishart DPMM model. We will assume the following notation through out this section; X denotes a data covariance matrix, and specifically let X_i denotes the i^{th} data matrix. S is a scale matrix corresponding to a hyper prior parameter of the inveres Wishart distribution, r is the number of degrees of freedom, p denotes the dimension of the covariance matrix, Γ_p represents the multivariate Gamma function. The sampling model we assume is as follows: Assuming known S, p and r,

$$\begin{aligned} \Sigma &\sim IW(S, r) \\ X &\sim W(\Sigma, r, p). \end{aligned}$$

From [Eaton, 1983], we have the following form for the Wishart normalization constant:

$$c(r, p) = \frac{1}{2^{rp/2}\Gamma_p(\frac{r}{2})}. \tag{B.1}$$

Using the above notations, we have Wishart Distribution with scale parameter Σ is as follows:

$$W(X|\Sigma) = \frac{|X|^{\frac{r-p-1}{2}}}{c(r, p)|\Sigma|^{\frac{r}{2}}} exp\left(-\frac{1}{2}Tr(\Sigma^{-1}X)\right). \tag{B.2}$$

Inverse-Wishart distribution with scale parameter S is given by:

$$P(\Sigma|S) = \frac{|S|^{\frac{r}{2}}|\Sigma|^{-\frac{(r+p+1)}{2}}}{c(r, p)} exp\left(-\frac{1}{2}Tr(\Sigma^{-1}S)\right). \tag{B.3}$$

Joint Likelihood

Joint Likelihood is derived as follows:

$$P(X,\Sigma) = P(X|\Sigma)\ P(\Sigma|S) \tag{B.4}$$

$$= \frac{1}{c(r,p)^2}|S|^{\frac{r}{2}}|X|^{\frac{r-p-1}{2}}|\Sigma|^{-\frac{(2r+p+1)}{2}} exp\left(-\frac{1}{2}Tr\{\Sigma^{-1}(X+S)\}\right). \tag{B.5}$$

Marginal Distribution

Theorem B.1. *For a Wishart distributed data matrix X, and Inverse-Wishart hyperparameter S, the marginal distribution $P(X|S)$ over the space of all canonical parameter matrices Σ is given by:*

$$P(X|S) = \frac{1}{c(r,p)^2}|S|^{\frac{r}{2}}|X|^{\frac{(r-p-1)}{2}}\frac{c(2r,p)}{|X+S|^{\frac{2r}{2}}}. \tag{B.6}$$

Proof. Suppose that X is a data matrix following a Wishart distribution $W(X|\Sigma)$ and Σ follows an inverse-Wishart distribution $IW(\Sigma|S)$ (where S is a hyperparameter). Also from Proposition 5.16 of [Eaton, 1983], assume:

$$c(r,p) \ = \ \int_{\delta_p^+} |Y|^{\frac{r-p-1}{2}} exp\left[-\frac{1}{2}tr(Y)\right] dY. \tag{B.7}$$

Then,

$$P(X|S) \ = \ \int_{\delta_p^+} W(X|\Sigma)\ IW(\Sigma|S)d\Sigma \tag{B.8}$$

$$= \ \frac{1}{c(r,p)^2}|X|^{\frac{r-p-1}{2}}|S|^{\frac{r}{2}}\int_{\delta_p^+}|\Sigma|^{-\frac{2r+p+1}{2}} exp\left[-\frac{1}{2}tr(\Sigma^{-1}(X+S))\right] d\Sigma. \tag{B.9}$$

Using the Jacobian of transformation, ($\Sigma^{-1} \ \Rightarrow \ \Sigma$), let $\Sigma^{-1} = R$, then Jacobian of transformation is $|\Sigma^{-2p}|d\Sigma = dR$. Since we deal with symmetric matrices, $J_\Sigma(\Sigma^{-1}) = \Sigma^{-(p+1)}$. See the Appendix of [Anderson, 1958] for details of this. Thus the Jacobian can be written as $d\Sigma = \frac{dR}{|R|^{p+1}}$. Using this Jacobian, we can write the (B.9) as:

$$\Rightarrow \ \frac{1}{c(r,p)^2}|S|^{\frac{r}{2}}|X|^{\frac{r-p-1}{2}}\int_{\delta_p^+}|R|^{\frac{2r-p-1}{2}} exp\left[-\frac{1}{2}tr(R(X+S))\right] dR. \tag{B.10}$$

Now, assume

$$R(X+S) = Y \tag{B.11}$$

Then $|X+S|^{\frac{p+1}{2}} dR = dY$. This can be proved as an extension of the following proposition (see [Eaton, 1983] for the proof).

