Sparse Models for Positive Definite Matrices

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

Ravishankar Sivalingam

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

Nikolaos P. Papanikolopoulos, Adviser

February, 2015

Acknowledgements

I sincerely thank my adviser Prof. Nikolaos Papanikolopoulos for his mentorship and guidance during my graduate education, and his trust and support during this period. I also thank my co-advisors Prof. Vassilios Morellas and Prof. Daniel Boley, for their guidance. In addition, I thank Profs. Arindam Banerjee, Gilad Lerman, Guillermo Sapiro, Jarvis Haupt for their collaboration and support of this thesis topic. I would also like to thank Prof. Tom Luo, since this thesis was spawned from a class project in his Convex Optimization course. I express my deepest appreciation for these and countless other collaborators in the course of my graduate education.

I thank all of my colleagues in the Distributed Robotics Laboratory, who provided great peer support beginning from my entry into research to the completion of my dissertation. I would specifically like to thank Dr. Guruprasad Somasundaram, a great friend, colleague, and collaborator of many years. I also thank my various funding agencies which have helped me with my research, learn and grow to produce this thesis.

I thank all my friends and family, who have stood by me throughout my graduate education, providing great moral support. While these are too numerous to name here, I would specifically like to thank my best friends: Arvind Karthikeyan, Aswini Gauthama Sankar, Avanidhar Chandrasekharan, and Priyadharshini Vijayakumar.

Most importantly, I sincerely thank my parents, Mr. E. Sivalingam and Mrs. Nirmaladevi Sivalingam, and my sister Mohanapriya Sivalingam, for their moral and financial support not just during my graduate education, but throughout my life.

Dedication

To my parents

Abstract

Sparse models have proven to be extremely successful in image processing, computer vision and machine learning. However, a majority of the effort has been focused on vector-valued signals. Higher-order signals like matrices are usually vectorized as a pre-processing step, and treated like vectors thereafter for sparse modeling. Symmetric positive definite (SPD) matrices arise in probability and statistics and the many domains built upon them. In computer vision, a certain type of feature descriptor called the region covariance descriptor, used to characterize an object or image region, belongs to this class of matrices. Region covariances are immensely popular in object detection, tracking, and classification. Human detection and recognition, texture classification, face recognition, and action recognition are some of the problems tackled using this powerful class of descriptors. They have also caught on as useful features for speech processing and recognition.

Due to the popularity of sparse modeling in the vector domain, it is enticing to apply sparse representation techniques to SPD matrices as well. However, SPD matrices cannot be directly vectorized for sparse modeling, since their implicit structure is lost in the process, and the resulting vectors do not adhere to the positive definite manifold geometry. Therefore, to extend the benefits of sparse modeling to the space of positive definite matrices, we must develop dedicated sparse algorithms that respect the positive definite structure and the geometry of the manifold.

The primary goal of this thesis is to develop sparse modeling techniques for symmetric positive definite matrices. First, we propose a novel sparse coding technique for representing SPD matrices using sparse linear combinations of a dictionary of atomic SPD matrices. Next, we present a dictionary learning approach wherein these atoms are themselves learned from the given data, in a task-driven manner. The sparse coding and dictionary learning approaches are then specialized to the case of rank-1 positive semi-definite matrices. A discriminative dictionary learning approach from vector sparse modeling is extended to the scenario of positive definite dictionaries. We present efficient algorithms and implementations, with practical applications in image processing and computer vision for the proposed techniques.

Contents

Acknowledgements **i**

Dedication **i**

Abstract **ii**

List of Tables **viii**

List of Figures **ix**

1 Introduction **1**

1.1 The Need for Dedicated Positive Definite Sparse Models 2

1.2 Organization of the Thesis 3

2 Related Work **5**

2.1 Positive Definite Matrices 5

2.2 Region Covariance Descriptors 8

2.3 Sparse Modeling 13

2.3.1 Sparse Coding 13

2.3.2 Dictionary Learning 16

2.4 Sparse Models for Positive Definite Matrices 18

3 Positive Definite Sparse Coding **20**

3.1 The Tensor Sparse Coding Problem . . . 22

3.2 The LogDet Divergence . . . 24

3.3 The Tensor Sparse Coding Formulation . . . 26

3.4 The MAXDET problem . . . 29

3.4.1 Type I: ℓ_1-regularized Sparse Coding . . . 29

3.4.2 Type II: ℓ_1-constrained Sparse Coding . . . 30

3.4.3 Semidefinite Signals . . . 31

3.5 Effect of Sparsity Constraints . . . 32

3.6 Effect of Atom Normalization . . . 35

3.7 Atom Coherence . . . 37

3.8 Completeness and Coefficient Recovery . . . 38

3.9 Comparison with Vector Sparse Coding . . . 41

3.10 Relation between D_{geo} and D_{ld} . . . 44

3.11 Relaxation of the Residual Constraint . . . 48

3.12 An Efficient Sparse Coding Algorithm . . . 50

3.13 Tensor Sparse Coding for Classification . . . 56

3.13.1 Human Appearance Descriptors . . . 58

3.13.2 Tensor Sparse Coding for Face Recognition . . . 61

3.13.3 Tensor Sparse Coding for Texture Classification . . . 64

3.13.4 Action Recognition with Kinect Motion Capture . . . 65

4 Positive Definite Dictionary Learning **69**

4.1 Dictionary Learning Formulation . . . 70

4.2 Approach: Alternating Minimization . . . 71

4.3 Atom Update . . . 72

4.3.1 Gradient Descent . . . 75

4.3.2 Newton Descent 76
4.3.3 Comparison of Atom Update Techniques 78
4.3.4 Matrix Conjugate Gradient 80
4.4 Online Dictionary Learning 83
4.5 Time Complexity 84
4.5.1 Sparse Coding 84
4.5.2 Dictionary Learning 84
4.6 Face Detection with Tensor Dictionary Learning 85
4.7 Cancer Tissue Classification with Positive Definite Dictionaries 87

5 Rank-One Tensor Dictionaries **90**
5.1 Efficient Sparse Coding over Rank-1 Dictionaries 92
5.2 Experiments with Rank-1 Sparse Coding 96
5.3 Learning Rank-1 Dictionaries 99
5.3.1 Gradient Descent 101
5.3.2 Newton Descent 101
5.3.3 Reconstruction Update 102
5.4 Time Complexity 103
5.4.1 Sparse Coding 103
5.4.2 Dictionary Update 103
5.5 Atom Recovery 105
5.6 USPS Digits Classification with Rank-1 Dictionaries 106

6 Discriminative Dictionary Learning **108**
6.1 Discriminative Dictionary Learning Formulation 109
6.2 Discriminative Dictionaries for Brodatz Textures 113
6.3 USPS Digits Classification with Discriminative Dictionary Learning 115
6.4 Action Recognition with Discriminative Learned Dictionaries 116

7 TeSLa: Tensor Sparse Library **119**

7.1 Sparse Coding 120

7.2 Dictionary Learning 122

7.3 Discriminative Dictionary Learning 123

8 Summary and Contributions **124**

8.1 Contributions of the Thesis 124

8.2 Future Directions 125

References **127**

List of Tables

3.1 Mean classification accuracy for the *Cam5* dataset 60
3.2 Features used in construction of region covariances for face recognition on the FERET face dataset. 61
3.3 Mean classification accuracy for the *FERET* face recognition dataset. 62
3.4 Mean classification accuracy for the *Brodatz* mosaic dataset. 65
3.5 Cross-validation accuracies over the Kinect motion capture dataset 68
4.1 Tissue classification results with positive definite dictionaries 89
6.1 Discriminative dictionary learning on the USPS dataset 115
6.2 Action recognition on the KTH dataset 118

List of Figures

3.1 Data points S on the manifold of positive definite matrices are to be represented by a linear combination of atoms A_i from the dictionary $\mathcal{A}$. 22

3.2 Convex feasible set for the signal reconstruction 31

3.3 Effect of sparsity constraints - ℓ_1-regularized sparse coding 33

3.4 Effect of sparsity constraints - ℓ_1-constrained sparse coding 34

3.5 Effect of atom normalization 36

3.6 Phase transition diagram for sparse coefficient recovery when $\mathbf{x} \geq 0$ 40

3.7 Comparison of 1-NN, tensor and vector sparse coding 43

3.8 Comparison of dissimilarity measures D_{ld} and D_{geo} 47

3.9 Improvement in reconstruction quality with relaxation of the residual constraint 49

3.10 Coordinate descent algorithm for sparse coding 52

3.11 Average sparse coding times 54

3.12 Average sparse coding times for varying λ 55

3.13 Representative images from the *Cam5* dataset. 59

3.14 Variation in classification accuracy (texture 12) with parameter choice. . . . 66

3.15 Screenshots from the OpenNI tracking program 67

4.1 Comparison of atom update methods 79

4.2 Average speedup of matrix conjugate gradient 82

4.3 Face detection results with positive definite dictionaries 86

4.4 Tissue image analysis with region covariance dictionaries 88

5.1 Coefficient error for sparse coding over rank-1 dictionaries 97
5.2 Objective function improvement for sparse coding over rank-1 dictionaries . . 98
5.3 Average sparse coding times for rank-1 dictionaries 104
5.4 Recovery of rank-1 atoms through dictionary learning 105
5.5 USPS sample digit images 106
5.6 USPS digit classification with rank-1 dictionaries 107
6.1 Reduction in coherence between dictionaries with discriminative learning . . 112
6.2 Discriminative dictionary learning on brodatz textures 114
7.1 Average sparse coding computation times for the TeSLa software 121

List of Algorithms

1 **Coordinate Descent Algorithm for Positive Definite Sparse Coding** 52

2 **Dictionary Learning** 74

3 **Atom Update** 74

4 **Matrix Conjugate Gradient** 81

5 **Efficient Sparse Coding for Rank-1 Dictionaries** 95

6 **Rank-1 Dictionary Learning** 100

7 **Rank-1 Atom Update** 100

8 **Discriminative Dictionary Learning** 111

Chapter 1

Introduction

The past decade has witnessed an enormous growth in the development of theory and algorithms for sparse representation and modeling. Sparse linear models have proven to be extremely successful in image processing, computer vision and machine learning. Image reconstruction, compression, denoising, inpainting and segmentation, and object detection, classification and recognition are some of the problems in image processing and computer vision which have benefited from the application of sparse representation techniques. However, a majority of the effort has been focused on vector-valued signals. Higher-order signals like matrices are usually vectorized as a pre-processing step, and treated like vectors thereafter for sparse modeling.

Symmetric positive definite (SPD) matrices arise in probability and statistics and the many domains built upon them. They also appear in control systems, diffusion tensor imaging and various other fields. In computer vision, region covariances are SPD matrices used as feature descriptors to characterize an object or image region. Since their introduction as image features in 2006, region covariance descriptors (RCDs) have become immensely popular in object detection, tracking, and classification. Human detection and recognition, texture classification, and face recognition are some of the problems tackled using this powerful class of descriptors. The employment of spatio-temporal features for region covariances also helps in mining information from video data, such as for action recognition. Region covariances have also caught on as useful features for speech processing and recognition. The diffusion tensors used to represent water diffusion in human tissue in DT-MRI, kernel matrices in machine learning and dynamic covariances used for modeling time-varying phenomena such as stock prices and climate variables are some other examples where positive definite matrices are encountered.

1.1 The Need for Dedicated Positive Definite Sparse Models

Due to the popularity of sparse modeling in the vector domain, it is enticing to apply sparse representation techniques to SPD matrices as well. The advantages of sparse modeling include not only compact representations, but also the development of interpretable models. The components of the learned models usually incorporate semantic information relevant to the problem domain. In images, there is an inherent sparsity in image patches under certain bases (for *e.g.*, DCT). Similarly, there can be inherent sparsity in the positive definite signals we are trying to model, based on the data they are derived from. Even if the SPD matrices themselves are not generated from sparse models, the use of sparsity has been shown to be a suitable regularization technique for learning classifiers, where parsimonious models are preferred. Further, a sparse linear model can be used as a powerful hyper-prior for parameters in certain exponential family models, such as for the covariance (or precision) matrix of a multivariate Gaussian model. In machine learning, data-dependent kernels are learned by selecting a sparse subset of parametric kernels whose combination performs the best classification. The use of sparsity in image denoising and inpainting applications suggests the use of similar models for positive definite tensor fields, such as those occurring in diffusion tensor imaging (DTI).

However, SPD matrices cannot be directly vectorized for sparse modeling, as is often done for general matrices. Positive definite matrices bear an implicit structure in their eigenvalues, which become meaningless upon vectorization. Further, the geometry of the manifold of SPD matrices is lost when they are vectorized. Therefore to extend the benefits of sparse modeling to the space of positive definite matrices, it is imperative to develop dedicated sparse algorithms that respect the positive definite structure and the geometry of the manifold. A new sparse toolbox for handling positive definite data can significantly enhance the state-of-the-art techniques in the many fields in which they occur.

1.2 Organization of the Thesis

The thesis is organized as follows:

- In Chapter 2 we give an overview of positive definite matrices, the related work on covariance descriptors, and sparse coding and dictionary learning approaches in the vector sparse modeling domain.

- Chapter 3 proposes a novel sparse coding approach for representing positive definite matrices as a non-negative linear combination of a dictionary of atomic positive (semi-) definite matrices. The effect of the various parameters involved are analyzed, and an efficient coordinate descent approach is developed to solve the sparse coding problem. Practical applications of positive definite sparse coding are shown, showing the suitability of the approach to real-world computer vision problems.

- Chapter 4 develops a sparse modeling procedure to learn positive definite dictionaries from the training data, in a task-driven manner. The non-convex formulation is tackled using an alternating minimization approach, and an online generalization is also briefly outlined.

- In Chapter 5, we extend the positive definite sparse coding and dictionary learning approaches to rank-1 semidefinite dictionary atoms, and derive efficient algorithms suited to this specialized case.
- Chapter 6 incorporates the cross-coherence between different class dictionaries into the learning procedure, and provides a way to learn the dictionaries in a discriminative manner.
- Chapter 7 presents the Tensor Sparse Library (TeSLa), a collection of optimized C++ implementations of the algorithms presented in this thesis.
- We conclude the thesis in Chapter 8, and discuss potential future research directions.

Chapter 2

Related Work

2.1 Positive Definite Matrices

An $n \times n$ symmetric[1] matrix A is said to be *positive semidefinite* (denoted by $\mathbf{S}_{+}^{n}$) if, for any non-trivial vector $\mathbf{v} \in \mathbf{R}^n$,

$$\mathbf{v}^T A \mathbf{v} \geq 0. \tag{2.1}$$

When the above holds with strict inequality, A is said to be *positive definite* (denoted by $\mathbf{S}_{++}^{n}$). Another equivalent definition of a positive definite (or semidefinite) matrix is that it has only positive (or non-negative) eigenvalues. A positive definite (semidefinite) matrix is a natural generalization to a positive (non-negative) number.

Conversely, a *negative definite* (*semidefinite*) matrix is one in which $\mathbf{v}^T A \mathbf{v} < 0$ (≤ 0) for any non-trivial vector $\mathbf{v} \in \mathbf{R}^n$. The eigenvalues are appropriately negative or non-positive. If neither of these inequalities hold, and A has both positive and negative eigenvalues, it is denoted as *indefinite*.

[1] Throughout this work, positive (semi-)definite indicates symmetric matrices only. Although it is possible to have a non-symmetric matrix A such that $\mathbf{v}^T A \mathbf{v} \geq 0$, we are interested only in the symmetric case, since most of the data we consider with - covariances, kernels, diffusion tensors - are all symmetric matrices by construction.

The space of $n \times n$ positive definite[2] matrices forms a connected Riemannian manifold. Given two PD matrices A and B, the Riemannian distance metric $D_{\text{geo}}(A, B)$ gives the length of the geodesic connecting these two points on this manifold. This is given by [Pennec et al., 2006],

$$D_{\text{geo}}(A, B) = \left\|\log\left(B^{-1/2} A B^{-1/2}\right)\right\|_F, \tag{2.2}$$

where $\log(\cdot)$ represents the matrix logarithm and $\|\cdot\|_F$ is the Frobenius norm. This can also be written as

$$D_{\text{geo}}(A, B) = \sqrt{\sum_{i=1}^{n} \log^2 \lambda_i\left(A, B\right)}, \tag{2.3}$$

where $\lambda_i\left(A, B\right), i = 1, \ldots, n$ are the generalized eigenvalues of (A, B). The geodesic distance is affine-invariant, in that any non-singular transformation on the covariances does not change the distance:

$$D_{\text{geo}}(XAX^T, XBX^T) = D_{\text{geo}}(A, B) \quad \text{for any invertible } X. \tag{2.4}$$

The logarithm map of $A \in \mathbf{S}_{+}^{n}$ at $B \in \mathbf{S}_{++}^{n}$ associated with the Riemannian manifold is:

$$\log_B(A) = B^{1/2} \log\left(B^{-1/2} A B^{-1/2}\right) B^{1/2}, \tag{2.5}$$

and the exponential map of A at B is:

$$\exp_B(A) = B^{1/2} \exp\left(B^{-1/2} A B^{-1/2}\right) B^{1/2}, \tag{2.6}$$

where log and exp are the matrix logarithm and matrix exponential respectively. The logarithm map is associated with the projection of A onto the tangent space of the Riemannian manifold at B, which is given by $\log\left(B^{-1/2} A B^{-1/2}\right)$. The tangent space of PD matrices $\mathbf{S}_{++}^{n}$ is the space of $n \times n$ symmetric matrices $\mathbf{S}^n$, which is Euclidean. It does not have structure in the eigenvalues like the points on the positive definite manifold, and symmetry is the

[2] We will refer to symmetric positive definite matrices as SPD or PD matrices (symmetry is implicitly understood.) Symmetric positive semidefinite matrices are denoted as SPSD or PSD.

only constraint. Therefore, it is possible to vectorize the upper triangular part of symmetric matrix for further processing. The Riemannian manifold and its tangent space both have dimension $n(n+1)/2$.

The geodesic distance metric is computationally intensive in that when computing distances between all pairs of matrices in a set of N PD matrices we need to solve $N(N-1)/2$ generalized eigenvalue problems. Therefore, as an approximation, [Arsigny et al., 2006] propose another metric known as the Log-Euclidean metric, given by:

$$D_{\mathrm{LE}}(A, B) = \|\log A - \log B\|_F \,. \tag{2.7}$$

This is essentially the Euclidean distance between the projections of A and B onto the tangent space of the manifold at the identity matrix. The Log-Euclidean metric is a lower bound on the actual geodesic distance [Bhatia, 2007], and is exact when the two matrices commute. To compute all the pairwise distances in a set of N PD matrices, we only need to solve N generalized eigenvalue problems. Many works in the literature use this distance metric due to its efficiency.

Positive definite and semidefinite matrices arise in many domains. Covariance (and precision) matrices in probability and statistics are, in general, positive semidefinite. PD/PSD matrices also occur in control systems, and as kernel matrices in machine learning. In medical imaging, there is a new technique known as diffusion tensor imaging (DTI), where each voxel that is imaged is represented as a 3×3 positive definite matrix, called the diffusion tensor. The principal eigenvalue and eigenvector of this matrix give the physical magnitude and direction of diffusion of water molecules in that voxel. The fact that positive definite matrices should be treated as such, without vectorization, is most evident in this example, since the positive eigenvalue actually signifies physical magnitude. The magnitude of water diffusion cannot be negative, and so it would not make sense to have an indefinite or negative semidefinite tensor. Therefore, any processing step should not violate the positive definiteness of the diffusion tensors.

In recent literature, covariances have been used extensively as feature descriptors of image regions in computer vision and image processing. We will elaborate on these region descriptors in the next section.

2.2 Region Covariance Descriptors

Region Covariance Descriptors (RCDs) were introduced by [Tuzel et al., 2006] as a novel region descriptor for object detection and texture classification. Given an image $\mathcal{I}$, let ϕ define a mapping function that extracts an n-dimensional feature vector z_i from each pixel $i \in \mathcal{I}$, such that

$$\phi(I, x_i, y_i) = z_i \ , \tag{2.8}$$

where $z_i \in \mathbb{R}^n$, and (x_i, y_i) is the location of the i^{th} pixel. A given image region R is represented by the $n \times n$ covariance matrix C_R of the feature vectors $\{z_i\}_{i=1}^{|R|}$ of the pixels in region R. Thus the region covariance descriptor is given by

$$C_R = \frac{1}{|R| - 1} \sum_{i=1}^{|R|} (z_i - \mu_R)(z_i - \mu_R)^T \ , \tag{2.9}$$

where, μ_R is the mean vector,

$$\mu_R = \frac{1}{|R|} \sum_{i=1}^{|R|} z_i \ . \tag{2.10}$$

The feature vector z usually consists of color information (in some preferred color–space, usually RGB) and information about the first and higher order spatial derivatives of the image intensity, depending on the application intended.

Although covariance matrices can be positive semi–definite in general, the covariance descriptors themselves are regularized by adding a small constant multiple of the identity matrix, making them strictly positive definite. Thus, the region covariance descriptors belong to $\mathbb{S}^n_{++}$.

As mentioned earlier, the geodesic distance is affine-invariant under a non-singular transformation X. This corresponds to a linear transformation of the feature vectors $z_i \mapsto Xz_i$. Region covariances are invariant to illumination, orientation and scale of the image region, depending on the features used and how the regions are defined. Many existing classification algorithms for region covariances use the geodesic distance in a K-nearest-neighbor framework. The geodesic distance can also be used with a modified K-means algorithm for clustering.

[Porikli and Tuzel, 2006] describe a technique for fast construction of region covariances for rectangular image windows, using integral images, enabling the use of these compact features for many practical applications that demand real–time performance.

[Wildenauer et al., 2007] incorporate the region covariances with connected regions from multi-scale segmentations to efficiently segment textures. [Tou et al., 2009] use Gabor-based covariance descriptors for texture classification. [Ge and Yu, 2008a] perform scene classification by regarding them as textures, using vectorized region covariances as inputs to SVM classifiers.

[Porikli et al., 2006] use the covariance descriptors for tracking non-rigid objects with an update mechanism based on a Lie algebra defined at the tangent space of the identity matrix.

[Prakash et al., 2007, Sharif et al., 2008b, Wu et al., 2008, Wang et al., 2009, Wu et al., 2009a, Wu et al., 2009b, Yinghui and Jianjun, 2009, Ding et al., 2010, Austvoll and Kwolek, 2010] all use the covariance descriptors for object tracking. [Li et al., 2008], [Wang and Yagi, 2008] use the covariance descriptor with the Log-Euclidean distance metric for robust object tracking. In [Ge and Yu, 2008b, Hu et al., 2008, Palaio and Batista, 2008, Palaio et al., 2009, Palaio and Batista, 2009a, Palaio and Batista, 2009b] region covariances are combined with particle filters for object tracking. In [Kwolek, 2009], the author describes a particle swarm optimization algorithm for object tracking using region covariances. [Karasev et al., 2008] compute a kernel-weighted region covariance for object tracking. [Kwon and Park, 2008] perform visual tracking using incremental principal geodesic analysis. [Arif and Vela, 2009] introduce a kernelized version of the region covariance using kernel PCA and apply it to object tracking. [Artner et al., 2009] apply an elaborate framework for tracking articulated objects in a coarse-to-fine pyramidal approach. [Zheng et al., 2009] apply a manifold learning method for tracking people with region covariances. [Wang and Wu, 2010] perform object tracking using region covariances by incrementally learning a low-dimensional model for the covariances in an adaptive manner.

In [Porikli and Kocak, 2006], the authors develop an algorithm for robust license plate detection using covariance descriptors, by using them as feature inputs for a neural network. [Ruta et al., 2009] use region covariances, among other features, for traffic sign detection. In [Osman, 2009a, Osman, 2009b], a hardware setup for object recognition using region covariances is described, along with an online variation of the random forests classifier. In [Cargill et al., 2009], the authors provide a performance evaluation of the covariance descriptor as a suitable feature for generic target detection.

In [Tuzel et al., 2007, Tuzel et al., 2008], region covariances are vectorized and used in a cascade of LogitBoost classifiers for pedestrian detection. [Martelli et al., 2010] present an FPGA architecture for classification based on the above algorithm. [Gualdi et al., 2009] also use covariance descriptors in a LogitBoost framework for human detection, but also incorporate motion information as well as scene structure. In [Gualdi et al., 2010], they further incorporate relevance feedback along with weak calibration of the scene as contextual information for improved human detection. [Paisitkriangkrai et al., 2008b] also use a cascade of boosted classifiers based on region covariances for pedestrian detection. [Paisitkriangkrai et al., 2007, Paisitkriangkrai et al., 2008a, Paisitkriangkrai et al., 2008c] compare the performance of covariance descriptors with other state-of-the-art image features for pedestrian detection. [Hussein et al., 2009] provide a comprehensive evaluation of different features for the problem of human detection.

In [Ma et al., 2007], the authors use region covariances for image retrieval in a multi-camera surveillance setting. Each person in each frame is represented by the covariance descriptor of that region, and the geodesic distance is used as a metric for query-based retrieval of other frames where the person was present. In [Alahi et al., 2008], region covariances are used for object matching across two cameras, which are described as a master-slave setup. [Cai et al., 2010] use region covariances for matching groups of people across cameras with non-overlapping fields-of-view. [Sivalingam et al., 2009] describe a framework for metric learning over positive semi-definite matrices, for the semi-supervised clustering of human appearance descriptors across multiple cameras. [Kwolek, 2010, Kuo et al., 2010] also use region covariances for inter-camera association. In [Sharif et al., 2008a], crowd behavior is monitored to detect abnormal events using covariance matrices computed over the optical flow of crowd motion.

In [Pang et al., 2008], the authors use covariance descriptors computed over Gabor filter responses for face recognition. [Huo and Feng, 2010] combine Gabor-based region covariances with an Active Appearance Model (AAM) for face recognition. [Zheng et al., 2010] recognize facial emotions using covariance descriptors within a Bayesian discriminant analysis framework. In [Han et al., 2009], a symmetric correlation matrix of directional features is used for palmprint recognition. [Lu et al., 2009] use covariance descriptors on Gabor filter responses for palmprint recognition.

[Guo et al., 2010b] use region covariances based on optical flow for action recognition. In [Yuan et al., 2010], the authors also perform action recognition using the Log-Euclidean distance metric. In [Guo et al., 2010a], the covariance descriptors are taken to the tangent space, by the logarithm map, which is Euclidean and vector sparse coding is performed in this space. The resultant algorithm performs extremely well for action recognition in video.

Region covariances have spread to some other domains as well. [Ye et al., 2008] use covariance descriptors computed over acoustic features for speech emotion recognition.

[Shinohara et al., 2010] perform clustering of covariance descriptors for acoustic applications. [Kilic et al., 2010] use region covariances for classification of images of colonic polyps in CT-colonography.

[Porikli, 2010] provides a collective description of most of the different learning algorithms used above for region covariances. The most successful algorithms are those which respect the structure of the Riemannian manifold. Hence it is imperative to do the same for the development of successful sparse representation models in this domain.

Next we explore some of the related work on sparse representation and modeling in the vector domain.

2.3 Sparse Modeling

Linear regression involves the representation of an output signal, or response, by a linear combination of a set of input signals, or predictor variables. These inputs constitute a *dictionary*. The set of regression coefficients obtained may be dense, *i.e.*,, the response may depend on all of the inputs. However, in many practical scenarios or due to the preference of parsimonious models, the output is modeled as depending only on a sparse subset of the inputs.

Sparse linear regression, or *sparse coding*, involves the representation of a signal by a linear combination of a sparse subset of signals from a dictionary. It is a fundamental tool required for the development of sparse linear models.

In this section, we review the relevant literature on vector sparse coding and modeling techniques. The development of sparse representation models involves two primary steps:

Sparse coding Sparse linear regression, or sparse coding, involves the decomposition of a given signal $\mathbf{x}$ in terms of sparse linear combination α of atoms from a fixed (usually over-complete) dictionary D.

Dictionary Learning In many applications, it is also desirable to learn this dictionary D in a data-driven manner, to encode prior knowledge about the problem domain. Given a sufficiently large set of training signals $\mathcal{X} = \{\mathbf{x}_i\}$, a sparsity-promoting, over-complete dictionary D and the corresponding sparse coefficients $\mathcal{A} = \{\alpha_i\}$ are learned.

2.3.1 Sparse Coding

Sparse linear regression and sparse signal recovery can be considered as two faces of the same coin. In the former, we are given a signal $\mathbf{x}$ and dictionary D, and we attempt to find a decomposition α of the signal by a linear combination of a sparse subset of columns from the dictionary, called atoms. In sparse signal recovery, we assume the signal of interest is α, and D is a measurement matrix. The measurements available $\mathbf{x}$ consist of inner products of the rows of the dictionary D with the signal α, and the goal is to recover α by solving this under-determined system of equations. In essence, both these problems are equivalent, but the different interpretations give rise to different ways of analyzing the sparse coding process.

Ideally, sparsity is quantified by the ℓ_0 'pseudo-norm', which is the number of non-zero elements in a vector. The sparse coding problem is given by

$$\min_{\alpha} \qquad \|\alpha\|_0 \tag{2.11}$$

$$s.t. \qquad \mathbf{x} = D\alpha \ , \tag{2.12}$$

or, in a sparsity-constrained version given by

$$\min_{\alpha} \qquad \|\mathbf{x} - D\alpha\|_2^2 \tag{2.13}$$

$$s.t. \qquad \|\alpha\|_0 \leq T \ , \tag{2.14}$$

where T is a constraint on the maximum number of non-zero elements allowed in α. As the ℓ_0-term is a non-convex function, solving for the exact optimum is not possible. The problem is combinatorial and requires time exponential in the dimension of α. However, greedy approximation algorithms such as the Matching Pursuit (MP) of [Mallat and Zhang, 1993], the Orthogonal Matching Pursuit (OMP) of [Pati et al., 1993], [Davis et al., 1997] have been used extensively, and performance guarantees for OMP and necessary conditions for optimality are provided in [Tropp, 2004, Tropp and Gilbert, 2007]. These algorithms sequentially select the best atom in the dictionary for reducing the current objective, in a greedy fashion. [Donoho et al., 2006] also introduced the Stagewise OMP (StOMP) procedure, which selects sets of atoms at each step.

A convex relation of the ℓ_0 constraint, involves regularization or constraint by using the ℓ_1 norm of the signal. [Tropp, 2006] elaborates in detail the convex relation of the above problem, especially in the presence of noise, and proves that under certain conditions, solving for the ℓ_1 criterion gives almost the exact solution for the ℓ_0 problem. Some of the ℓ_1-regularized/constrained formulations include the LASSO of [Tibshirani, 1996] as well as the Basis Pursuit (BP) and the Basis Pursuit De-Noising (BPDN) problems by [Chen et al., 2001]. Different algorithms have been introduced to solve the ℓ_1 sparse coding problems, such as LARS/homotopy-based methods [Efron et al., 2004, Osborne et al., 2000]. The many variants of the ℓ_1 sparse coding problem are:

$$\begin{aligned} \min_{\alpha} \quad & \|\alpha\|_1 \\ s.t. \quad & \mathbf{x} = D\alpha \quad \text{or} \quad \|\mathbf{x} - D\alpha\|_2^2 \leq \epsilon \ , \end{aligned}$$

or

$$\begin{aligned} \min_{\alpha} \quad & \|\mathbf{x} - D\alpha\|_2^2 \\ s.t. \quad & \|\alpha\|_1 \leq T \ , \end{aligned}$$

or

$$\min_{\alpha} \quad \|\mathbf{x} - D\alpha\|_2^2 + \lambda\|\alpha\|_1 \ .$$

[Wright et al., 2009] apply sparse representation for face recognition, with immense success. They do not learn a dictionary, but simply use the training data directly as the dictionary. [Wright et al., 2010] provide a review of some successful applications of sparse representation techniques to solve problems in computer vision.

2.3.2 Dictionary Learning

The K-SVD algorithm for dictionary learning was developed by [Aharon et al., 2006] for learning an over-complete dictionary D from a training set of signals $\mathcal{X} = \{\mathbf{x}_i\}$. Dictionary update and sparse coding stages are alternated, with the dictionary learning stage optimizing each atom sequentially. [Engan et al., 1999] also developed the Method of Optimal Directions (MOD), which is simply a least-squares method of solving for the entire dictionary directly. However, this method does not consider the sparsity structure of the coefficients, and therefore is not sparsity-promoting. Further, the K-SVD algorithm has been extremely successful and popular for its speed of convergence and excellent results, and has been the algorithm of choice for dictionary learning applications. [Rubinstein et al., 2010a] provides a survey of various approaches for training a dictionary from a given set of training signals.

Dictionary learning been used for image denoising in [Elad and Aharon, 2006], and [Mairal et al., 2008a] perform denoising, demosaicking and image inpainting using dictionary learning. Also, in [Mairal et al., 2009] the authors use dictionary learning and sparse coding in a non-local-means-type framework for image denoising.

[Mairal et al., 2007] learn dictionaries at multiple scales on a quad-tree decomposition, aiming for the multi-scale performance characteristic of wavelet dictionaries. [Rubinstein et al., 2010b] learn sparse dictionaries as well as sparse coefficients, denoted as *double sparsity.* [Bar and Sapiro, 2010] use dictionary learning in a hierarchical architecture to make the process invariant to rigid transformations. [Duarte-Carvajalino and Sapiro, 2009] use dictionary learning in a compressive sensing framework, where they learn both the dictionary as well as the sensing matrix simultaneously, given a training set of signals.

[Mairal et al., 2009, Mairal et al., 2010] present an online algorithm for dictionary learning which is highly suited for large datasets, extremely fast, and enjoys theoretical performance guarantees. [Mairal et al., 2008, Mairal et al., 2008b] augment the dictionary learning optimization with a discriminative term for performing classification with multiple dictionaries. [Sprechmann and Sapiro, 2010] use dictionary learning in an EM framework for unsupervised clustering. [Ramirez et al., 2010] apply dictionary learning to both clustering and classification applications. In [Castrodad and Sapiro, 2011], the authors learn dictionaries at two levels, one on the image features and the next on the consolidated sparse coefficients from sparse coding over the first dictionary. With this approach, they demonstrate state-of-the-art results for action recognition on various datasets.

This list described above is not claimed to be a comprehensive review of sparsity-related algorithms, but a representative sample which is relevant to the domain of computer vision and image processing. They show the power of sparse models in solving computer vision problems, and motivate the need to extend these powerful models to other classes of features, such as covariance matrices.

2.4 Sparse Models for Positive Definite Matrices

The vector sparse modeling tools of sparse coding and dictionary learning have been helpful not only for learning compact representations, but also for developing interpretable models and extracting semantic information from the signals of interest. The extension of these tools for positive definite matrices will therefore greatly benefit not only computer vision applications, but also many of the other domains where the data points are positive definite matrices.

The literature on region covariances shows the many different applications of these descriptors, and the variety of algorithms used. Although there has been extensive work in the machine learning and statistics literature on low-rank modeling of matrices and sparse inverse covariance estimation, there has been very little work attempting to extend sparse linear regression to positive definite matrices.

In [Guo et al., 2010a], the authors take the covariance descriptors to the tangent space at the identity, resulting in symmetric matrices which are then vectorized. They then perform vector sparse coding in this Euclidean space. This is still an approximation, and the decomposition is not linear in the covariances themselves, but linear in the logarithmic map. In fact, their decomposition can be viewed in a way as a sparse product of dictionary atoms, rather than a sparse linear combination. However, the excellent results demonstrated in this work motivates the development of sparse coding and dictionary learning techniques which can operate directly on the manifold of region covariances.

[Pfander et al., 2008] decompose a general matrix as a sparse linear combination of a dictionary of matrices by multiplying all the involved matrices on a known vector reducing the matrix problem to a known vector problem with well-established guarantees. [Wang et al., 2010] present the *Common Component Analysis* problem, where the authors learn a common low-dimensional subspace for a set of high-dimensional covariance matrices.

In a similar approach, [Sra and Cherian, 2011] learn a generalized dictionary of rank-1 positive semidefinite atoms to sparsely represent covariance descriptors. However, the authors in the above two approaches use the Frobenius norm as the error metric.

In this thesis, we present a novel sparse coding approach that uses a distortion function more appropriate for positive definite matrices - motivated from the Wishart probability distribution and respecting the Riemannian manifold geometry. A sparse modeling approach to learn dictionaries of positive definite matrices from the training data is proposed, and extensions to rank-1 dictionary atoms and discriminative dictionary learning are also presented. We explore the effects of various quantities involved in the sparse coding and dictionary learning framework, and provide efficient implementations for the algorithms in this thesis. From the community, [Wang et al., 2012] have already used our proposed sparse coding and dictionary learning approaches to model the covariance matrix Σ in a graphical model and apply this model to classify images from different scene categories. The usefulness of the sparse covariance models described in this thesis to the computer vision and image processing community is demonstrated by the experiments presented in this thesis.

Chapter 3

Positive Definite Sparse Coding

The first step is the development of sparse models for positive definite matrices is the design of sparse linear regression techniques. We denote this as *tensor sparse coding*[1] to contrast this with the usual methods of vector sparse coding.

In this chapter, we formulate the tensor sparse coding problem with an appropriate distortion measure for positive semidefinite matrices. We then show that our formulation falls under a well-known class of convex optimization problems. The effects of various parameters such as the dictionary size, sparsity regularizer, data dimension, and normalization scheme are explored, giving an in-depth understanding of the possibilities of tensor sparse coding. An empirical phase transition diagram showing the coefficient recovery properties of this formulation, under a compressive sensing viewpoint.

Next, we compare the performance of our method with vector sparse coding on a synthetic dataset, showing the necessity of a direct tensor approach to sparse coding positive definite matrices. A relation between our sparse coding distortion measure and the Riemannian geodesic distance is derived, showing that our proposed method conforms to the manifold structure. An efficient algorithm for solving the sparse coding problem is also

[1] The name *tensor* is inspired from diffusion tensor imaging, where the positive definite matrix at each voxel is referred to as the *diffusion tensor*.

presented, which gives 2 orders of magnitude speed-up over off-the-shelf interior point methods. Experiments on real-world computer vision applications show the practical usefulness of positive definite sparse coding.

3.1 The Tensor Sparse Coding Problem

We begin with a known dictionary consisting of K $n \times n$ positive definite matrices $\mathcal{A} = \{A_i\}_{i=1}^K$, where each $A_i \in \mathbb{S}_{++}^n$ is referred to as a dictionary atom. Given a signal $S \in \mathbb{S}_{++}^n$, our goal is to represent S as a linear combination of the dictionary atoms, *i.e.*,

$$S = x_1 A_1 + x_2 A_2 + \ldots + x_K A_K = \sum_{i=1}^{K} x_i A_i, \tag{3.1}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_K)^T$ is the coefficient vector.

With a slight abuse of notation, we will henceforth represent the sum $\sum_{i=1}^K x_i A_i$ as $\mathcal{A}\mathbf{x}$ for the sake of convenience[2] .

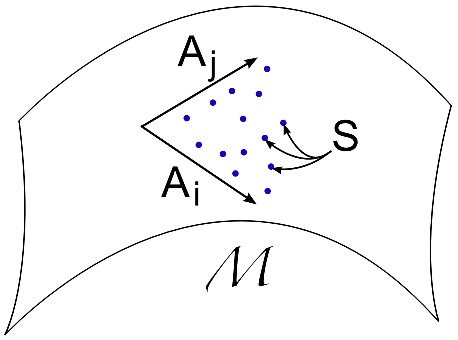


Figure 3.1: Data points S on the manifold of positive definite matrices are to be represented by a linear combination of atoms A_i from the dictionary $\mathcal{A}$.

Since only a non-negative linear combination of positive definite matrices is guaranteed to yield a positive definite matrix, we impose the constraint $\mathbf{x} \geq 0$ on the coefficient vector. However, we will also explore the effect of removing this constraint in later sections.

It is to be noted that the given matrix S need not always be exactly representable as a sparse non-negative linear combination of the dictionary atoms. In other words, S need

[2] This can be distinguished from the regular $A\mathbf{x}$ matrix-vector multiplication through the calligraphic notation of $\mathcal{A}$.

not be exactly sparse in the space of the dictionary $\mathcal{A}$. Hence, we will try to find the best approximation $\hat{S} = \mathcal{A}\mathbf{x}$ to S, by minimizing the residual approximation error.

$$S \approx \hat{S} = \mathcal{A}\mathbf{x}^*, \text{ where } \mathbf{x}^* = \arg\min_{\mathbf{x}} \; d\left(\mathcal{A}\mathbf{x}, S\right), \tag{3.2}$$

and $d(\cdot, \cdot)$ is an appropriate distortion measure over positive definite matrices.

Since we are reconstructing a positive definite signal S, we also require the approximation $\hat{S}$ to be positive definite,

$$\hat{S} \succeq 0 \Longrightarrow x_1 A_1 + x_2 A_2 + \ldots + x_K A_K \succeq 0. \tag{3.3}$$

Although this would be ensured by construction due to the non-negativity of $\mathbf{x}$ and the strictly positive definite dictionary atoms, we nonetheless retain this constraint explicitly for reasons which will become clear shortly.

We further require that the representation be sparse, *i.e.*, S is to be represented by a sparse linear combination of the dictionary atoms. To this effect, we impose a constraint on the ℓ_0 "pseudo-norm" of $\mathbf{x}$,

$$\|\mathbf{x}\|_0 \leq T, \tag{3.4}$$

where T is a pre-defined parameter, denoting the maximum number of non-zero elements of x.

Next we need to select the distortion measure in Equation (3.2). While the Riemannian geodesic distance (2.2) would be our first choice - however it is a non-convex function (consider $|\log x|$) and therefore difficult to optimize directly. Hence we search for another loss function to optimize. The LogDet divergence, as we will elaborate next, is a well-suited distortion measure, not only due to its significant relation with Wishart and Gaussian distributions, but also because it results in a well-known and tractable convex optimization problem.

3.2 The LogDet Divergence

The LogDet divergence [Kulis et al., 2006] $D_{\text{ld}}\left(X, Y\right)$ is a Bregman divergence [Bregman, 1967] between two matrices $X \in \mathbb{S}_{+}^{n}$ and $Y \in \mathbb{S}_{++}^{n}$, and is given by,

$$D_{\text{ld}}(X, Y) = \operatorname{tr}\left(XY^{-1}\right) - \log\det\left(XY^{-1}\right) - n. \tag{3.5}$$

It is asymmetric (and therefore, a divergence) $D_{\text{ld}}\left(X, Y\right) \neq D_{\text{ld}}\left(Y, X\right)$, and is convex only in the first argument. It is also known as Stein's loss in covariance estimation in statistics, or the Burg matrix divergence (a matrix generalization of the Burg divergence).

The LogDet divergence is equal to twice the *Kullback-Leibler* divergence (K-L divergence) between two multivariate Gaussians with equal mean [Davis et al., 2007]. Consider:

$$P_x = \mathcal{N}\left(\mu_x, \Sigma_x\right), \tag{3.6}$$

$$P_y = \mathcal{N}\left(\mu_y, \Sigma_y\right), \tag{3.7}$$

where $\mu_x, \mu_y \in \mathbb{R}^n$ and $\Sigma_x, \Sigma_y \in \mathbb{S}_{++}^{n}$. The K-L divergence between P_x and P_y is given by

$$D_{\text{KL}}\left(P_x \| P_y\right) = \frac{1}{2}\Big(\operatorname{tr}\left(\Sigma_y^{-1}\Sigma_x\right) - \log\det\left(\Sigma_y^{-1}\Sigma_x\right) + \left(\mu_x - \mu_y\right)^T \Sigma_y^{-1}\left(\mu_x - \mu_y\right) - n\Big). \tag{3.8}$$

When $\mu_x = \mu_y$,

$$D_{\text{KL}}\left(P_x \| P_y\right) = \frac{1}{2}\left(\operatorname{tr}\left(\Sigma_y^{-1}\Sigma_x\right) - \log\det\left(\Sigma_y^{-1}\Sigma_x\right) - n\right), \tag{3.9}$$

$$\therefore D_{\text{KL}}\left(P_x \| P_y\right) = \frac{1}{2} D_{\text{ld}}\left(\Sigma_x, \Sigma_y\right). \tag{3.10}$$

According to [Banerjee et al., 2005], there exists a bijection between regular exponential families and a large class of Bregman divergences known as regular Bregman divergences. For example, the squared-error loss function which is minimized in vector sparse coding methods comes from the squared Euclidean distance, which is the Bregman divergence corresponding to the multivariate Gaussian distribution. Thus, the minimization of a squared error objective function corresponds to the assumption of Gaussian noise. The Wishart distribution [Wishart, 1928], which is a distribution over $n \times n$ positive definite matrices, with positive definite parameter matrix $\Theta \in \mathbb{S}^n_{++}$ and degrees of freedom $p \geq n$, is given by

$$\Pr(X|\Theta, p) = \frac{|X|^{(p-n-1)/2} \exp\left(-\frac{1}{2}\mathrm{tr}\left(\Theta^{-1}X\right)\right)}{2^{pn/2}|\Theta|^{p/2}\Gamma_n(p/2)}, \tag{3.11}$$

where $|\cdot|$ is the determinant. The LogDet divergence $D_{\mathrm{ld}}(X, \Theta)$ is the Bregman divergence corresponding to the Wishart distribution $\Pr(X|\Theta, p)$ [Wang et al., 2009].