Proposition B.2. *Let A be a $p \times p$ nonsingular matrix and define the function g on the linear space δ_p^+ of $p \times p$ real symmetric matrices by*

$$g(S) \;=\; ASA^T = (A \otimes A)S. \tag{B.12}$$

Then $J_g(S) \;=\; |det(A)|^{p+1}$.

Using the above propostion, where $A = (X+S)$, we get the Jacobian of the transformation (B.11) as $|det(X+S)|^{\frac{p+1}{2}}$. Substituting this in (B.10), we have

$$\Rightarrow \;\frac{1}{c(r,p)^2}|S|^{\frac{r}{2}}|X|^{\frac{r-p-1}{2}} \int_{\delta_p^+} |Y(X+S)^{-1}|^{\frac{2r-p-1}{2}} exp\left[-\frac{1}{2}tr(Y)\right] dY/|X+S|^{\frac{p+1}{2}} \tag{B.13}$$

$$= \frac{1}{c(r,p)^2}\frac{|S|^{\frac{r}{2}}|X|^{\frac{r-p-1}{2}}}{|X+S|^{2r/2}} \int_{\delta_p^+} |Y|^{\frac{2r-p-1}{2}} exp\left[-\frac{1}{2}tr(Y)\right] dY. \tag{B.14}$$

Now, using (B.7) in (B.14), we have

$$P(X) \;=\; \frac{c(2r,p)}{c(r,p)^2}\frac{|S|^{\frac{r}{2}}|X|^{\frac{r-p-1}{2}}}{|X+S|^{\frac{2r}{2}}}. \tag{B.15}$$

□

Posterior Hyperparameters

Assuming the degrees of freedom for each of the sample matrices is the same, when a new covariance matrix gets added to an existing WIW cluster, we have the following update equations for the parameters of this cluster: assuming the new *degrees of freedom* parameter as r' and the new hyperparameter scale matrix S', we have:

$$r' \;=\; r + nr. \tag{B.16}$$

Scale matrix is updated as

$$S' \;=\; S + \sum_{i=1}^{n} X_i. \tag{B.17}$$

Posterior

For a covariance matrix Σ sampled, the posterior distribution takes the following form:

$$P(\Sigma|X) = \frac{P(X|\Sigma)\ P(\Sigma|S)}{P(X)} \tag{B.18}$$

$$= \frac{1}{c(2r,p)}|S+X|^{r}\ |\Sigma|^{-\frac{(2r+p+1)}{2}} exp\left(-\frac{1}{2}Tr\{\Sigma^{-1}(X+S)\}\right). \tag{B.19}$$

Joint Distribution for n data matrices

$$P(X_1X_2...X_n) = \int_{\delta_p^+} P(X_1X_2...X_n|\Sigma)\ P(\Sigma|S)\ d\Sigma \tag{B.20}$$

$$= \frac{1}{c(r,p)^{n+1}}|X_1X_2...X_n|^{\frac{r-p-1}{2}}\ |S|^{\frac{r}{2}} \int_{\delta_p^+} |\Sigma|^{-\frac{(n+1)r+p+1}{2}} exp\left(-\frac{1}{2}Tr\{\Sigma^{-1}(\sum_i X_i\ +\ S)\}\right) d\Sigma \tag{B.21}$$

$$= \frac{c\left((n+1)r,p\right)}{c(r,p)^{n+1}}|S|^{\frac{r}{2}} \frac{\prod_{i=1}^{n}|X_i|^{\frac{(r-p-1)}{2}}}{|\sum_{i=1}^{n}X_i+S|^{\frac{(n+1)r}{2}}}. \tag{B.22}$$

The above derivation uses the conditional independence assumption of the data matrices $X_1, X_2, ...X_n$ given Σ.

Predictive Distribution

Now, we are ready with all the necessary ingredients to derive the predictive distribution using the WIW DPMM model, which is the crux of the Chinese Restaurant Process sampling scheme:

$$P(X_n|X_1X_2...X_{n-1}) = \int_{\delta_p^+} P(X_n|\Sigma)\ P(\Sigma|X_1X_2...X_{n-1})d\Sigma \tag{B.23}$$

$$= \int_{\delta_p^+} \frac{P(X_n|\Sigma)\ P(X_1X_2...X_{n-1}|\Sigma)P(\Sigma|S)}{P(X_1X_2...X_{n-1})}\ d\Sigma \tag{B.24}$$

$$= \int_{\delta_p^+} \frac{P(X_1X_2...X_{n-1}X_n|\Sigma)\ P(\Sigma|S)}{P(X_1X_2...X_{n-1})}\ d\Sigma \tag{B.25}$$

$$= \frac{c\left((n+1)r,p\right)}{c(r,p)\ c(nr,p)}\ \frac{|X_n|^{\frac{(r-p-1)}{2}}\ |\sum_{i=1}^{n-1}X_i+S|^{\frac{nr}{2}}}{|\sum_{i=1}^{n}X_i\ +\ S|^{\frac{(n+1)r}{2}}}. \tag{B.26}$$

Appendix C

Distribution using Riemannian Metric

In this section, we will derive a pseudo-exponential family type probability distribution using the natural Riemannian metric on the symmetric positive definite cone as the base measure. We will refer to this density the *geodesic distribution* as the distance between points on the Riemannian surface will now be geodesics.