The Wishart distribution is also a conjugate prior for the inverse sample covariance matrix (or precision matrix) of a multivariate Gaussian distribution. Correspondingly, the inverse Wishart distribution is the conjugate prior for the sample covariance matrix. [Gelman et al., 2003]. Since

$$D_{\mathrm{ld}}(X, Y) = D_{\mathrm{ld}}(Y^{-1}, X^{-1}), \tag{3.12}$$

the Bregman divergence for the inverse Wishart distribution $\Pr(X^{-1}|\Theta^{-1}, p)$ is $D_{\mathrm{ld}}(\Theta^{-1}, X^{-1})$. Here Θ^{-1} refers to the true covariance of the multivariate Gaussian distribution and X^{-1} the sample covariance matrix.

In the sparse coding framework, if $\hat{S}$ is the true covariance, and S is the sample covariance signal[3], the goal is to estimate the true covariance as a sparse linear combination of certain basis atoms. Therefore, the Logdet divergence $D_{\mathrm{ld}}(\hat{S}, S)$ appears to be a suitable candidate as the objective function for the sparse coding formulation.

Note that the LogDet divergence is also affine-invariant like the geodesic distance, in terms of its arguments:

$$D_{\mathrm{ld}}(XAX^T, XBX^T) = D_{\mathrm{ld}}(A, B) \quad \text{for any invertible } X. \tag{3.13}$$

In the later sections we will also show a further relation between the Riemannian geodesic distance (2.2) and the LogDet divergence (3.5).

[3] The notation is, unfortunately, counter-intuitive, since we usually denote the true signal X and the estimate $\hat{X}$.

3.3 The Tensor Sparse Coding Formulation

Motivated by the aforementioned reasons, the optimization problem is defined as minimizing the LogDet divergence $D_{\text{ld}}(\hat{S}, S)$ between the approximation $\hat{S}$ and the given matrix S.

$$D_{\text{ld}}(\hat{S}, S) = \operatorname{tr}\left(S^{-1}\mathcal{A}\mathbf{x}\right) - \log\det\left(S^{-1}\mathcal{A}\mathbf{x}\right) - n. \tag{3.14}$$

In order to reduce the problem to a canonical form, and to improve numerical stability, we apply the invariant property of the trace and the log det under similarity transformations. The objective function is unaffected by the similarity map $X \mapsto S^{1/2}XS^{-1/2}$, where X is the argument of the trace or log det.

$$D_{\text{ld}}(\hat{S}, S) = \operatorname{tr}\left(S^{-1/2}\left(\mathcal{A}\mathbf{x}\right)S^{-1/2}\right) - \log\det\left(S^{-1/2}\left(\mathcal{A}\mathbf{x}\right)S^{-1/2}\right) - n \tag{3.15}$$

$$= \operatorname{tr}\left(\hat{\mathcal{A}}\mathbf{x}\right) - \log\det\left(\hat{\mathcal{A}}\mathbf{x}\right) - n, \tag{3.16}$$

where $\hat{\mathcal{A}} == \{\hat{A}_i\}_{i=1}^{K}$, and $\hat{A}_i = S^{-1/2}A_iS^{-1/2}$. Exploiting the linearity of the trace, setting $\mathbf{c} : c_i = \operatorname{tr}\hat{A}_i$, and discarding the constant n,

$$f(\mathbf{x}) = D_{\text{ld}}(\hat{S}, S) = \mathbf{c}^T\mathbf{x} - \log\det\left(\hat{\mathcal{A}}\mathbf{x}\right). \tag{3.17}$$

In order to learn the dictionary $\mathcal{A}$, it becomes necessary to impose a constraint that the residual $E = S - \hat{S}$ be positive semidefinite, and not indefinite. The minimum eigenvalue of the residual $\lambda_{\min}\left(S - \hat{S}\right)$ should therefore be non-negative and as close to zero as possible. Later, we will compare the effects of relaxing this constraint in the experiments.

$$\hat{S} = \mathcal{A}\mathbf{x} \preceq S \quad \text{or} \quad \hat{\mathcal{A}}\mathbf{x} \preceq I_n, \tag{3.18}$$

where I_n is the $n \times n$ identity matrix. Combining with Equation (3.3), we get

$$0 \preceq \hat{\mathcal{A}}\mathbf{x} \preceq I_n. \tag{3.19}$$

The ℓ_0 sparsity constraint in Equation (3.4) is non-convex and and therefore we replace this with its nearest *convex relaxation* - the ℓ_1 norm of $\mathbf{x}$. Under certain assumptions

[Tropp, 2006], minimizing the ℓ_1 penalty has been proven to yield equivalent results as minimizing $\|\mathbf{x}\|_0$ for sparse vector decompositions. Hence it is appealing to perform the same relaxation here as well.

Combining all the above constraints with the objective function we wish to minimize, we have the following optimization problem:

$$\min_{\mathbf{x}\geq 0} \qquad \mathbf{c}^T\mathbf{x} - \log\det\left(\hat{\mathcal{A}}\mathbf{x}\right) + \lambda\left\|\mathbf{x}\right\|_1 \tag{3.20}$$

$$\text{s.t.} \qquad 0 \preceq \hat{\mathcal{A}}\mathbf{x} \preceq I_n, \tag{3.21}$$

where $\lambda \geq 0$ is a parameter which represents a trade–off between a sparser representation and a more accurate reconstruction. Since the x_i's are non–negative, the ℓ_1 norm simply becomes the sum of the components of $\mathbf{x}$, *i.e.*,

$$\left\|\mathbf{x}\right\|_1 = \sum_{i=1}^{K} x_i, \tag{3.22}$$

yielding the optimization problem:

$$\min_{\mathbf{x}\geq 0} \qquad \hat{\mathbf{c}}^T\mathbf{x} - \log\det\left(\hat{\mathcal{A}}\mathbf{x}\right) \tag{3.23a}$$

$$\text{s.t.} \qquad 0 \preceq \hat{\mathcal{A}}\mathbf{x} \preceq I_n, \tag{3.23b}$$

where the sparse regularization is absorbed into the first linear term, with $\hat{c}_i = c_i + \lambda$.

Concurrent with other vector sparse coding techniques, we may express this optimization problem in an alternate form which puts a hard constraint on the ℓ_1 norm of $\mathbf{x}$ instead of a penalty term $\lambda\left\|\mathbf{x}\right\|_1$ in the objective function.

$$\min_{\mathbf{x}\geq 0} \qquad \mathbf{c}^T\mathbf{x} - \log\det\left(\hat{\mathcal{A}}\mathbf{x}\right) \tag{3.24a}$$

$$\text{s.t.} \qquad \sum_{i=1}^{K} x_i \leq T \tag{3.24b}$$

$$0 \preceq \hat{\mathcal{A}}\mathbf{x} \preceq I_n, \tag{3.24c}$$

We denote the optimization problems defined by (3.23) and (3.24) as Type I (ℓ_1-*regularized*) and Type II (ℓ_1-*constrained*) respectively.

Tensor Sparse Coding: Type-I (ℓ_1-*regularized*)

Given $S \succ 0$, $\mathcal{A} = \{A_i \mid A_i \succeq 0\}_{i=1}^{K}$:

$$\min_{\mathbf{x}} \quad \sum_{i=1}^{K} x_i \operatorname{tr}\left(A_i S^{-1}\right) - \log\det\left(\sum_{i=1}^{K} x_i A_i S^{-1}\right) + \lambda \left\|\mathbf{x}\right\|_1$$

$$\text{s.t.} \quad \mathbf{x} \geq 0, \qquad 0 \preceq \textstyle\sum_{i=1}^{K} x_i A_i \preceq S$$

Tensor Sparse Coding: Type-II (ℓ_1-*constrained*)

Given $S \succ 0$, $\mathcal{A} = \{A_i \mid A_i \succeq 0\}_{i=1}^{K}$:

$$\min_{\mathbf{x}} \quad \sum_{i=1}^{K} x_i \operatorname{tr}\left(A_i S^{-1}\right) - \log\det\left(\sum_{i=1}^{K} x_i A_i S^{-1}\right)$$

$$\text{s.t.} \quad \mathbf{x} \geq 0, \qquad \left\|\mathbf{x}\right\|_1 \leq T, \qquad 0 \preceq \textstyle\sum_{i=1}^{K} x_i A_i \preceq S$$

3.4 The MAXDET problem

The above formulations of tensor sparse coding fall under a general class of optimization problems known as determinant maximization problems [Vandenberghe et al., 1998], (MAXDET), of which semi-definite programming (SDP) and linear programming (LP) are special cases. The MAXDET problem is defined as [Vandenberghe et al., 1998]:

$$\min_{\mathbf{x}} \quad \mathbf{c}^T\mathbf{x} + \log\det G(\mathbf{x})^{-1} \tag{3.25a}$$

$$\text{s.t.} \quad G(\mathbf{x}) \triangleq G_0 + x_1G_1 + \ldots + x_KG_K \succ 0 \tag{3.25b}$$

$$F(\mathbf{x}) \triangleq F_0 + x_1F_1 + \ldots + x_KF_K \succeq 0, \tag{3.25c}$$

where $\mathbf{x} \in \mathbb{R}^K$, $G_i \in \mathbb{S}^n$ and $F_i \in \mathbb{S}^N$. The MAXDET problem is convex and efficient interior point (IP) methods exist for solving them.

Note that the $G(\mathbf{x})$ inside the $\log\det$ term also explicitly appears as a constraint in the standard form of the MAXDET problem, leading to our inclusion of the same in our formulation.

3.4.1 Type I: ℓ_1-regularized Sparse Coding

Comparing to the optimization problem Type I in (3.23), we have

$$c_i = \text{tr}\hat{A}_i + \lambda, \quad \text{for } i = 1, \ldots, K \tag{3.26}$$

$$G(\mathbf{x}) = \sum_{i=1}^{K} x_i\hat{A}_i \succ 0 \tag{3.27}$$

$$F(\mathbf{x}) = \left[\begin{array}{c|c} \text{diag}(\mathbf{x}) & 0 \\ \hline 0 & I_n - \sum_{i=1}^{K} x_i\hat{A}_i \end{array}\right] \succeq 0, \tag{3.28}$$

with $N = K + n$. The corresponding component matrices are given by

$$\begin{aligned} &G_0 = 0, && G_i = \hat{A}_i, && \text{for } i = 1, \ldots, K, \\ &F_0 = \left[\begin{array}{c|c} 0 & 0 \\ \hline 0 & I_n \end{array}\right], && F_i = \left[\begin{array}{c|c} \text{diag}(\mathbf{e}_i) & 0 \\ \hline 0 & -\hat{A}_i \end{array}\right], && \text{for } i = 1, \ldots, K, \end{aligned} \tag{3.29}$$

where $\mathbf{e}_i$, $i = 1, \ldots, K$ are the canonical basis vectors in $\mathbb{R}^K$.

3.4.2 Type II: ℓ_1-constrained Sparse Coding

Comparing to the optimization problem Type II in (3.24), we have

$$c_i = \text{tr}\hat{A}_i, \quad \text{for } i = 1, \ldots, K \tag{3.30}$$

$$G(\mathbf{x}) = \sum_{i=1}^{K} x_i \hat{A}_i \succ 0 \tag{3.31}$$

$$F(\mathbf{x}) = \left[\begin{array}{c|c|c} \text{diag}(\mathbf{x}) & 0 & 0 \\ \hline 0 & T - \sum_{i=1}^{K} x_i & 0 \\ \hline 0 & 0 & I_n - \sum_{i=1}^{K} x_i \hat{A}_i \end{array}\right] \succeq 0, \tag{3.32}$$

with $N = K + n$. The corresponding component matrices are given by

$$\begin{aligned} &G_0 = 0, && G_i = \hat{A}_i, && \text{for } i = 1, \ldots, K, \\ &F_0 = \left[\begin{array}{c|c|c} 0 & 0 & 0 \\ \hline 0 & T & 0 \\ \hline 0 & 0 & I_n \end{array}\right], && F_i = \left[\begin{array}{c|c|c} \text{diag}(\mathbf{e}_i) & 0 & 0 \\ \hline 0 & -1 & 0 \\ \hline 0 & 0 & -\hat{A}_i \end{array}\right], && \text{for } i = 1, \ldots, K, \end{aligned} \tag{3.33}$$

where $\mathbf{e}_i$, $i = 1, \ldots, K$ are the canonical basis vectors in $\mathbb{R}^K$.

Thus, we have formulated two variations of our tensor sparse coding problem (ℓ_1-regularized and ℓ_1-constrained), both of which are convex and have been expressed in the standard MAXDET form. The feasible set consists of the region of intersection of two positive semidefinite cones (see Figure 3.2), one centered at the origin O, and the other

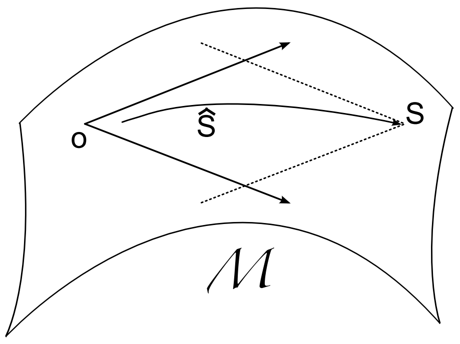


Figure 3.2: The convex feasible set for reconstruction $\hat{S} : 0 \preceq \hat{S} \preceq S$

- an inverted cone centered at S. The approximation $\hat{S}$ lies in the strict interior of this closed convex set. The $-\log\det$ term in the objective serves two purposes - a) it pushes the approximation $\hat{S}$ toward S, motivating a better approximation, and b) it serves as the log-barrier function preventing the reconstruction from becoming indefinite. The linear term serves as a weighted regularizer on the coefficients $\mathbf{x}$.

3.4.3 Semidefinite Signals

It is important to note here that the signal S must be strictly positive definite in our sparse coding problem, but the atoms in the dictionary can be semidefinite (in fact, even rank-1, which will be handled in a specialized manner in Chapter 5). In order to sparse code positive semidefinite signals $S \in \mathbf{S}^n_+$, we suggest the following procedure:

1. Add a small multiple $\delta > 0$ times the $n \times n$ identity matrix I_n to the signal: $\tilde{S} \leftarrow S + \delta I_n$.

2. Concatenate the dictionary with the identity matrix: $\tilde{\mathcal{A}} \leftarrow [\ \mathcal{A}\ \ I_n\ ]$.

3. Sparse code $\tilde{S}$ over $\tilde{\mathcal{A}}$ to get $\tilde{\mathbf{x}} = [\ \mathbf{x}\ \ x_I\ ]$, where x_I corresponds to the coefficient of I_n.

4. Ignore x_I and compute the reconstruction $\hat{S} = \mathcal{A}\mathbf{x}$.

3.5 Effect of Sparsity Constraints

Our first set of experiments were run on a synthetic data set, comprised of covariance matrices. We start with a randomly generated $n \times n$ covariance matrix C ($n = 5$) and generate sets of samples from a multivariate Gaussian distribution $\mathcal{N}(0, C)$. There are $O(n^2)$ samples per set, from which we compute the sample covariance for each of these sample sets. These covariance matrices forms our data set. We select $K = 60$ of these matrices to form our dictionary $\mathcal{A} = \{A_i\}_{i=1}^{K}$. The sample point S to be sparse-coded is also generated in this manner. The covariance matrix of a multivariate Gaussian distribution follows an inverse Wishart distribution, and therefore our optimization problem is well-suited to this model. The quantities we consider to represent the performance of the reconstruction are the LogDet divergence $D_{\mathrm{ld}}(\hat{S}, S)$, the geodesic distance $D_{\mathrm{geo}}(\hat{S}, S)$, the ℓ_1 norm of the optimal coefficient vector $\|\mathbf{x}^*\|_1$ and the minimum eigenvalue of the residual $\lambda_{\min}(S - \hat{S})$.

Figure 3.3 shows the effect of varying λ on the quality of reconstruction, under the ℓ_1-regularized problem. The geodesic distance can be seen to vary in a smooth and similar fashion to the LogDet divergence, reaffirming our choice of objective function. We also show the actual solution vector $\mathbf{x}^*$ for $\lambda = 0$, where it can be seen that even the unconstrained case results in a sparse solution vector. This is due to the non-negativity constraint on the coefficient vector, and it is widely noted in the vector-domain that non-negative decompositions result in sparsity, under certain conditions [Donoho and Tanner, 2005, Donoho and Stodden, 2004, Lee and Seung, 2000].

Figure 3.4 shows a similar set of plots for the ℓ_1-constrained problem. Instead of plotting $\|\mathbf{x}\|_1$, which is anyways constrained to be less than or equal to T, we plot $\|\mathbf{x}\|_0$ vs. T. We set a threshold of 10^{-8} for the coefficients (Note the staircase-like graph of $\|\mathbf{x}\|_0$).

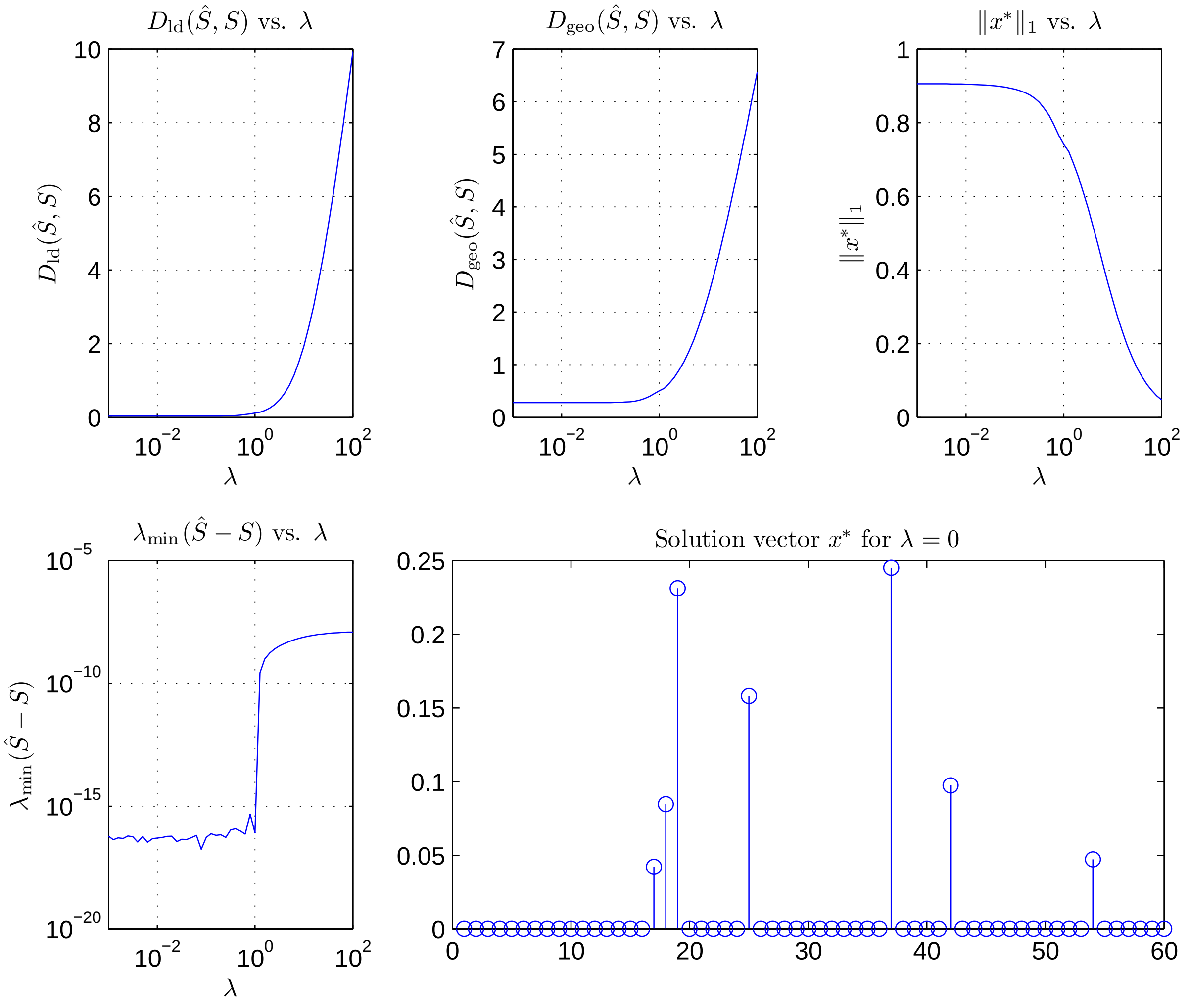


Figure 3.3: Effect of sparsity constraints, shown for $n = 5$, $K = 60$. We show $D_{\mathrm{ld}}(\hat{S}, S)$, $D_{\mathrm{geo}}(\hat{S}, S)$, $\|\mathbf{x}^*\|_1$, as well as $\lambda_{\min}(S - \hat{S})$ plotted in logarithmic scale. The λ values are varied logarithmically. The solution vector $\mathbf{x}^*$ in the unconstrained case is also shown on the right, and is observed to be sparse even without explicitly enforcing any sparsity.

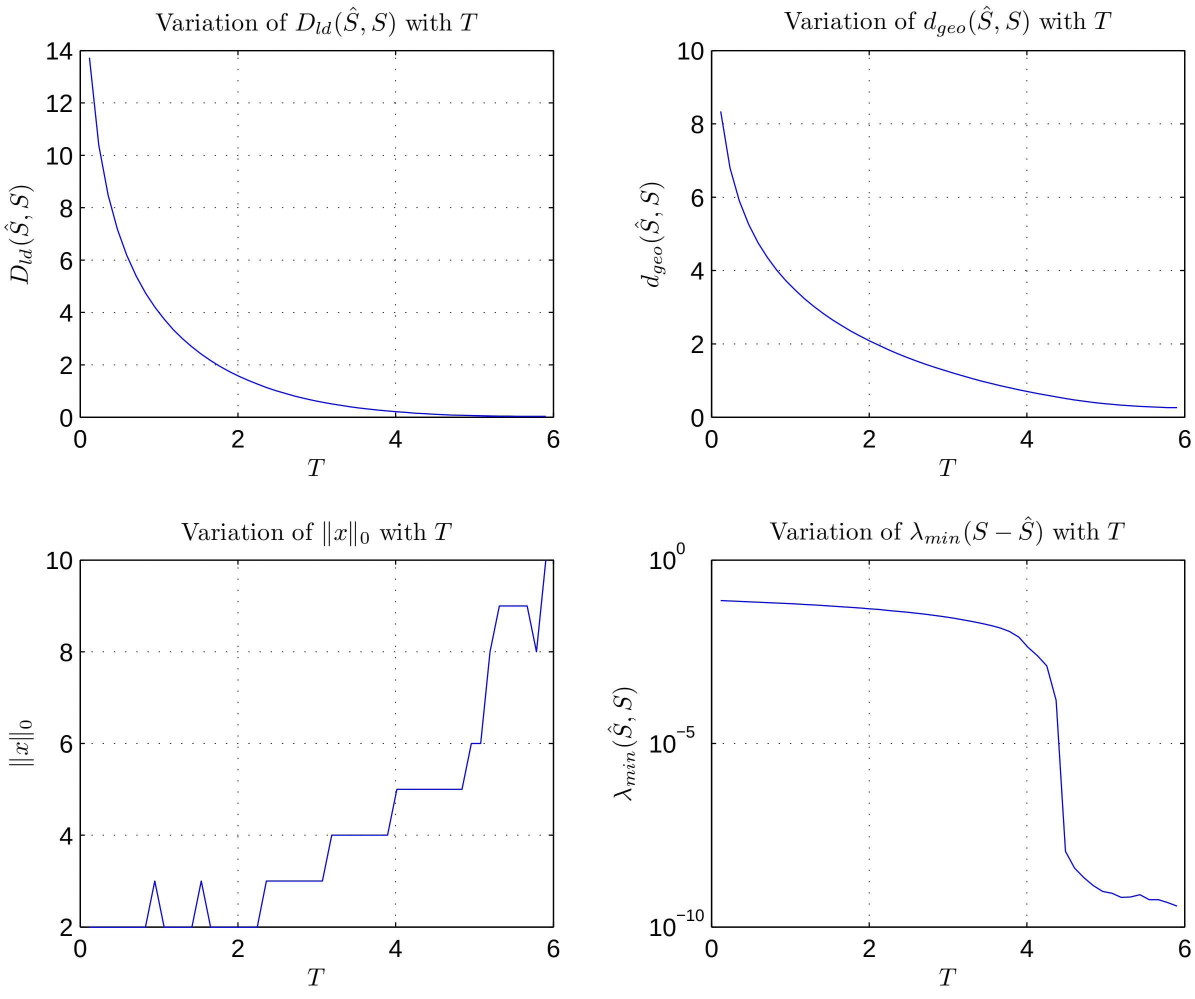


Figure 3.4: Plot of the various quantities vs. T for $n = 5$, $K = 60$. We show $D_{\mathrm{ld}}(\hat{S}, S)$, $D_{\mathrm{geo}}(\hat{S}, S)$, $\|\mathbf{x}^*\|_1$, as well as $\lambda_{min}(S - \hat{S})$ plotted in logarithmic scale. The T values are varied linearly, from $T_{\min}$ to $T_{\max}$, in steps of T_{step}. The solution vector $\mathbf{x}^*$ in the unconstrained case $T = \infty$ is exactly the same as shown in Figure 3.3.

3.6 Effect of Atom Normalization

In vector sparse coding and dictionary learning, when trying to approximate a signal $\mathbf{x}$ by a dictionary D and a coefficient vector α, the dictionary atoms are usually normalized to have unit length, since the decomposition $\mathbf{x} = D\alpha$ can be determined only up to a scaling factor. We have this same issue in the tensor sparse coding problem as well, and hence we need a standard method of normalization throughout this work. Different ways to normalize the dictionary atoms were tested:

- normalization by spectral norm, $\|A_i\|_2 = 1$.
- normalization by Frobenius norm, $\|A_i\|_F = 1$.
- normalization by trace, $\operatorname{tr}(A_i) = 1$.

Since all matrix norms are equivalent [Golub and Loan, 1996], we expect to see only a proportional change in any of the output characteristics between the different normalization schemes. Figure 3.5 shows that this is indeed the case. Throughout the rest of this work, we adhere to normalization by Frobenius norm, due to the relation to the atom coherence definition in the next section.

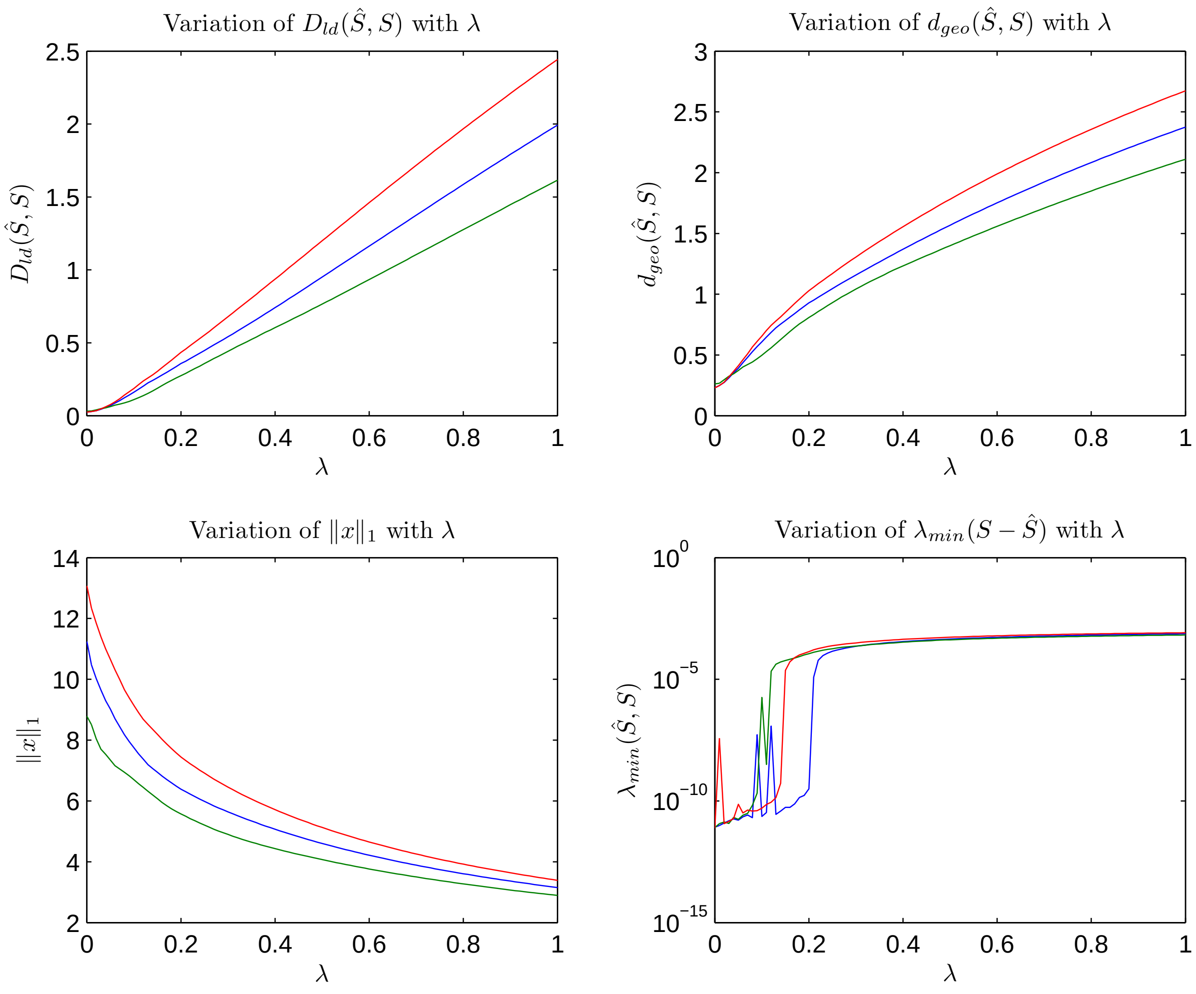


Figure 3.5: Plot of the various quantities vs. λ for $n = 5$, $k = 60$, showing the effect of different normalizations - (i) blue line - 2-norm, (ii) green line - Frobenius norm, and (iii) red line - trace. We show the LogDet divergence $D_{\text{ld}}(\hat{S}, S)$, the geodesic distance $D_{\text{geo}}(\hat{S}, S)$, the ℓ_1 norm of $\mathbf{x}$ $\|\mathbf{x}\|_1$, and the minimum eigenvalue of the residual $\lambda_{\min}(S - \hat{S})$ plotted in logarithmic scale.

3.7 Atom Coherence

Before we proceed any further, it is important to define a fundamental property of the dictionary. We extend the coherence property for vector dictionaries to dictionaries of positive semi-definite atoms. The inner product in a positive definite matrix space is given by $\langle A_i, A_j \rangle = \operatorname{tr}(A_i A_j)$.

Definition 1. *The **coherence** between two symmetric positive (semi-)definite dictionary atoms A_i and A_j is given by*

$$\mu(A_i, A_j) = \langle A_i, A_j \rangle = \operatorname{tr}(A_i A_j), \tag{3.34}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product in matrix space.

The triangle inequality gives

$$\operatorname{tr}(A_i A_j) \quad \leq \quad \|A_i\|_F \|A_j\|_F. \tag{3.35}$$

Therefore if the atoms are normalized to unit Frobenius norm as mentioned in the previous section, we have the following bounds on the coherence measure:

$$0 \quad \leq \quad \mu(A_i, A_j) \quad \leq \quad 1, \qquad \text{for } \; A_i, A_j \in \mathbb{S}^n_+, \quad \|A_i\|_F = \|A_j\|_F = 1. \tag{3.36}$$

For non-trivial A_i and A_j:

- $\mu(A_i, A_j) = 0$ if and only if they are low-rank (semi-definite) and their eigenspaces are disjoint.

- $\mu(A_i, A_j) = 1$ if and only if $A_i = A_j$.

3.8 Completeness and Coefficient Recovery

We define the terms *undercomplete*, *complete* and *overcomplete* with respect to positive semidefinite atoms. Since there are $M = n(n+1)/2$ variables in an $n \times n$ positive semidefinite matrix, this denotes the ambient dimension in which these matrices are present. There can be at most M linearly independent $n \times n$ positive semidefinite matrices. Although there are further constraints on the eigenvalues, we define completeness with reference to this quantity.

Definition 2. *A* ***complete*** *dictionary of positive semidefinite matrices has* $K = M = n(n+1)/2$ *atoms. A dictionary with* $K < M$ *is denoted as* ***undercomplete*** *and that with* $K > M$ *is* ***overcomplete****.*

The MAXDET formulation (3.25) guarantees a unique optimal solution as long as the dictionary atoms A_i, $i = 1, \ldots, K$ are linearly independent [Vandenberghe et al., 1998], regardless of the sparsity of the coefficient vector $\mathbf{x} \in \mathbf{R}^K$. Therefore, so long as $K \leq M$, the MAXDET optimal solution is unique.

However, we are more interested in the overcomplete case ($K > M$) and the solution vector $\mathbf{x}^*$ is sparse, with only k non-zero elements in K dimensions (k-sparse). From the signal measurement and recovery viewpoint of compressed sensing, we may consider M as the number of measurements.

[Donoho and Tanner, 2005, Donoho and Tanner, 2009, Donoho and Tanner, 2010] use *phase transition diagrams* to show the conditions of exact recovery of sparse signals, in terms of the sparsity fraction $\rho = k/M$ and the undersampling ratio $\delta = M/K$. We show a similar empirical phase transition diagram in Figure 3.6 through the following described experiment.

A synthetic dictionary $\mathcal{A}$ of K $n \times n$ positive definite atoms is generated, and a random sparse vector $\mathbf{x}^* \in \mathbf{R}_+^K$ is synthesized, with varying sparsity $k = 1, \ldots, M$. A signal is constructed $S = \mathcal{A}\mathbf{x}^*$, and sparse-coded over $\mathcal{A}$ to obtained the recovered estimate $\hat{\mathbf{x}}$. The sparse vector $\mathbf{x}^*$ is said to be correctly recovered if the relative error is less than or equal to 1%:

$$\frac{\|\hat{\mathbf{x}} - \mathbf{x}^*\|_2}{\|\mathbf{x}^*\|_2} \leq 0.01$$

The experiment was performed for $n \in \{3, 4, 5, 6\}$ and averaged over $N = 500$ trials each, and the fraction of correct recovery is shown in Figure 3.6. We see a similar trend as shown in [Donoho and Tanner, 2009], where even for extremely overcomplete dictionaries, when the signal is sparse enough, it is correctly recovered. The overlaid line indicates the boundary of $\sim 50\%$ correct reconstruction.

Throughout this thesis, we constrain $\mathbf{x}$ to be non-negative, since only non-negative linear combinations of positive definite matrices are guaranteed to be positive definite. However, the MAXDET problem [Vandenberghe et al., 1998] itself does not have such constraints on the coefficients. The semidefinite constraints are sufficient to keep the reconstruction $\hat{S}$ positive definite. The guarantee of unique recovery when $K \leq M$ still holds for general $\mathbf{x} \in \mathbf{R}^K$ for any sparsity k.

To test whether unique recovery of general $\mathbf{x}$ is still possible with overcomplete dictionaries $K > M$, we relax the non-negative constraints on the coefficients $\mathbf{x}$. The ground truth $\mathbf{x}^* \in \mathbf{R}^K$ is generated by ensuring that the reconstruction $S = \mathcal{A}\mathbf{x}^*$ is positive definite[4] ($S \succ 0$). When the same experiment as above is repeated for general $\mathbf{x}$, the coefficient recovery *fails for all sparsity levels* $k = 1, \ldots, M$, when $K > M$.

Thus, non-negativity of the coefficients $\mathbf{x}$ is a required constraint in our sparse coding formulation.

[4] This is non-trivial, and requires some sort of "eigenvalue completion", wherein the general signed coefficients are first randomly sampled, and then adjusted to make the reconstruction S positive definite. Due to this reason, we cannot have a $k = 1$-sparse negative coefficient vector.

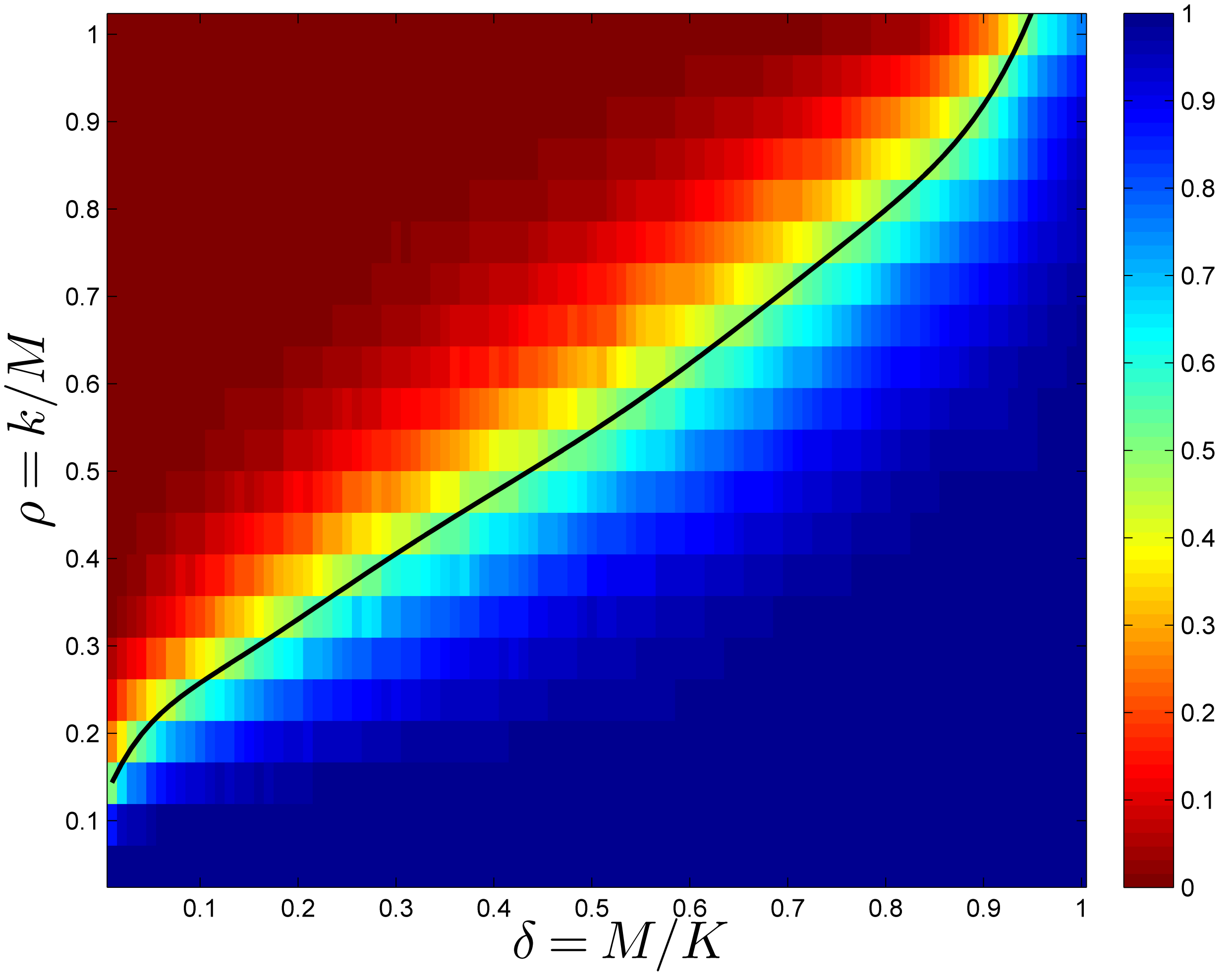


Figure 3.6: Empirical phase transition diagram showing the fraction of correct recovery (corresponding to a relative reconstruction error $\leq 1\%$) for varying undersampling ratio $\delta = M/K$ and sparsity fraction $\rho = k/M$. Results are averaged over $n \in \{3, 4, 5, 6\}$ for 500 trials each.

3.9 Comparison with Vector Sparse Coding

In order to clarify the need for a direct tensor sparse coding method, instead of vectorizing the SPD matrix and performing vector sparse coding, the advantages of the former over the latter must be demonstrated.

The dictionary $\mathcal{A} = \{A_i\}_{k=1}^{K}$ is generated as follows: each positive definite dictionary atom is computed as $A_k = W_k W_k^T$, where $W_k \in \mathbf{R}^{n \times n}$ and each $W_k(i,j)$, $i,j = 1, \ldots, n$, is sampled i.i.d from $\mathcal{U}(0,1)$. A known k-sparse vector $\mathbf{x}^* \in \mathbf{R}_+^K$ is first generated - the support of $\mathbf{x}^*$ is generated by selecting k of K locations uniformly at random without replacement, and the non-zero values in $\mathbf{x}^*$ are sampled i.i.d. from $\mathcal{U}(0,1)$. The true signal is constructed as $S^* = \mathcal{A}\mathbf{x}^*$, and the test signal S to be sparse-coded is obtained as the sample covariance from a set of N i.i.d. multivariate Gaussian samples from $\mathcal{N}(\mathbf{0}, S^*)$ (with $N = 10n^2$). The sample covariance matrix of a multivariate Gaussian distribution follows a Wishart distribution [Wishart, 1928], and therefore our optimization problem is well suited to this model.

The quantities we consider to represent the performance of the reconstruction are the Logdet divergence $D_{\text{ld}}(\hat{S}, S)$, the geodesic distance $D_{\text{geo}}(\hat{S}, S)$, the ℓ_1 norm $\|\hat{\mathbf{x}}\|_1$ of the estimated coefficient vector $\hat{\mathbf{x}}$ and the minimum eigenvalue $\lambda_{\min}(S - \hat{S})$ of the residual $(S - \hat{S})$.

Since we know the true $\mathbf{x}^*$ that generated the test signal S from the dictionary, we can consider the efficiency in recovering this true coefficient vector. The ℓ_1-constrained sparse coding technique is used, where the constraint T is varied as a fraction of the true required 'budget' $\|\mathbf{x}^*\|_1$, *i.e.*, $T \in [0, \|\mathbf{x}^*\|_1]$. We show results for cases where the constraint $\hat{S} \preceq S$ is retained ("2-cone") and relaxed ("1-cone"). For a baseline, we also show the performance of the 1-nearest-neighbor reconstruction (1-NN), where $\mathbf{x}^*$ is an all-zero vector except for a non-zero coefficient at the index corresponding to the nearest atom.

For the vector sparse coding case, we vectorize, for both the signal and the dictionary, and solve the following optimization problem:

$$\begin{array}{ll} \min\limits_{\mathbf{x} \geq 0} & \|\mathbf{s} - D\mathbf{x}\|_2^2 \\ \\ \text{s.t.} & \|\mathbf{x}\|_1 \leq T, \end{array}$$

where $\mathbf{s} = \text{vecu}(S)$, $D = [\mathbf{a}_1 \dots \mathbf{a}_K]$ where $\mathbf{a}_i = \text{vecu}(A_i)$, and vecu is a function denoting the vectorization of the upper triangular part of the argument matrix. We retain the non-negativity constraint on the coefficients here as well for a fair comparison. The matrix reconstruction is then obtained as $\hat{S} = \text{vecu}^{-1}(\hat{\mathbf{s}})$ where $\hat{\mathbf{s}} = D\mathbf{x}$ and vecu^{-1} denotes the inverse of the upper triangular vectorization operation. This is repeated for matrix logarithms (since $\log : \mathbf{S}^n_{++} \mapsto \mathbf{S}^n$) and the Cholesky factors of the positive definite matrices.

We compare the geodesic distance between the reconstruction and the true covariance $D_{\text{geo}}(\hat{S}, S^*)$ as well as the error in the coefficient vector $\|\mathbf{x} - \mathbf{x}^*\|_2^2$ in the tensor and vector sparse coding approaches.

This is performed over 100 different coefficient vectors, given a fixed dictionary. The ℓ_1-constrained sparse coding is used for both the tensor and vector cases, and the constraint T is varied as a fraction of the true required 'budget' $\|\mathbf{x}^*\|_1$.