Geodesic Distribution

We will start with an alternative definition to the Riemannian metric (slightly different in notation to what we had in Chapter 3).

Definition C.1. *Let $M, X \in \mathcal{S}_{++}^p$, then the geodesic distance d_{geo} between M and X is defined as:*

$$d_{geo}^2(X, M) = \operatorname{Tr} \log^2(M^{-1}X). \tag{C.1}$$

Now, we define a Geodesic distribution of the exponential type (this is technically not an exponential distribution as the base measure is not convex) as follows:

Definition C.2. *For $M, \Sigma \in \mathcal{S}_{++}^p$, for a random matrix $X \in \mathcal{S}_{++}^p$, and let $G(M, \sigma)$ denote the Geodesic distribution with geodesic centroid M and variance σ, then we say $X \sim G(M, \sigma)$ if the probability density of X is given by:*

$$p(X|M, \sigma) := \frac{1}{Z} \exp\left\{-\frac{1}{2\sigma^2} d_{geo}^2(M, X)\right\}, \tag{C.2}$$

where Z is a normalization constant.

Unfortunately, this normalization constant does not have a closed form and thus this distribution is not of much practical use. Nevertheless, in the following subsections, we will provide derivations for this normalization constant Z.

Derivation of the Normalizer

In this section, we compute the normalization constant Z in (C.2). Since we are integrating over the entire cone of SPD matrices, we replace the $M^{-1}X$ in (C.2) by another symmetric SPD matrix S.

$$Z = \int_{\mathcal{S}^p_{++}} \exp\big(-\operatorname{Tr}\log^2(S)\big)\partial S. \tag{C.3}$$

Next, let the SVD decomposition of $S = U\Sigma U^T$. To compute the Jacobian, we take differentials on both sides of (C.3):

$$\partial S = \partial U\Sigma U^T + U\partial\Sigma U^T + U\Sigma\partial U^T. \tag{C.4}$$

Now, using $U^TU = I$, we have $\partial U^TU = -U^T\partial U$, which when substituted in the above equation gives:

$$\partial S = \partial U\Sigma U^T + U\partial\Sigma U^T - U\Sigma U^T\partial UU^T \tag{C.5}$$

$$U^T\partial SU = U^T\partial U\Sigma - \Sigma U^T\partial U + \partial\Sigma. \tag{C.6}$$

If $(\partial\Sigma)^\wedge = \partial\sigma_1\partial\sigma_2\cdots\partial\sigma_p$, then using wedge products [Muirhead, 1982], we have[1]

$$U^T\partial SU = (\Sigma\otimes U - (I\otimes U\Sigma))\partial U + \partial\Sigma \tag{C.7}$$

$$(\partial S)^\wedge = \prod_{i<j}|\sigma_i - \sigma_j|(\partial\Sigma)^\wedge((U^T\partial U))^\wedge, \tag{C.8}$$

where σ_i is the ith diagonal element of Σ and the last reduction uses the idea that the wedge product volume is invariant to a linear transformation. Using (C.8) as Jacobian and substituting in (C.3), we have:

$$Z = \int_{\mathbb{R}^{p+}} \exp\big(-\operatorname{Tr}\log^2(\Sigma)\big)\prod_{i<j}|\sigma_i - \sigma_j|(\partial\Sigma)^\wedge\int_{\mathcal{O}_p}(U^T\partial U)^\wedge. \tag{C.9}$$

[1] Also see http://web.mit.edu/18.325/www/handouts/handout3.pdf for details

Let use write $Z = Z_1 Z_2$ for the two separable integrals in (C.9) respectively. It is easy to see that the second integral is the volume of the *Stiefel* manifold of dimension p, which is

$$Z_2 = Vol(V_{p,p}) = \frac{2^p \pi^{p^2/2}}{\Gamma_p(p/2)}. \tag{C.10}$$

Now, for the first part of (C.9), using the substitution $\sigma_i = \exp \sigma_i$, we have:

$$Z_1 = \int_{\mathbb{R}^p} \prod_i \exp\left(-\sigma_i^2\right) \prod_{i<j} |\exp \sigma_i - \exp \sigma_j| (\partial\Sigma)^{\wedge}, \tag{C.11}$$

which can be separated into p independent integrals on $\mathbb{R}^1$ and can be evaluated analytically.