Figure 3.7 shows the comparison of geodesic distance between the reconstruction and the true covariance, for varying 'budget' constraints on the ℓ_1 norm of $\mathbf{x}$. Clearly the tensor sparse coding provides a more rigorous reconstruction in terms of the distance metric on the manifold. In fact even when the full ℓ_1 budget is provided, the vector case does not provide as good a reconstruction as the tensor algorithm that operates directly in the space of SPD matrices. The plot is shown in a log-scale to clearly show the gap between the two curves at $T = \|\mathbf{x}^*\|_1$. The "1-cone" and "2-cone" curves are alike up to a certain T, but after that the effect of the extra constraint in preventing a more closer approximation is visible.

From a sparse signal recovery viewpoint, we may compare the coefficient estimation error, also shown in Figure 3.7. In this case as well, the tensor sparse coding outperforms the vector method above a certain ℓ_1 constraint limit. The results are shown for three different problem sizes (n, K, k): $(5, 15, 3)$, $(6, 18, 3)$ and $(7, 28, 3)$.

This experiment validates the importance of being able to perform sparse coding of positive definite matrices directly without resorting to vectorization.

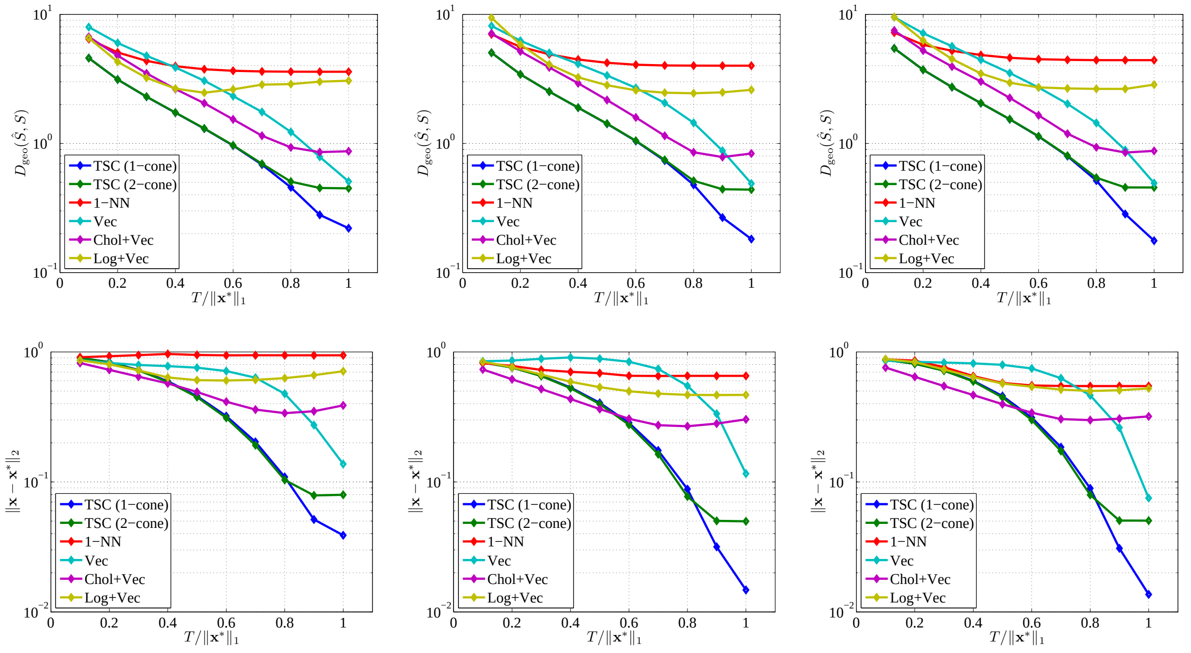


Figure 3.7: Comparison of 1-NN, tensor and vector sparse coding - geodesic distance (upper row) and coefficient estimation error (lower row). The x-axis shows the normalized ℓ_1 constraint parameter $T/\|\mathbf{x}^*\|_1$, *i.e.*, the ℓ_1 'budget' is varied as a fraction of the ℓ_1 norm of the true solution $\mathbf{x}^*$. The problem sizes are $(n, K, k) = (5, 15, 3)$ for column 1, $(6, 18, 3)$ for column 2, and $(7, 28, 3)$ for column 3 (Best viewed in color).

3.10 Relation between D_{geo} and D_{ld}

In this section we derive an interesting connection between the Riemannian geodesic distance and the LogDet divergence. Let $\lambda \sim \lambda(A, B)$ be the generalized eigenvalues of two positive definite matrices (A, B).

The Riemannian geodesic distance between A and B is given by

$$D_{\text{geo}}(A, B) = \left\| \log \left(B^{-1/2} A B^{-1/2} \right) \right\|_F \quad . \tag{3.37}$$

In terms of the generalized eigenvalues, the geodesic distance

$$D_{\text{geo}}(A, B) = \|\log \lambda\|_2 = \left\| \log \left(\frac{1}{\lambda} \right) \right\|_2 \quad . \tag{3.38}$$

The general form of a Bregman divergence for matrix arguments is given by [Kulis et al., 2009]

$$D_\varphi(X, Y) = \varphi(X) - \varphi(Y) - \langle \nabla \varphi(Y), (X - Y) \rangle, \tag{3.39}$$

where $\varphi(\cdot)$ is a strictly convex function over a convex set $\mathcal{S}$, and is differentiable in relint$(\mathcal{S})$ (relative interior). The last term denotes the matrix inner product $\langle A, B \rangle \; = \; \operatorname{tr}\left(AB^T \right)$.

The LogDet divergence is derived from $\varphi(X) = -\log \det X$ and is given by:

$$D_{\text{ld}}(A, B) = \log \det A^{-1} - \log \det B^{-1} - \langle -B^{-1}, A - B \rangle$$

$$\text{since } \nabla \left(-\log \det X \right) = -X^{-1}$$

$$= -\log \det \left(B^{-1} A \right) + tr \left(B^{-1} A - B^{-1} B \right)$$

$$\therefore D_{\text{ld}}(A, B) = tr \left(B^{-1} A \right) - \log \det \left(B^{-1} A \right) - n. \tag{3.40}$$

The second term in the above equation can be written in terms of λ as:

$$-\log \det \left(B^{-1} A \right) = \operatorname{tr} \left(\log \left(B^{-1} A \right)^{-1} \right) = \sum_{i=1}^{n} \log \left(\frac{1}{\lambda_i} \right). \tag{3.41}$$

In our sparse coding formulation, we require that the approximation $\hat{S} \preceq S$, the original signal. If $B = S$ and $A = \hat{S}$, then $A \preceq B$, or $B^{-1}A \preceq I_n$. Therefore, for $i = 1, \dots, n$,

$$\lambda_i \leq 1 \Longrightarrow \frac{1}{\lambda_i} \geq 1 \Longrightarrow \log\left(\frac{1}{\lambda_i}\right) \geq 0. \tag{3.42}$$

Since the elements in the sum are all non-negative,

$$-\log\det\left(AB^{-1}\right) = \sum_{i=1}^{n} \log\left(\frac{1}{\lambda_i}\right) = \sum_{i=1}^{n} \left|\log\left(\frac{1}{\lambda_i}\right)\right| \tag{3.43}$$

$$= \left\|\log\left(\frac{1}{\lambda}\right)\right\|_1. \tag{3.44}$$

Looking back into the expression for the Logdet divergence, we have

$$D_{\mathrm{ld}}(A, B) = \left\|\log\left(\frac{1}{\lambda}\right)\right\|_1 + \langle B^{-1}, A - B\rangle, \tag{3.45}$$

which is a combination of

1. an ℓ_1-norm term of reciprocal generalized eigenvalues of (A, B), denoted by $D_{\mathrm{L1}}(A, B)$, and
2. the component of the difference between A and B in the direction of the tangent of $\varphi(\cdot) = -\log\det(\cdot)$ evaluated at B.

When λ is very close to 1, or $|1 - \lambda| \ll 1$, setting $x = 1 - \lambda$ and using the Taylor's approximation $\log(1 + x) \approx x$ when $|x| \ll 1$, the geodesic distance can be rewritten as follows:

$$D_{\mathrm{geo}}(A, B) = \|\log(\lambda)\|_2 \approx \|\lambda - 1\|_2$$

$$= \left\|B^{-1}A - I_n\right\|_F$$

$$= \left\|B^{-1}(A - B)\right\|_F$$

$$D^2_{\mathrm{geo}}(A, B) \approx tr\left\{\left(B^{-1}(A - B)\right)^2\right\} \quad \text{when } \lambda \approx 1. \tag{3.46}$$

Similarly, rewriting the second term in Equation (3.45), we get

$$D_{\mathrm{ld}}(A,B) = \left\|\log\left(\frac{1}{\lambda}\right)\right\|_1 + tr\left\{B^{-1}(A-B)\right\}. \tag{3.47}$$

It is interesting that the second term of the Logdet divergence forms a different ℓ_1-ℓ_2 type similarity with the approximate geodesic distance when $\lambda \approx 1$. Thus there is a two-fold connection between the Riemannian geodesic distance and the LogDet divergence.

Therefore, in our framework, specifically under the condition that $\hat{S} \preceq S$,

$$D_{\mathrm{geo}}(A,B) = \left\|\log\left(\frac{1}{\lambda}\right)\right\|_2 \tag{3.48}$$

$$D_{\mathrm{L1}}(A,B) = \left\|\log\left(\frac{1}{\lambda}\right)\right\|_1 \tag{3.49}$$

$$D_{\mathrm{ld}}(A,B) = D_{\mathrm{L1}}(A,B) + tr\left\{B^{-1}(A-B)\right\} \tag{3.50}$$

$$D^2_{\mathrm{geo}}(A,B) \approx tr\left\{\left(B^{-1}(A-B)\right)^2\right\} \quad \text{when } \lambda \approx 1. \tag{3.51}$$

This clearly illustrates an analogy of the geodesic distance and the LogDet divergence to the ℓ_2 and ℓ_1 distances in more than one way.

This supports the use of the LogDet divergence in our model, and also intuitively explains the similarity in the trend of the geodesic distance and LogDet divergence across varying approximations in the sparse coding decompositions. Further, since the ℓ_1 norm tends to push most of the components to zero, the ℓ_1 term on the log-reciprocal generalized eigenvalues pushes most of the generalized eigenvalues to 1, thus giving us a closer approximation $\hat{S}$ to S, and a semidefinite residual $E = S - \hat{S}$.

The three dissimilarity measures can be compared for the simple case of 2×2 SPD matrices, and the eigenvalues (λ_1, λ_2) are varied in $[0,1]$, the domain of our problem. In Figure 3.8, we show the slice of this surface at $\lambda_1 = \lambda_2$.

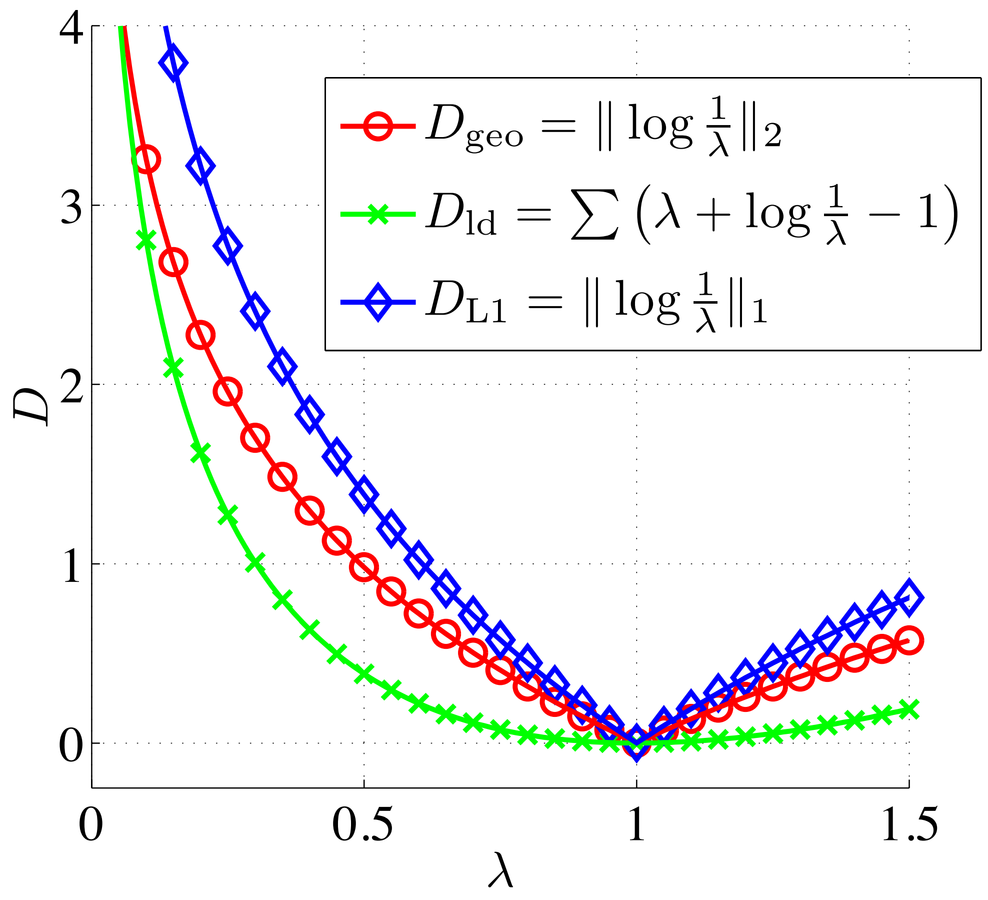


Figure 3.8: Comparison of dissimilarity measures in the 2×2 case: Slice at $\lambda_1 = \lambda_2 = \lambda$. Clearly all three distance functions have their minimum at $\lambda_1 = \lambda_2 = 1$. In terms of how 'strong' the objective function is in pushing the λ_i's to 1, $D_{\text{ld}} < D_{\text{geo}} < D_{\text{L1}}$.

3.11 Relaxation of the Residual Constraint

In other sparse-coding or reconstruction problems, the goal is to reach within a certain ϵ-ball around the target, from any possible direction. Since the constraint of $\mathcal{A}\mathbf{x} \preceq S$ allows us to approach the target only from one side, we relax this to achieve a closer approximation, with all other constraints and parameters staying the same. While it is straightforward to understand that the relaxation of this constraint enables better approximations, this is also illustrated in the following experiment.

For different values of the data dimension n, we synthesize dictionaries of $K = n(n+1)/2$ atoms. Random positive definite signals $\mathcal{S} = \{S_1, \ldots, S_{100}\}$ are generated, and are sparse coded over the corresponding dictionary, both with and without the constraint $\mathcal{A}\mathbf{x} \preceq S$. The optimum approximation error using both approaches are compared - we denote D_1 to be the average approximation error when the constraint is removed, and D_2 to be that when the constraint is present. The results were averaged over 10 trials, and the ratio D_1/D_2 is shown in Figure 3.9. As n increases, the improvement in the approximation when the upper cone constraint is relaxed is more pronounced.

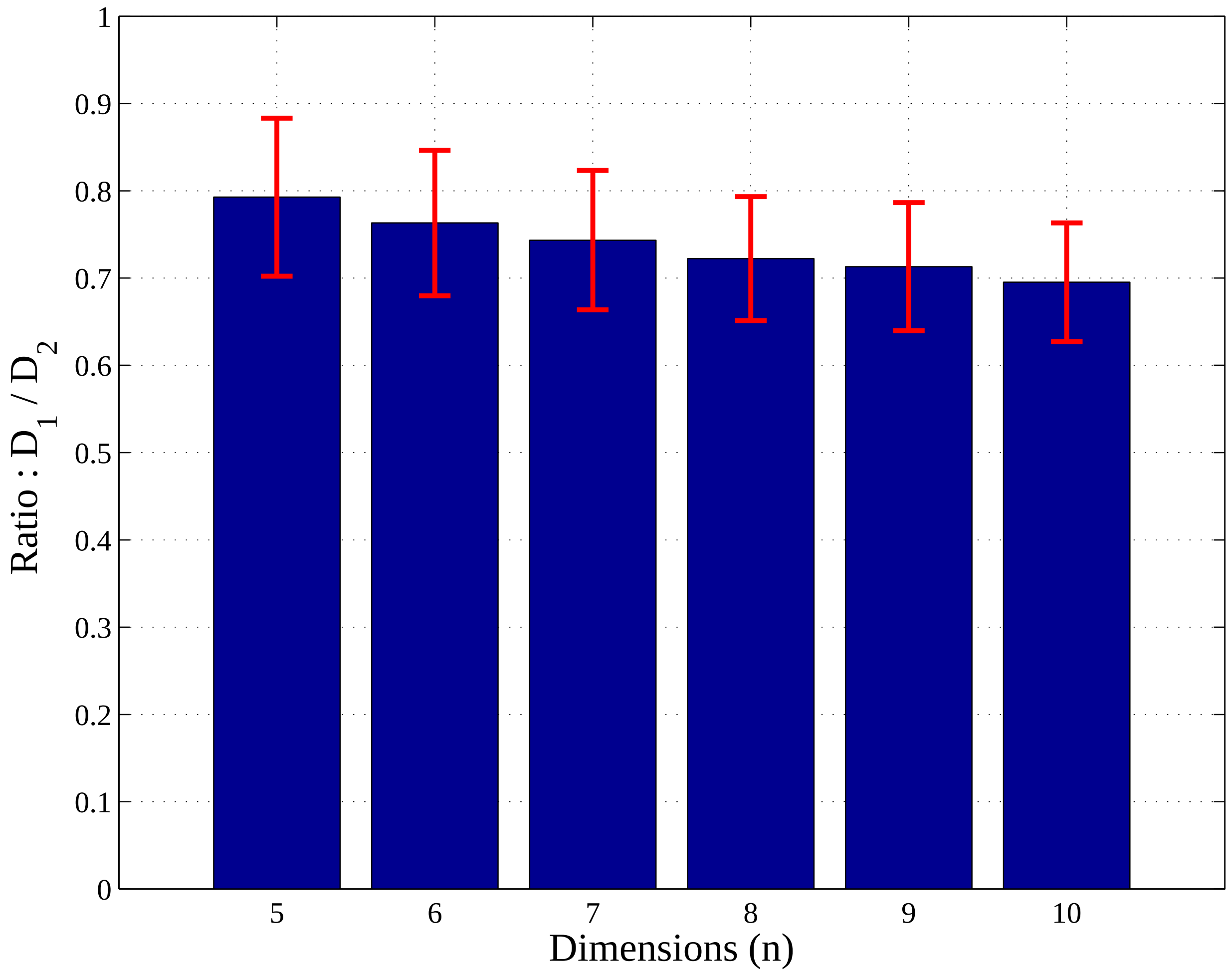


Figure 3.9: Relative reconstruction error with the estimated coefficient vector $\hat{\mathbf{x}}$ with (D_2) and without (D_1) the $\mathcal{A}\mathbf{x} \preceq S$ constraint. The improvement in the approximation error is shown for different values of n, and averaged over 25 iterations. As n increases, this improvement is more pronounced. (1σ bars are also shown.)

3.12 An Efficient Sparse Coding Algorithm

In this section, we present our implementation for solving the ℓ_1-regularized sparse coding problem, in which the upper cone constraint is relaxed:

$$\min_{\mathbf{x}\geq 0} \quad D_{\mathrm{ld}}\left(\mathcal{A}\mathbf{x}, S\right) + \lambda \left\|\mathbf{x}\right\|_1 \tag{3.52a}$$

$$\text{s.t.} \quad \mathcal{A}\mathbf{x} \succeq 0 \tag{3.52b}$$

We developed a first-order coordinate descent approach [Luo and Tseng, 1992] to sparse coding, which updates one coordinate in the coefficient vector $\mathbf{x}$ at a time. We cyclically iterate over each of the coefficients and repeat this process until convergence.

Expanding and simplifying the objective in the sparse coding problem (3.52) and removing terms independent of $\mathbf{x}$:

$$\min_{\mathbf{x}\geq 0} \quad \sum_{i=1}^{K} \left(c_i + \lambda\right) x_i - \log\det\left(\sum_{i=1}^{K} x_i A_i\right) \tag{3.53a}$$

$$\text{s.t.} \quad \sum_{i=1}^{K} x_i A_i \succeq 0, \tag{3.53b}$$

where $c_i = \operatorname{tr}\left(A_i S^{-1}\right)$, $i = 1, \dots, K$. Let the objective in (3.53) be denoted by $f\left(\mathbf{x}\right)$.

To update coordinate x_k,

$$\min_{x_k \geq 0} \quad \left(c_k + \lambda\right) x_k - \log\det\left(\sum_{i=1}^{K} x_i A_i\right). \tag{3.54}$$

Denote the objective in (3.54) to be $g(x_k)$. The gradient with respect to the variable x_k is:

$$\nabla g(x_k) = c_k + \lambda - \operatorname{tr}\left(\left(\sum_{i=1}^{K} x_i A_i\right)^{-1} A_k\right) = c_k + \lambda - \operatorname{tr}\left(A_k \hat{S}^{-1}\right). \tag{3.55}$$

The minimum of the convex function $g(x_k)$ can be found by setting $\nabla g(x_k) = 0$. Unfortunately, this does not admit a closed form expression to solve for x_k. However, we can solve this one-dimensional optimization by proceeding along the gradient descent direction $\delta x_k = -\nabla g(x_k)$ at the appropriate stepsize β selected by exact line search. Since the magnitude of δx_k can be absorbed into the stepsize β, we are only interested in the sign of δx_k.

Therefore, the update expression for coordinate x_k is given by:

$$x_k \leftarrow x_k + \beta \delta x_k \tag{3.56}$$

with the descent direction

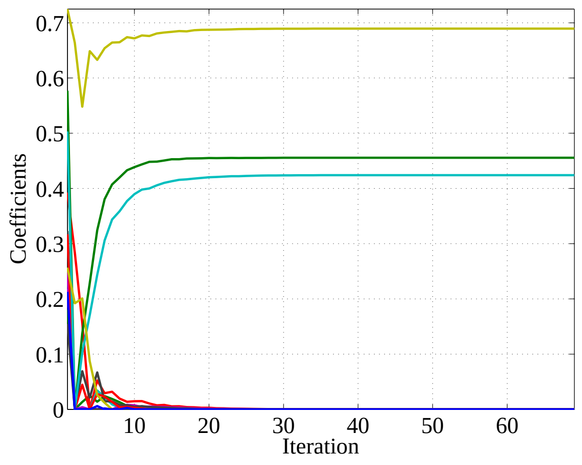
$$\delta x_k = \operatorname{sign}\left(\operatorname{tr}\left(A_k \hat{S}^{-1}\right) - c_k - \lambda\right). \tag{3.57}$$

The stepsize β is chosen by line search along δx_k such that it minimizes $f\left(\mathbf{x} + \beta \delta x_k \mathbf{e}_k\right)$, where $\mathbf{e}_k \in \mathbf{R}^K$ is the k-th canonical basis vector. While exact line search is preferred, it is also possible to use backtracking (Armijo) line search to ensure a sufficient reduction in the objective function at each iteration.

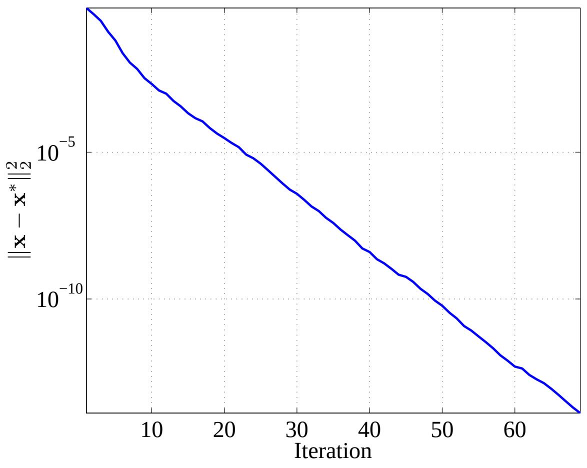
We mention here some empirical observations seen in practice: we see that a single step of stepsize $\beta = |\nabla f(x_k)|/|\nabla^2 f(x_k)|$ at each iteration works amazingly well. In very few iteration, the zeros of the true coefficient $\mathbf{x}^*$ are attained, and the non-zero coordinates come very close to their true target values, thereafter converging linearly. This behavior is shown for a synthetic example in Figure 3.10. We can further speed up the algorithm by choosing a smart initialization of $\mathbf{x}$. Note that when the true solution $\mathbf{x}^*$ is 1-sparse, that one non-zero value is equal to $\min_\lambda \lambda(S, A_{i^*})$, where i^* is the location of the non-zero coordinate, and $\lambda(S, A_i)$ are the generalized eigenvalues of the pair (S, A_i). Therefore, instead of initializing the coefficients to ϵ, we observed that setting the initial value $x_i = \min_\lambda \lambda(S, A_i)$ gives the fastest solution. When $\mathbf{x}^*$ is 1-sparse the corresponding coordinate in the initial $\mathbf{x}$ already has the right coefficient, and it now becomes a matter of just identifying the right support (which as mentioned is extremely fast).

Algorithm 1 **Coordinate Descent Algorithm for Positive Definite Sparse Coding**

```
Input: Signal $S$, dictionary $\mathcal{A} = \{A_i\}_{i=1}^K$, parameter $\lambda$
Output: Coefficient vector $\mathbf{x}$
  Compute $\mathbf{c}$: $c_i = \mathrm{tr}\,(A_i S^{-1})$, $i = 1, \dots, K$
  Initialize $\mathbf{x} = \epsilon \mathbf{1}$, $\epsilon > 0$
  Compute $\hat{S} = \sum_{i=1}^K x_i A_i$
  repeat
    for $k = 1$ to $K$ do
      Compute descent direction $\delta x_k = \mathrm{sign}\left(\mathrm{tr}\left(A_k \hat{S}^{-1}\right) - c_k - \lambda\right)$
      Compute stepsize $\beta$ along $\delta x_k$ that minimizes $f\,(\mathbf{x} + \beta \delta x_k \mathbf{e}_k)$
      Update $x_k \leftarrow (x_k + \beta \delta x_k)_+$, where $(a)_+ = \max\,(a, 0)$
      Set $\alpha = x_k^{\mathrm{new}} - x_k^{\mathrm{old}}$
      Update $\hat{S} \leftarrow \hat{S} + \alpha A_k$
    end for
  until convergence
```



(a)



(b)

Figure 3.10: Convergence of the coordinate descent procedure for tensor sparse coding ($n = 5, K = 15, k = 3$.) In very few iterations, the zeros are attained and the non-zeros reach close to their final values.

Our optimized C++ implementation of the sparse coding coordinate descent algorithm yields an average of 2 orders of magnitude[5] speed-up compared to the interior-point implementations of SDPT3 [Tutuncu et al., 2003] (with YALMIP [Löfberg, 2004]) in MATLAB. We utilize the *Eigen* library [Guennebaud et al., 2010] with OpenMP on an Intel Core i7-Q720 1.6GHz 64-bit QuadCore laptop with 4GB RAM. Figure 3.11 shows the average sparse coding time *in milliseconds* per signal for different values of (n, K), averaged over 50 runs of 1000 signals each. Figure 3.12 shows the sparse coding timing for different values of (K, λ) for $n = 10$.

[5] In practice we saw speed-ups between 50 and 250.

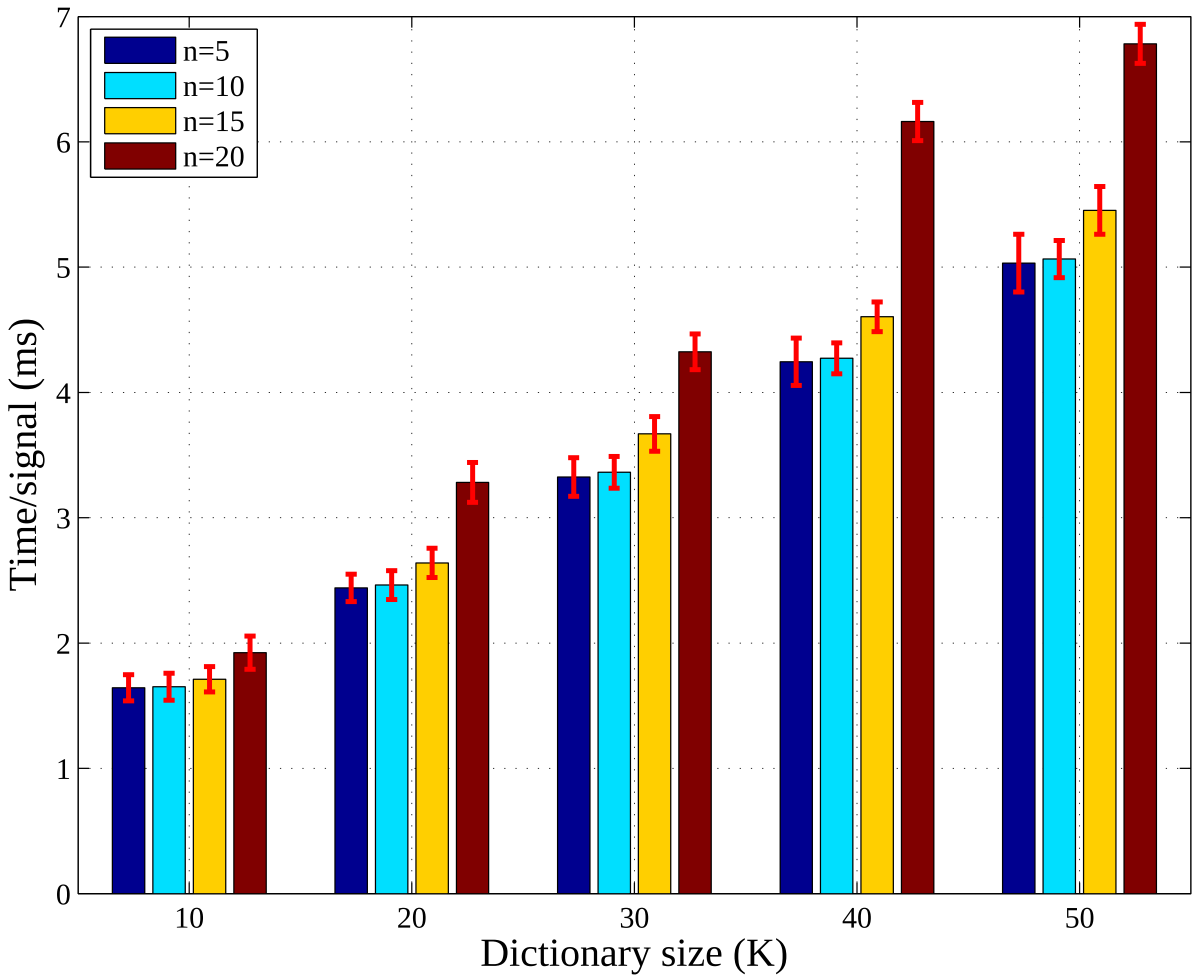


Figure 3.11: Average sparse coding times per $n\times n$ positive definite signal S over a dictionary $\mathcal{A}$ of size K, using the coordinate descent algorithm ($\lambda = 0.1$). 3σ bars are also shown.

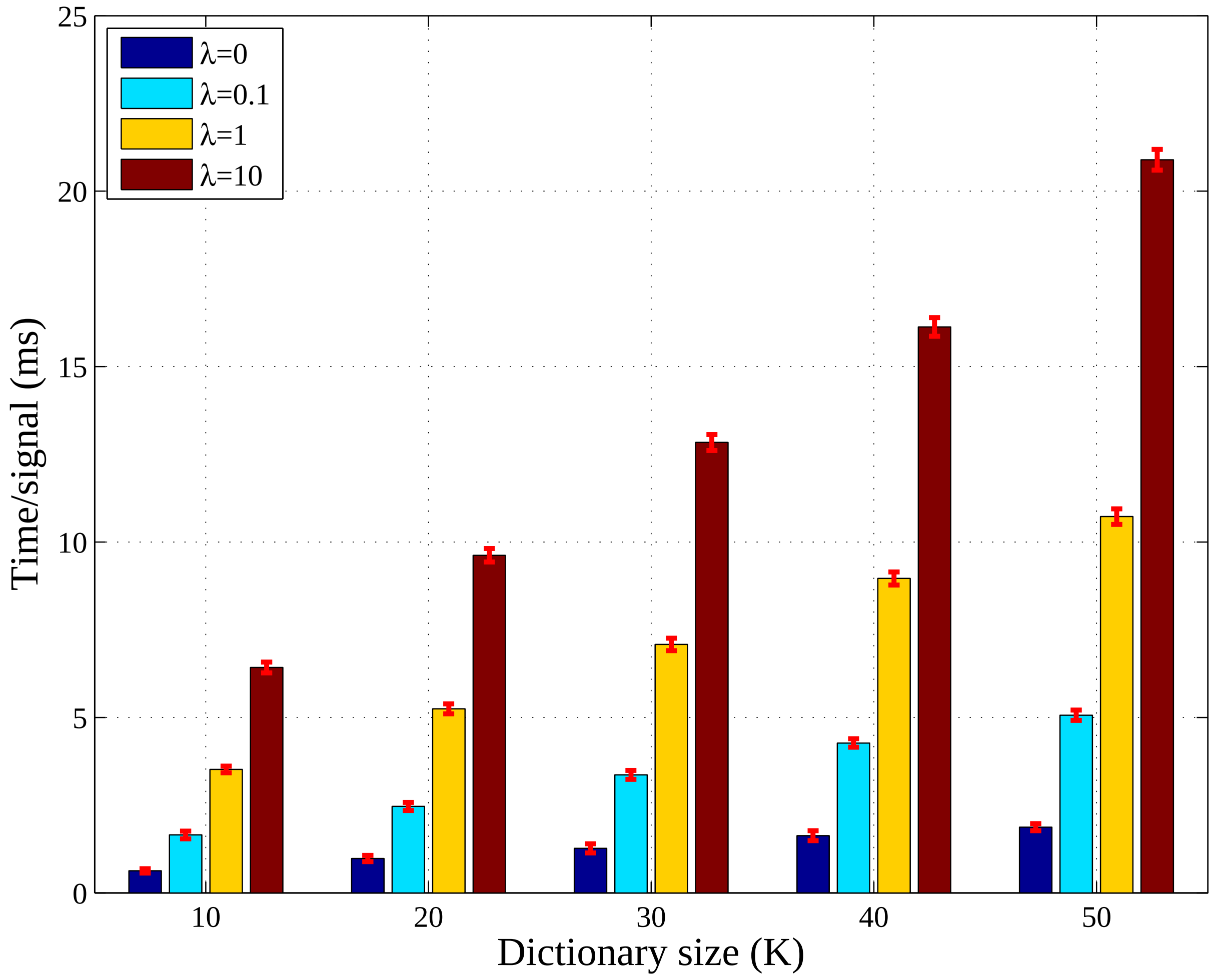


Figure 3.12: Average sparse coding times per 10×10 positive definite signal S over a dictionary $\mathcal{A}$ of size K, using the coordinate descent algorithm for varying λ. 3σ bars are also shown.

3.13 Tensor Sparse Coding for Classification

Sparse coding has been applied to classification problems in many domains. Here we present applications where we use the tensor sparse coding for classification. Let us denote the number of classes by C. Two of the main approaches to classifying with dictionaries are:

1. Maintain separate dictionaries for each class $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_C$. Sparse code the test signal S independently over each dictionary to get the coefficients $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_C$ respectively.

2. Create a combined dictionary $\mathcal{A} = [\ \mathcal{A}_1 \mid \mathcal{A}_2 \mid\ \ldots\ \mid \mathcal{A}_C\ ]$ by concatenating the individual class dictionaries, each of size N_c, $c = 1, \ldots, C$. The test signal S is sparse coded over the combined dictionary to obtain the coefficient vector $\mathbf{x}$, which is then split into the components corresponding to each of the sub-dictionaries $\mathbf{x} = \left[\ \mathbf{x}_1^T\ \mathbf{x}_2^T\ \ldots\ \mathbf{x}_C^T\ \right]^T$.

The different class reconstructions are computed as $\hat{S}_c = \mathcal{A}_c \mathbf{x}_c$, $c = 1, \ldots, C$, and the test signal is assigned the label c^* of the class which gives the closest approximation:

$$\text{label}\ \ c^* = \arg\min_{c}\ D_{\text{ld}}\left(\hat{S}_c, S\right). \tag{3.58}$$

These two approaches are usually referred to as *reconstruction error-based* classifiers - the former is denoted the *separate dictionary* approach, and the latter as the *combined dictionary* approach.

It is also possible to get the reconstruction error for different values of the regularizer λ (or constraint T in Type-II sparse coding), and use the sequence of reconstruction errors $D_{\text{ld}}(\lambda)$ (or $D_{\text{ld}}(T)$) as a feature for classification with another classifier, say, support vector machines (SVM). The coefficients themselves are useful indicators and can be used as features. In a combined dictionary approach, the distribution of the coefficients over the different classes shows a sort of affinity towards the respective classes. [Castrodad and Sapiro, 2011] use the ℓ_1 norms of the individual class coefficient vectors as features $\phi = [\ \|\mathbf{x}_1\|_1 \ \ \|\mathbf{x}_2\|_1 \ \ \ldots \ \ \|\mathbf{x}_C\|_1\ ]^T \in \mathbf{R}^C$ in a second level of dictionary learning, forming a type of *deep learning* approach.

3.13.1 Human Appearance Descriptors

In this section, we present experiments on classification of human appearances, based on region covariance features. We use a subset of the 18-class *Cam5* dataset from [Sivalingam et al., 2009], from which we choose the 16 classes which contain at least 10 data points each. The dataset contains a total of 407 images from these 16 classes. Representative images from the dataset are shown in Figure 3.13. The descriptors are 5×5 covariances computed from the $\{$R,G,B,I_x,$I_y\}$ features at each pixel corresponding to the human foreground blobs. From each of the 16 classes, we select 5 points for training and the remaining are used for testing.

Our sparse coding method is used for classification as follows: The training data from each class forms a dictionary $\mathcal{A}_m$, $m = 1, \ldots, M$, where M is the number of classes ($M = 16$ here). The class dictionaries are concatenated into one large dictionary $\mathcal{A}$:

$$\mathcal{A} = [\ \mathcal{A}_1 \mid \mathcal{A}_2 \mid \ \ldots \ \mid \mathcal{A}_M\ ] \,.$$

The test signal S is sparse coded over this *combined* dictionary, to yield a sparse coefficient vector $\mathbf{x}$. This vector consists of the coefficients corresponding to atoms from different classes $1, \ldots, M$, and can be written as

$$\mathbf{x} = \left[\ \mathbf{x}_1^T \mid \mathbf{x}_2^T \mid \ \ldots \ \mid \mathbf{x}_M^T\ \right]^T .$$

The class-wise reconstruction $\hat{S}_m$ is then obtained as $\hat{S}_m = \mathcal{A}_m \mathbf{x}_m$, and the class-wise reconstruction error is computed as $E_m = D_{\text{ld}}\left(\hat{S}_m, S\right)$. The label m^* of the dictionary offering the minimum reconstruction error is then assigned to the test signal S.

$$m^* = \arg\min_m \ D_{\text{ld}}\left(\hat{S}_m, S\right) .$$

This approach is adapted from [Wright et al., 2009], and we refer to this as the *combined dictionary approach.*

We apply this combined dictionary approach to the problem of classifying human appearances, forming a dictionary $\mathcal{A}$ of $K = 80$ atoms. For this experiment, in addition to the reconstruction error-based classification (REC), we also compute a weighted label vote (WLV) for each class from the corresponding coefficient values, and use this as a score for classification:

$$m^* = \arg\max_m \quad \|\mathbf{x}_m\|_1.$$

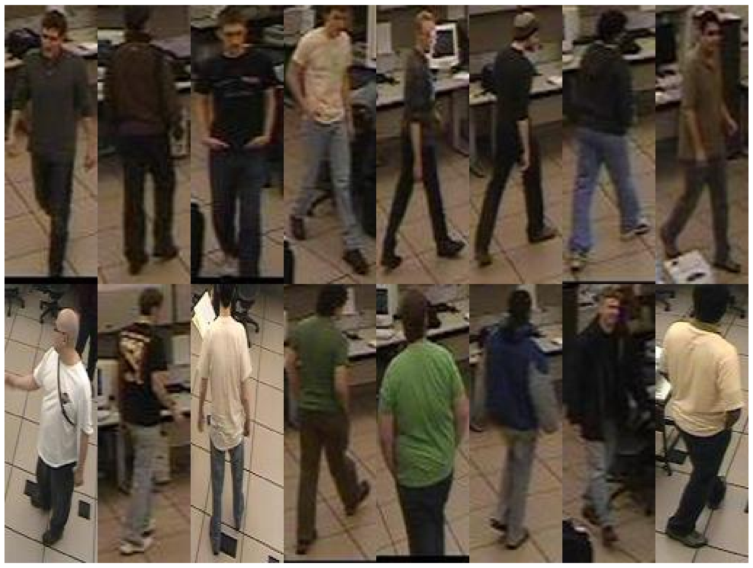


Figure 3.13: Representative images from the *Cam5* dataset.

The classification accuracy for this dataset averaged over 100 random train-test splits is shown in Table 3.1. The sparse coding results provide a notable increase in accuracy compared to the KNN or SVM techniques. We also show the REC and WLV classification accuracies with the vectorized upper-triangular parts of the covariances. This is obtained using traditional vector sparse coding (VSC), *i.e.*, the Lasso problem of [Tibshirani, 1996]. In addition, the vectorized upper-triangular part of the Cholesky factor of each positive definite matrix descriptor is also used in the vector sparse coding framework for both REC and WLV classification. These results are also included in Table 3.1.

The tensor sparse coding approaches for appearance recognition outperform the KNN and SVM baseline algorithms, and also the vector sparse coding-based approaches. This demonstrates that sparse coding techniques that retain the positive definiteness of the data

Classifier	Accuracy (%)
Geo-KNN ($K = 5$)	66.95 (4.89)
Geo-SVM ($\sigma = 0.5$)	77.64 (5.96)
VSC + WLV (Vec)	62.00 (3.89)
VSC + REC (Vec)	62.16 (3.67)
VSC + WLV (Chol)	73.53 (2.98)
VSC + REC (Chol)	76.40 (2.84)
TSC + WLV	**78.62** (1.49)
TSC + REC	77.85 (2.50)

Table 3.1: Mean classification accuracy for the *Cam5* dataset. Results are averaged over 100 trials and standard deviation values are also shown in parentheses.

points yield better results not only with synthetic data but also in practical computer vision applications.

3.13.2 Tensor Sparse Coding for Face Recognition

In this section, we present experimental results for face recognition from grayscale images. This is performed over a subset of the FERET face database [Phillips et al., 2000], consisting of grayscale images of 10 subjects, where each individual represents a separate class. The frontal or near-frontal images corresponding to the two-letter codes *'ba'*, *'bd'*, *'be'*, *'bf'*, *'bg'*, *'bj'*, and *'bk'* are used for our experiments, leading to a total of 70 face images. We extract Gabor-based region covariances from each face image following the approach of Pang et al. [Pang et al., 2008].

We crop the images based on the eye positions, and resize them to be of size 60×60 pixels. The Gabor filters [Pang et al., 2008] corresponding to 8 orientations ($u = 0, \ldots, 7$) and 5 scales ($v = 0, \ldots, 4$) are applied to each image, resulting in 40 different filter responses g_{uv}. In addition, we also test on features such as (x, y) spatial location of pixels in the image, image intensity I, derivatives of image intensity I_x, I_y, I_{xx}, I_{yy} and gradient orientation $\arctan I_y/I_x$. The different sets of features used in the covariance descriptor construction are described in Table 3.2.

Mode	Feature Set
1	$[\ x\ y\ I\ \lvert I_x\rvert\ \lvert I_y\rvert\ \lvert I_{xx}\rvert\ \lvert I_{yy}\rvert\ ]$
2	$\left[\ x\ y\ \lvert I_x\rvert\ \lvert I_y\rvert\ \lvert I_{xx}\rvert\ \lvert I_{yy}\rvert\ \arctan \frac{\lvert I_y\rvert}{\lvert I_x\rvert}\ \right]$
3	$[\ x\ y\ \lvert I_x\rvert\ \lvert I_y\rvert\ \lvert I_{xx}\rvert\ \lvert I_{yy}\rvert\ ]$
4	$\left[\ x\ y\ I\ \lvert I_x\rvert\ \lvert I_y\rvert\ \lvert I_{xx}\rvert\ \lvert I_{yy}\rvert\ \arctan \frac{\lvert I_y\rvert}{\lvert I_x\rvert}\ \right]$
5	$[\ x\ y\ g_{00}\ g_{01}\ \cdots\ g_{7v_{\max}}\ ]$
6	$[\ x\ y\ I\ g_{00}\ g_{01}\ \cdots\ g_{7v_{\max}}\ ]$
7	$[\ g_{00}\ g_{01}\ \cdots\ g_{7v_{\max}}\ ]$

.

Table 3.2: Features used in construction of region covariances for face recognition on the FERET face dataset. Feature sets 5–7 consist of 5 subsets each (a)–(e), where the number of octaves is varied from $v_{\max} = 0, \ldots, 4$.

.

Table 3.3: Mean classification accuracy for the *FERET* face recognition dataset. Results are averaged are over 35 trials, and standard deviations are provided in parenthesis.

Mode	Covariances				Precisions				Geo-KNN	Geo-SVM
(n)	Separate (%)		Combined (%)		Separate (%)		Combined (%)		(%)	(%)
	1-cone	2-cone	1-cone	2-cone	1-cone	2-cone	1-cone	2-cone	$K = 1$	$\sigma = 20.0$
1 (7)	**85.81** (9.57)	**85.81** (10.40)	83.24 (10.40)	79.14 (11.14)	76.48 (10.26)	76.67 (11.63)	40.86 (6.14)	43.81 (5.17)	77.62 (9.55)	66.95 (7.23)
2 (7)	69.24 (12.75)	64.76 (11.93)	**71.33** (11.82)	64.19 (13.20)	53.71 (10.59)	54.95 (11.94)	20.95 (7.28)	24.76 (6.49)	62.67 (9.62)	49.62 (8.08)
3 (6)	65.24 (11.96)	64.48 (14.14)	**71.43** (13.12)	65.33 (13.62)	53.24 (10.00)	52.57 (10.78)	16.48 (6.90)	17.43 (6.81)	61.33 (10.76)	49.33 (7.51)
4 (8)	**86.76** (9.27)	84.19 (10.55)	84.76 (10.46)	76.95 (9.87)	77.33 (10.47)	79.05 (11.00)	44.10 (3.57)	48.57 (4.67)	78.48 (10.28)	67.71 (7.84)
5a (10)	**83.52** (10.69)	73.05 (11.69)	**83.52** (12.39)	75.62 (11.43)	38.67 (8.33)	38.29 (9.74)	18.38 (6.87)	18.48 (6.96)	79.62 (12.47)	70.57 (8.38)
5b (18)	93.24 (4.81)	80.00 (8.69)	**94.10** (4.79)	79.81 (8.35)	47.43 (10.14)	53.43 (9.68)	20.19 (7.47)	23.71 (6.51)	86.10 (7.83)	83.62 (7.62)
5c (26)	**93.81** (4.79)	76.19 (7.35)	91.43 (4.80)	72.95 (6.98)	72.57 (10.51)	71.81 (11.91)	50.38 (9.94)	56.10 (8.49)	90.57 (6.78)	88.86 (6.76)
5d (34)	**95.81** (3.77)	74.29 (9.55)	92.48 (4.80)	67.52 (9.41)	81.52 (10.15)	67.14 (10.99)	58.38 (8.52)	63.52 (9.39)	91.81 (6.34)	91.71 (6.29)
5e (42)	94.76 (5.36)	70.00 (10.54)	90.10 (7.01)	64.10 (8.73)	91.62 (6.49)	69.52 (9.92)	76.29 (8.57)	63.62 (10.37)	92.48 (5.54)	**94.95** (4.67)
6a (11)	89.24 (7.81)	80.95 (11.73)	**89.33** (7.92)	81.33 (10.87)	48.76 (10.24)	48.67 (10.05)	20.10 (9.61)	20.19 (9.79)	85.81 (9.10)	79.14 (7.57)
6b (19)	94.10 (4.79)	83.90 (8.22)	**95.33** (4.52)	83.05 (7.53)	54.38 (12.03)	61.81 (8.45)	22.00 (7.61)	27.90 (6.62)	89.81 (5.91)	88.19 (7.53)
6c (27)	**95.62** (3.63)	79.43 (6.64)	92.86 (4.00)	76.00 (8.20)	74.48 (10.98)	76.10 (12.28)	51.43 (9.80)	58.86 (10.30)	93.14 (6.07)	91.14 (6.37)
6d (35)	**96.48** (3.73)	75.14 (9.13)	94.29 (4.33)	70.38 (8.76)	84.19 (8.99)	68.67 (11.77)	61.43 (8.02)	65.33 (9.77)	92.76 (6.09)	92.57 (5.41)
6e (43)	**95.52** (4.91)	71.05 (11.35)	91.24 (6.67)	65.24 (10.09)	93.24 (5.77)	70.10 (9.14)	78.10 (7.86)	64.57 (10.11)	92.76 (5.49)	**95.52** (4.29)
7a (8)	78.76 (9.30)	73.24 (11.03)	**79.52** (9.96)	73.24 (9.81)	38.86 (9.01)	39.24 (8.84)	25.71 (6.20)	25.81 (6.14)	70.95 (12.74)	63.43 (10.88)
7b (16)	**92.19** (5.80)	77.81 (8.24)	91.71 (5.93)	77.52 (7.74)	46.29 (10.26)	50.10 (7.99)	20.67 (6.70)	21.90 (5.82)	83.62 (9.84)	84.29 (7.02)
7c (24)	**92.10** (5.86)	75.43 (5.35)	87.14 (5.92)	71.90 (8.41)	69.62 (9.01)	65.71 (11.42)	48.86 (9.15)	53.14 (8.35)	86.10 (7.62)	86.29 (7.47)
7d (32)	**93.05** (4.94)	72.29 (9.49)	90.48 (5.86)	65.24 (9.16)	78.67 (8.88)	62.29 (11.15)	53.05 (9.51)	57.81 (8.76)	89.14 (7.14)	89.43 (6.50)
7e (40)	**93.05** (5.77)	68.86 (11.27)	88.29 (6.34)	61.90 (12.35)	84.95 (7.36)	68.86 (9.01)	72.67 (8.00)	60.67 (9.75)	89.71 (6.44)	92.95 (5.39)
Mean	**88.86** %	75.31 %	87.50 %	72.18 %	66.63 %	61.84 %	42.11 %	42.96 %	**83.92** %	**80.33** %

We compute the region covariance descriptor over the entire face only, and not subsections of each face image as was done in [Pang et al., 2008]. At each iteration of the experiment, 4 out of 7 images from each subject are taken for training, and the remaining 3 are used as test images, yielding a total of $\binom{7}{3} = 35$ different train-test splits.

The face recognition is performed using the reconstruction error-based approach. In addition to the combined dictionary approach explained before, we also classify the signal by sparse coding it with each class dictionary $\mathcal{A}_m$ independently to obtain the coefficient vector $\mathbf{x}_m$, and predicting the label m^* as:

$$m^* = \arg\min_{m} \; D_{\text{ld}}\left(\hat{S}_m, S\right).$$

We refer to this method as the *separate dictionary approach.*

The dictionaries are composed of the covariance descriptors from the training images. This is compared to the recognition performance using geodesic KNN and geodesic SVM.

Since the inverse of a positive definite matrix is also positive definite, we repeat the same experiment with the inverse covariances (or precision matrices). Since the geodesic distance between two matrices A and B is identical to that between A^{-1} and B^{-1},

$$D_{\text{geo}}(A, B) = D_{\text{geo}}\left(A^{-1}, B^{-1}\right),$$

the KNN and SVM classifiers do not differ in performance between covariance and precision matrices.

Further, we show the recognition performance when the upper cone constraint is relaxed ("1-cone") and compare it to the case where it is retained ("2-cone").

The mean classification accuracy over 35 trials is presented in Table 3.3 for each covariance feature mode. The best performance is obtained when using feature set $6d$ - the (x, y) location, image intensity, and 4 octaves of Gabor filter responses.

3.13.3 Tensor Sparse Coding for Texture Classification

In this section we present experimental results on texture classification with the Brodatz dataset [Randen and Husøy, 1999]. We use the training images from the dataset which form the five 5-class, two 10-class, two 16-class, and three 2-class texture mosaics. Each texture class corresponds to one training image of 256×256 pixels, which is broken down into non-overlapping blocks of 32×32 pixels. A 5×5 covariance descriptor is then computed from each of these blocks, using the grayscale intensities and absolute values of the first- and second-order spatial derivatives, $\{I, |I_x|, |I_y|, |I_{xx}|, |I_{yy}|\}$.

There are 64 covariance descriptors from each texture class, of which 8 descriptors from each class are chosen for training, and the remaining are used for testing. The classification results are averaged over 20 random train-test splits, and are shown in Table 3.4.

Similar to the previous section, we also repeat the same experiments with the inverse covariances descriptors, and by relaxing the extra cone constraint. The best sparse coding-based approach performs competitively with the baseline KNN and SVM approaches.

Note that the KNN and SVM approaches have had their respective parameters optimized for best performance through cross-validation. Their accuracy varies quite drastically for different parameter choices. On the other hand, our method's classification performance does not vary substantially with λ. In fact, for a wide variation in the values of λ, the final classification performance does not change drastically (although the individual coefficients of sparse coding do). While increasing λ results in a poorer reconstruction $\hat{S}$, we are comparing the effect of different class dictionaries - the quality of approximation is decreased ($D_{\text{ld}}(\hat{S}_m, S)$ increases) for all classes $m = 1, \ldots, M$, leading to similar classification accuracies. This shows a certain robustness in our method with respect to the choice of parameter. Figure 3.14 shows how the accuracy varies with parameter choice for our method against the geodesic SVM for texture 12.

.

Table 3.4: Mean classification accuracy for the *Brodatz* mosaic dataset. Results are averaged are over 20 trials, and standard deviations are provided in parenthesis.

Mode	Covariances				Precisions				Geo-KNN	Geo-SVM
(n)	Separate (%)		Combined (%)		Separate (%)		Combined (%)		(%)	(%)
	1-cone	2-cone	1-cone	2-cone	1-cone	2-cone	1-cone	2-cone	$K = 1$	$\sigma = 0.6$
1 (5)	99.43 (0.63)	99.00 (0.62)	99.29 (0.41)	98.79 (0.74)	99.41 (0.62)	**99.45** (0.47)	99.18 (0.41)	98.84 (0.54)	98.88 (0.63)	99.14 (0.72)
2 (5)	93.13 (2.75)	91.66 (2.97)	86.09 (2.36)	84.89 (2.31)	**93.20** (2.61)	92.32 (2.87)	87.98 (2.47)	86.86 (2.50)	92.00 (2.27)	91.04 (2.31)
3 (5)	89.25 (2.47)	87.95 (2.55)	81.93 (2.59)	80.00 (3.04)	**89.32** (2.20)	87.86 (2.89)	82.54 (2.87)	82.11 (2.61)	87.21 (2.19)	88.79 (2.19)
4 (5)	85.36 (3.28)	84.05 (3.14)	83.41 (2.59)	82.05 (2.55)	85.64 (2.84)	84.05 (2.82)	83.66 (1.97)	82.39 (2.57)	92.55 (1.47)	**94.79** (1.38)
5 (5)	86.52 (1.83)	84.21 (2.48)	76.91 (3.26)	74.39 (3.38)	87.02 (1.50)	86.93 (1.99)	75.34 (2.51)	73.89 (2.61)	92.84 (1.48)	**94.55** (0.98)
6 (16)	**85.59** (1.02)	84.19 (0.84)	80.02 (1.15)	78.90 (1.05)	85.56 (1.08)	84.50 (1.36)	79.47 (0.97)	78.33 (1.11)	83.91 (0.98)	82.04 (1.98)
7 (16)	78.95 (1.52)	76.57 (1.30)	70.11 (0.99)	68.47 (1.35)	79.15 (1.35)	77.58 (1.67)	71.73 (1.32)	70.08 (1.47)	76.57 (1.34)	**80.18** (1.07)
8 (10)	87.71 (1.65)	86.13 (2.15)	84.81 (2.20)	83.79 (2.03)	87.48 (1.48)	86.59 (1.77)	84.40 (2.04)	83.46 (2.06)	**87.84** (1.48)	86.83 (3.94)
9 (10)	80.19 (1.88)	78.26 (1.69)	71.63 (1.84)	70.29 (2.84)	81.06 (1.83)	79.78 (1.97)	71.80 (2.19)	71.50 (2.15)	80.45 (2.08)	**82.21** (4.01)
10 (2)	99.87 (0.32)	99.87 (0.32)	99.91 (0.27)	99.82 (0.36)	99.96 (0.19)	**100.00** (0.00)	99.87 (0.32)	99.78 (0.56)	99.15 (0.82)	99.82 (0.36)
11 (2)	99.20 (1.23)	98.84 (1.41)	98.79 (1.17)	97.99 (1.46)	99.42 (1.07)	99.33 (1.26)	98.53 (1.50)	98.93 (1.40)	99.82 (0.36)	**100.00** (0.00)
12 (2)	98.30 (1.49)	96.43 (2.02)	96.34 (2.49)	94.33 (2.51)	98.62 (1.48)	99.06 (0.96)	98.13 (1.54)	98.79 (1.30)	**100.00** (0.00)	**100.00** (0.00)
Mean	90.29 %	88.93 %	85.77 %	84.48 %	**90.49** %	89.79 %	86.05 %	85.41 %	**90.94** %	**91.62** %

3.13.4 Action Recognition with Kinect Motion Capture

Since the time that the Microsoft Kinect sensor was introduced (end of 2010) there has been a great impact on many application areas of computer vision mainly due to its low cost and its augmented information content (depth information). We used the Kinect to obtain motion capture data of different individuals performing a set of actions. The OpenNI[6] platform and PrimeSense software were used to obtain motion information of tracked joints of the human body. The following 15 joints of the human skeleton were tracked: head, neck, torso, right shoulder, left shoulder, right elbow, left elbow, right hand, left hand, right hip, left hip, right knee, left knee, right foot, left foot. Data was obtained in our laboratory from 6 subjects, in 3 sessions of 10 actions each. This resulted in a total of 180 sequences. The actions studied were: jumping, boxing, jogging, left hand waving, right hand waving, both hands waving, jumping jack, shuffling, marching and clapping. Two of these actions

[6] http://www.openni.org/

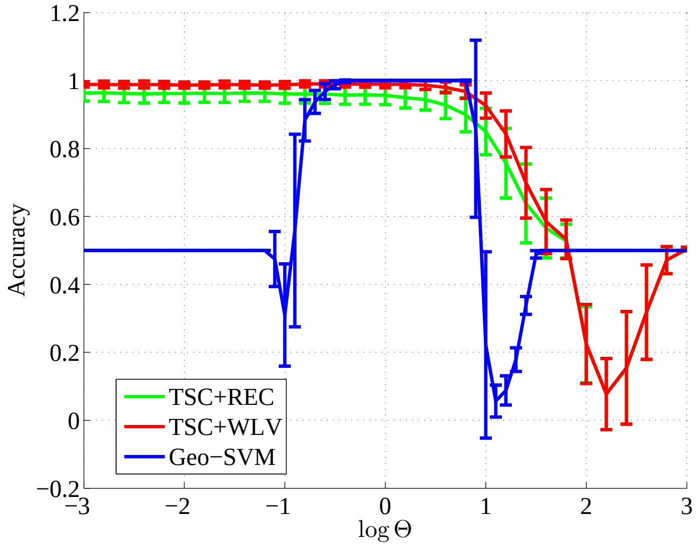


Figure 3.14: Variation in classification accuracy (texture 12) with parameter choice for tensor sparse coding and SVM approaches. The parameter $\log_{10}\Theta$ is varied along the x-axis, and $\Theta = \lambda$ for the tensor sparse coding approach and $\Theta = \sigma$ for the geodesic SVM classifier. The former approach shows a largely consistent high performance. 1σ standard deviation bars are also shown (Best viewed in color).

are shown in Figure 3.15.

The motion capture was acquired at a sampling rate of 30 frames per second. Each action was recorded for 10 seconds after the initial skeleton calibration was performed. From the tracker we obtain the (x, y, z) information for all 15, joints resulting in a 45-dimensional feature per sample, and then compute the 45×45 covariance of these features across each sequence.

Action recognition was performed using leave-person-out cross-validation (LPOCV), where in each iteration, we keep the data from 5 subjects for training and use the 6th subject for testing. We also performed leave-session-out cross-validation (LSOCV), where out of 18 sessions, 1 session was kept as test data and the remaining were used for training. The training covariances from each action class constitute the dictionary for that class, and a separate dictionary approach was used. The average cross-validation accuracy for both

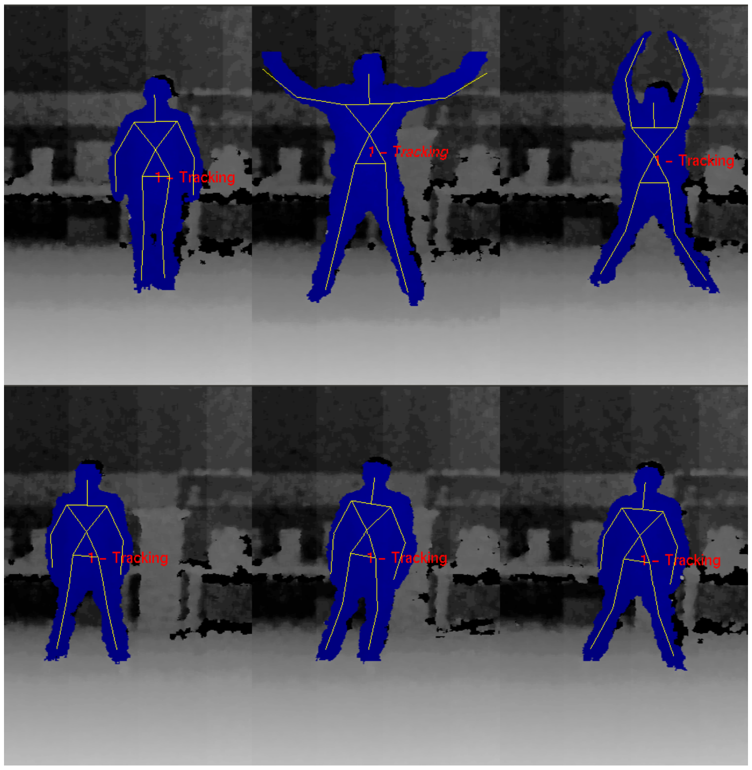


Figure 3.15: Screenshots from the OpenNI tracking program. Top row shows the *jumping jack* action and the bottom row shows the *shuffling* action.

K-nearest-neighbor and tensor sparse coding approaches are presented in Table 3.5.

Cross-validation	K-NN	TSC+REC
LPOCV	98.33%	97.78%
LSOCV	98.89%	98.89%

Table 3.5: Cross-validation accuracies over the Kinect motion capture dataset

Chapter 4

Positive Definite Dictionary Learning

Sparse coding methods transform a given signal into a set of sparse coefficients with the help of a dictionary or basis set. In the vector domain, pre-defined dictionaries are available which can be constructed using analytical expressions - for *e.g.*: Fourier, DCT, wavelets, etc. However, for applications involving only specific classes of signals, it is more interesting to use a domain-specific dictionary rather than universal dictionaries. Previous experiments in this thesis used the training data (or a subset thereof) to form the dictionary.

This chapter addresses the issue of learning a *data-driven* dictionary from a training set of positive definite matrices. The dictionary learning problem is formulated, analogous to similar approaches in vector dictionary learning. An alternating minimization approach to learn the dictionary is presented, and iterative gradient and Newton methods for updating the dictionary atoms are derived. When the dimensions of the data become too large, we propose an efficient matrix conjugate gradient approach to compute the Newton direction. A way to learn the dictionary in an online manner is also briefly explained. As a practical application, we apply the dictionary learning to learn face dictionaries, and use the learned model to detect faces. We also show the usefulness of covariance dictionaries to model tissue architecture and classify tissue image regions in surgical pathology.

4.1 Dictionary Learning Formulation

Given a training set $\mathcal{S} = \{S_j\}_{j=1}^N$, $S_j \in \mathbb{S}_{++}^n$, the problem of learning the dictionary $\mathcal{A} = \{A_i\}_{i=1}^K$, $A_i \in \mathbb{S}_{++}^n$ can be formulated as:

Dictionary Learning:

$$\min_{\mathcal{A},X} \quad \sum_{j=1}^{N} D_{\text{ld}}\left(\mathcal{A}\mathbf{x}_j, S_j\right) + \lambda \left\|\mathbf{x}_j\right\|_1 \tag{4.1a}$$

$$\text{s.t.} \quad \mathbf{x}_j \geq 0 \qquad \text{for } j = 1, \ldots, N \tag{4.1b}$$

$$A_i \succeq 0 \qquad \text{for } i = 1, \ldots, K \tag{4.1c}$$

$$\|A_i\|_F^2 \leq 1 \qquad \text{for } i = 1, \ldots, K \tag{4.1d}$$

$$\mathcal{A}\mathbf{x}_j \succeq 0 \qquad \text{for } j = 1, \ldots, N \tag{4.1e}$$

Here $\mathbf{x}_j$ denotes the j-th column of coefficient matrix X. As mentioned earlier in Section 3.6, the atoms should be normalized by their Frobenius norm. However, the constraint $\|A_i\|_F^2 = 1$ is non-convex, and therefore we relax the constraint to be convex $\|A_i\|_F^2 \leq 1$.

The dictionary learning problem (4.1) is non-convex in $(\mathcal{A}, X)$, and therefore there is no unique minimizer $(\mathcal{A}^*, X^*)$. However, the problem is convex in one argument given the other fixed, as is also the case in the vector dictionary learning problem. This naturally leads to an alternating minimization approach to arrive at a stationary point of the optimization problem.

4.2 Approach: Alternating Minimization

Similar to other dictionary learning algorithms [Aharon et al., 2006], we approach this problem through *alternating minimization*, repeating the following steps:

(a) Given $\mathcal{S}$ and $\mathcal{A}$ fixed, solve for X.

(b) Given $\mathcal{S}$ and X fixed, solve for $\mathcal{A}$.

Although this approach does not guarantee reaching a universal minimizer, we are guaranteed to reach a local minimum of the objective function in (4.1). The first step mentioned above is simply the sparse coding of the training set $\mathcal{S}$, which we will refer to as the *sparse coding step* of the dictionary learning procedure. The second step involves updating the dictionary atoms while keeping the sparse coefficients fixed, which we denote as the *dictionary update step*. The training data is sampled to initialize the dictionary $\mathcal{A}^0$.

Motivated by the K-SVD algorithm by [Aharon et al., 2006], the dictionary update is performed sequentially, updating one atom $A_i \in \mathcal{A}$ at a time, keeping the sparsity structure of X fixed, but allowing the corresponding non-zero coefficients of A_i to change in value. At iteration k of the dictionary learning procedure (denoted in the superscript), the atom A_i^{k-1} is updated to A_i^k, given $\left\{A_1^k, A_2^k, \ldots, A_{i-1}^k, A_{i+1}^{k-1}, \ldots A_K^{k-1}\right\}$ and X^k.

4.3 Atom Update

In this section, we present the optimization subroutine to update atom A_i in the dictionary update step. Let ω_i be the *active set*, $\omega_i = \{j | j \subseteq \{1, \ldots, N\}, x_{ij} \neq 0\}$, *i.e.*, the subset of signals which use atom A_i.

The reconstruction $\hat{S}_j$ of each S_j, $j \in \omega_i$ can be decomposed into the constant and variable components under the optimization of A_i:

$$\hat{S}_j = \sum_{i' \neq i} x_{i'j} A_{i'} + x_{ij} A_i = \hat{S}_j^{(i)} + x_{ij} A_i. \tag{4.2}$$

$\hat{S}_j^{(i)}$ is the reconstruction of S_j without the contribution of A_i.

The sub-problem of (4.1) to optimize atom A_i keeping all other atoms fixed, is given by:

$$\min_{A_i \succeq 0} \quad \sum_{j \in \omega_i} D_{\text{ld}} \left(\sum_{i' \neq i} x_{i'j} A_{i'} + x_{ij} A_i, S_j \right) \tag{4.3}$$

Expanding Equation (4.3):

$$\min_{A_i \succeq 0} \quad \sum_{j \in \omega_i} \left[\operatorname{tr} \left(\left(\sum_{i' \neq i} x_{i'j} A_{i'} + x_{ij} A_i \right) S_j^{-1} \right) - \log\det \left(\left(\sum_{i' \neq i} x_{i'j} A_{i'} + x_{ij} A_i \right) S_j^{-1} \right) \right]$$

$$\min_{A_i \succeq 0} \quad \sum_{j \in \omega_i} \left[\sum_{i' \neq i} x_{i'j} \operatorname{tr} \left(A_{i'} S_j^{-1} \right) + x_{ij} \operatorname{tr} \left(A_i S_j^{-1} \right) - \log\det \left(\sum_{i' \neq i} x_{i'j} A_{i'} + x_{ij} A_i \right) + \log\det S_j \right]$$

Retaining only the terms relevant to A_i, we get:

$$\min_{A_i \succeq 0} \quad \sum_{j \in \omega_i} \left[x_{ij} \operatorname{tr} \left(A_i S_j^{-1} \right) - \log\det \left(\sum_{i' \neq i} x_{i'j} A_{i'} + x_{ij} A_i \right) \right] \tag{4.4}$$

Let the objective function for the atom update sub-problem be denoted as $f(A_i)$.

$$f(A_i) = \sum_{j \in \omega_i} \left[x_{ij} \operatorname{tr} \left(A_i S_j^{-1} \right) - \log\det \left(\sum_{i' \neq i} x_{i'j} A_{i'} + x_{ij} A_i \right) \right] \tag{4.5}$$

Taking the gradient of Equation (4.5) w.r.t. A_i,

$$\nabla f(A_i) = \sum_{j \in \omega_i} x_{ij} S_j^{-1} - x_{ij} \left(\hat{S}_j^{(i)} + x_{ij} A_i \right)^{-1}. \tag{4.6}$$

There is no closed form solution to the equation $\nabla f(A_i) = 0$, and therefore we resort to iterative descent methods such as gradient descent and Newton descent.

Algorithm 2 Dictionary Learning

```
Input: Data $\mathcal{S} = \{S_j\}_{j=1}^{N}$, dictionary size $K$, sparsity parameter $\lambda$
Output: $\mathcal{A} = \{A_i\}_{i=1}^{K}$
  $k = 0$
  Initialize $\mathcal{A}^0$ sampled from $\mathcal{S}$
  repeat
    $k \leftarrow k + 1$
    Given $\mathcal{S}$ and $\mathcal{A}^{k-1}$, compute the sparse coefficients $X^k$
    for $i = 1$ to $K$ do
      Update atom $A_i^{k-1}$ to $A_i^k$, along with the corresponding coefficients in $X^k$ (Algorithm 3)
    end for
  until convergence
```

Algorithm 3 Atom Update

```
Input: $A_i$, $\left\{x_{ij}, S_j, \hat{S}_j \mid j \in \omega_i\right\}$
Output: $A_i$, $\left\{x_{ij}, \hat{S}_j \mid j \in \omega_i\right\}$
  repeat
    Compute descent direction $\Delta A_i$ using (4.8) or (4.13)
    Choose stepsize $\alpha$ by line search s.t. $A_i + \alpha \Delta A_i \succeq 0$
    $A_i^{new} \leftarrow A_i + \alpha \Delta A_i$
    $\hat{S}_j \leftarrow \hat{S}_j + x_{ij} \left(A_i^{new} - A_i\right) \quad \forall j \in \omega_i$
    $t = \max\left\{\|A_i^{new}\|_F, 1\right\}$
    $A_i \leftarrow A_i^{new}/t$
    $x_{ij} \leftarrow t\ x_{ij} \quad \forall j \in \omega_i$
  until convergence
```

4.3.1 Gradient Descent

The gradient of the objective (4.5) is given by:

$$\nabla f(A_i) = \sum_{j \in \omega_i} x_{ij} \left(S_j^{-1} - \hat{S}_j^{-1} \right). \tag{4.7}$$

The gradient descent direction ΔA_i^g is given by the negative of the gradient:

$$\Delta A_i^g = \sum_{j \in \omega_i} x_{ij} \left(\hat{S}_j^{-1} - S_j^{-1} \right). \tag{4.8}$$

The gradient descent update algorithm is, therefore,

$$A_i^k \leftarrow A_i^{k-1} + \alpha \Delta A_i^g \quad \text{s.t. } A_i^k \succeq 0, \tag{4.9}$$

with stepsize $\alpha \geq 0$ determined using line search techniques. The stepsize should also satisfy the constraint that the updated atom A_i^k is positive semi-definite.

Two possibilities are to use the the *exact* line search or the *backtracking* (*Armijo*) line search. In practice, we see that these two methods do not provide much improvement in the objective function in each atom update iteration. Instead, we use the *Barzilai-Borwein* (BB) step sizes [Barzilai and Borwein, 1988]:

$$\alpha_{BB1}^k = \frac{\left\langle A_i^k - A_i^{k-1}, \nabla f\left(A_i^k\right) - \nabla f\left(A_i^{k-1}\right) \right\rangle}{\left\| \nabla f\left(A_i^k\right) - \nabla f\left(A_i^{k-1}\right) \right\|_F^2}, \tag{4.10}$$

$$\alpha_{BB2}^k = \frac{\left\| A_i^k - A_i^{k-1} \right\|_F^2}{\left\langle A_i^k - A_i^{k-1}, \nabla f\left(A_i^k\right) - \nabla f\left(A_i^{k-1}\right) \right\rangle}, \tag{4.11}$$

for iteration k. The BB stepsize choice yields a much stronger net decrease in the objective function value compared to exact or backtracking line searches.

4.3.2 Newton Descent

Taking the second derivative of the gradient (4.7), we get the expression for the Hessian:

$$\nabla^2 f(A_i) = \sum_{j\in\omega_i} \left[x_{ij} \left(x_{ij} A_i + \hat{S}_j^{(i)} \right)^{-1} \otimes x_{ij} \left(x_{ij} A_i + \hat{S}_j^{(i)} \right)^{-1} \right]$$

$$\therefore \nabla^2 f(A_i) = \sum_{j\in\omega_i} \left(x_{ij} \hat{S}_j^{-1} \right) \otimes \left(x_{ij} \hat{S}_j^{-1} \right) \tag{4.12}$$

The Newton descent direction ΔA_i^N is obtained by solving:

$$\nabla^2 f(A_i)\ \Delta A_i^N = -\nabla f(A_i)$$

$$\sum_{j\in\omega_i} x_{ij}^2 \hat{S}_j^{-1} \Delta A_i^N \hat{S}_j^{-1} = \sum_{j\in\omega_i} x_{ij} \left(\hat{S}_j^{-1} - S_j^{-1} \right) \tag{4.13}$$

The Newton descent update algorithm is, therefore,

$$A_i^k \leftarrow A_i^{k-1} + \alpha \Delta A_i^N \quad \text{s.t. } A_i^k \succeq 0, \tag{4.14}$$

with stepsize $\alpha \geq 0$.

The Newton direction computation involves solving an $n^2 \times n^2$ system of linear equations, given by:

$$\underbrace{\sum_{j\in\omega_i} \left(x_{ij} \hat{S}_j^{-1} \right) \otimes \left(x_{ij} \hat{S}_j^{-1} \right)}_{n^2\times n^2} \text{vec}\left(\Delta A_i^N \right) = \text{vec}\left(\sum_{j\in\omega_i} x_{ij} \left(\hat{S}_j^{-1} - S_j^{-1} \right) \right). \tag{4.15}$$

Let us denote this positive definite system as $A\mathbf{x} = \mathbf{b}$, with $A = \sum_{j\in\omega_i} \left(x_{ij} \hat{S}_j^{-1} \right) \otimes \left(x_{ij} \hat{S}_j^{-1} \right)$, $\mathbf{x} = \text{vec}\left(\Delta A_i^N \right)$, and $\mathbf{b} = \text{vec}\left(\sum_{j\in\omega_i} x_{ij} \left(\hat{S}_j^{-1} - S_j^{-1} \right) \right)$.

Explicitly forming A and solving the system is an expensive operation, even with decomposition methods. The cost of directly solving for the Newton direction has a cost of $\mathcal{O}\left(n^6\right)$, where n denotes the dimension of the dictionary atoms. In most of our applications pertaining to region covariance descriptors, n is very small ($\sim 5 - 10$), and therefore this is still acceptable in practice.

When n is much larger, we can take advantage of the fact that although solving for $A\mathbf{x} = \mathbf{b}$ with an explicit A is expensive, it is relatively inexpensive to apply the operator A on a given $\mathbf{x}$. This is due to the fact that A is composed of a sum of Kronecker products. This enables us to use iterative methods like conjugate gradient to directly solve Equation (4.13).

4.3.3 Comparison of Atom Update Techniques

We compare the gradient and Newton atom update techniques in terms of their effectiveness in optimizing the dictionary learning objective function. A set of K $n \times n$ positive definite atoms $\mathcal{A}_0$ were synthesized ($n = 3$). N k-sparse vectors $\{\mathbf{x}_j\}_{j=1}^{N}$ with $k < n$ were sampled, and signals $\mathcal{S} = \{S_1, \ldots, S_N\}$ were constructed. The number of signals generated was proportional to the dimensions and dictionary size $N = 4n^2K$. The dictionary learning was run for a maximum of 25 iterations, and the net reduction in the objective function was compared.

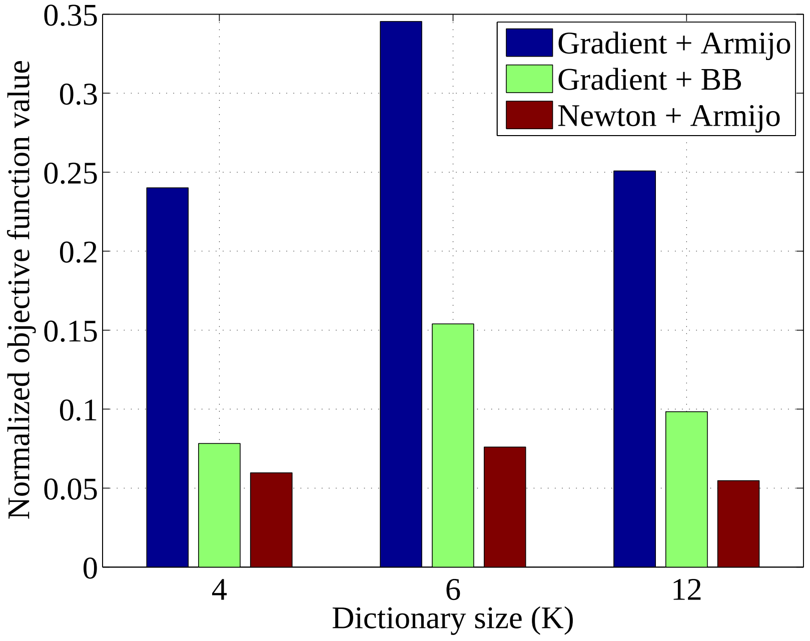
We used dictionary sizes of $K = 4, 6$, and 12, and repeated each (n, K) combination for 10 random trials. The choice of K covers three different scenarios - $K < M$, $K = M$, and $K > M$, where $M = n(n+1)/2$ - *undercomplete*, *complete*, and *overcomplete* cases.

Three atom update techniques were compared:

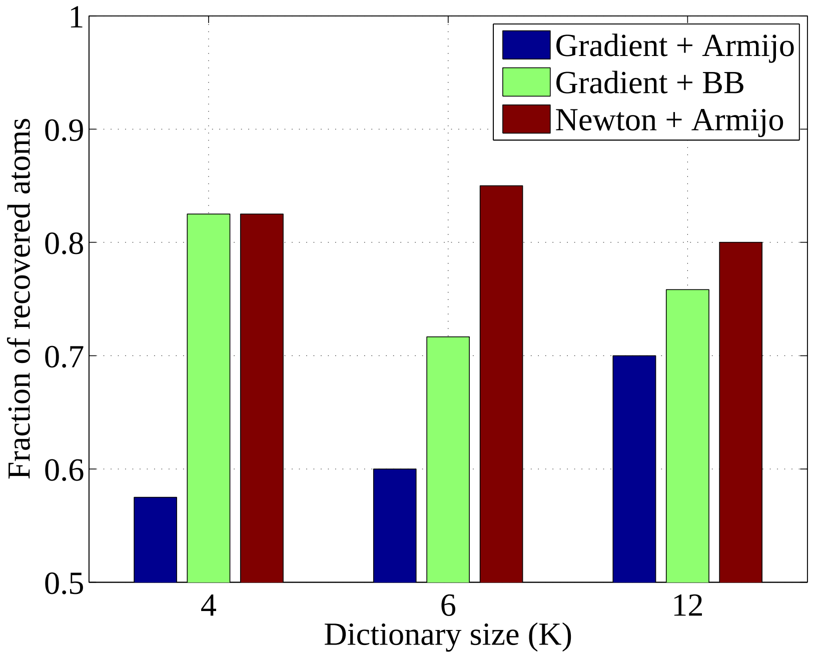
1. gradient descent with backtracking line search
2. gradient descent with Barzilai-Borwein stepsize (4.11)
3. Newton descent

The different techniques were initialized with the same random dictionary. The objective function values $f(\hat{\mathcal{A}})$ at the end of the learning procedure relative to the initial objective $f(\mathcal{A}^0)$ are estimated as an indicator of the quality of the local minimum attained in each learning procedure. This is shown in Figure 4.1(a). The Newton update method performs the best, as is expected, but the BB stepsize method greatly improves upon the gradient descent with backtracking line search.

We also test the number of atoms correctly recovered in the learned dictionary in each update technique. The learned atoms $\hat{A}_i$ are matched with the ground truth atoms A_j^* using a coherence threshold of $\mu = \operatorname{tr}\left(\hat{A}_i, A_j^*\right) \geq 0.95$. The Newton dictionary update approach does very well in recovering the dictionary atoms, and the Gradient+BB method performing equally well when K is small.



(a)



(b)

Figure 4.1: Comparison of atom update methods: (a) Objective function values $f(\hat{\mathcal{A}})$ at the end of the learning procedure relative to the initial objective $f(\mathcal{A}^0)$ and (b) Fraction of recovered atoms with a coherence threshold of $\mu_{\min} = 0.95$. Size of the dictionary K is varied in $\{4, 6, 12\}$ and the results are averaged over 10 different random trials.

4.3.4 Matrix Conjugate Gradient

In this section, we present a conjugate gradient method to directly solve (4.13) for the Newton descent direction. Writing the general form[1] of Equation (4.13),

$$\sum_{i=1}^{M} A_i X A_i^T = B. \tag{4.16}$$

The matrix conjugate gradient algorithm to iteratively solve Equation (4.16) for X is given in Algorithm 4.

We compare the direct inversion approach and the matrix conjugate gradient approach for computing the Newton direction during actual dictionary update iterations for synthetic datasets of varying dimensions n. The matrix conjugate gradient is implemented in MATLAB without any further code optimization. The time taken to explicitly construct A is also included in the computation time of the direct approach. The dimensions n is varied from 5 to 50 in steps of 5. The computation times for dimension n are averaged over $25n$ trials. In all comparisons, the returned solution from the conjugate gradient method X^{cg} is within 10^{-5} relative error of the direct solution X^*.

The average speedup obtained by using the matrix conjugate gradient algorithm over the direct inversion method is presented in Figure 4.2, along with 1σ standard deviation bars. The horizontal line at speedup of 1 shows the cross-over point when the conjugate gradient method overtakes the direct inversion approach in computation time. For $n \leq 15$, it is faster to directly solve for $\mathbf{x}$ than using iterative methods. For $n \geq 20$, the matrix conjugate gradient method gives significant speedups in solving systems of the presented structure. The speedup reported was obtained on the same machine used in Section 3.12.

As mentioned earlier, the code was not optimized at the low-level - for *e.g.*, exploiting the symmetry of all involved matrices. This optimization can yield improved computation times in practice, but does not change the order complexity of the algorithms themselves.

[1] The notations A_i, X, B in this section are different from the variables of the dictionary learning problem.

Algorithm 4 Matrix Conjugate Gradient

```
Input: $\{A_i\}_{i=1}^{M}$, $B$
Output: $X^*$
  $X_0 = 0_{n\times n}$
  $R_0 = B - \sum_{i=1}^{M} A_i X_0 A_i^T$
  $P_0 = R_0$
  $k = 0$
  repeat
    $\alpha_k = \frac{\langle\ R_k\ ,\ R_k\ \rangle}{\left\langle\ P_k\ ,\ \sum_{i=1}^{M} A_i P_k A_i^T\ \right\rangle}$
    $X_{k+1} = X_k + \alpha_k P_k$
    $R_{k+1} = R_k - \alpha_k \sum_{i=1}^{M} A_i P_k A_i^T$
    $\beta_k = \frac{\langle\ R_{k+1}\ ,\ R_{k+1}\ \rangle}{\langle\ R_k\ ,\ R_k\ \rangle}$
    $P_{k+1} = R_{k+1} + \beta_k P_k$
    $k \leftarrow k + 1$
  until convergence
  $X^* = X_{k+1}$
```

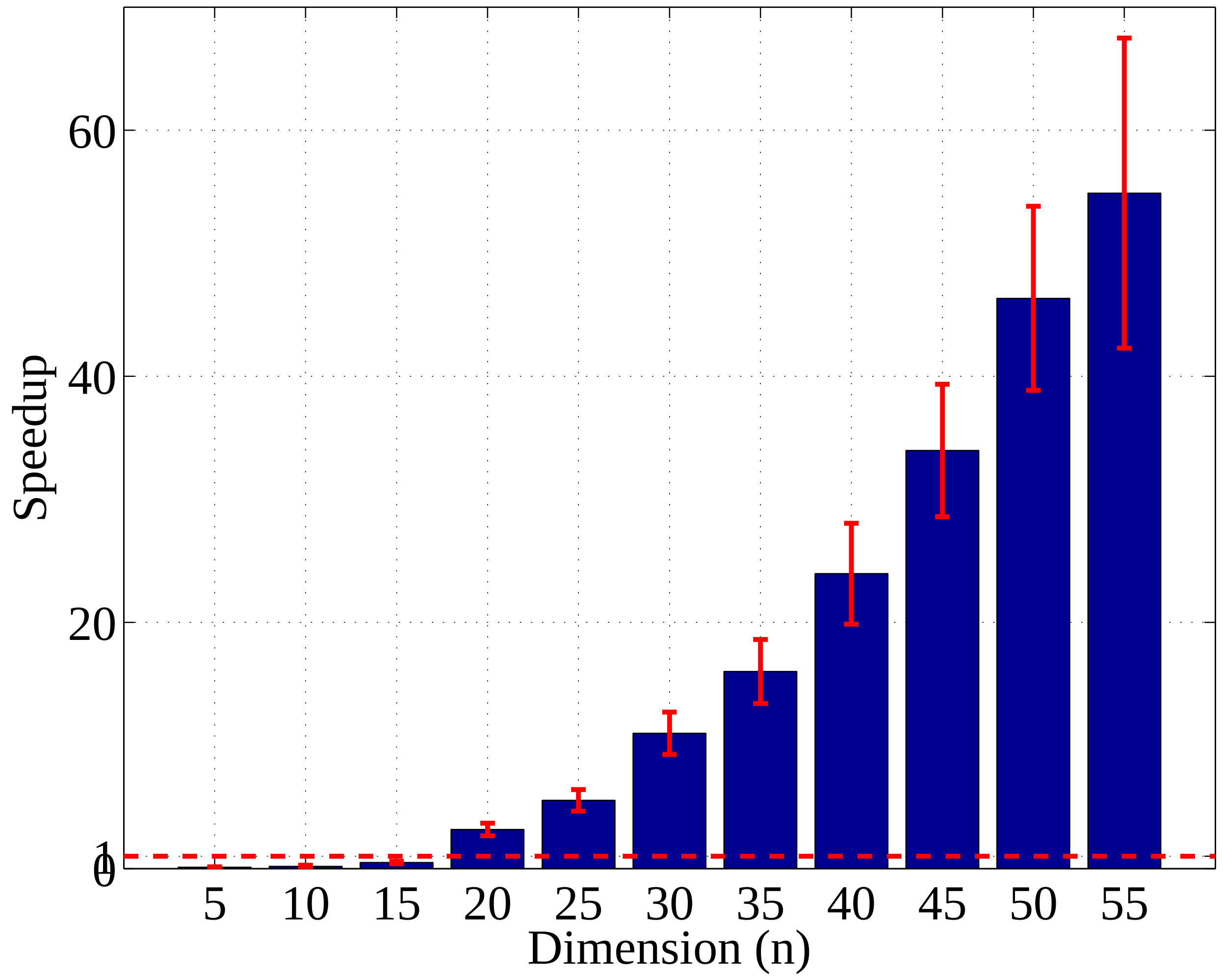


Figure 4.2: Average speedup of matrix conjugate gradient vs. direct $A\mathbf{x} = \mathbf{b}$ linear system solution for computation of the Newton descent direction. 1σ bars are also shown.

4.4 Online Dictionary Learning

The dictionary learning approach can be extended to an online learning setting, using stochastic gradient descent [Bottou, 1998]. Suppose at time t we get a new data point S_t, which sparse coded over the existing dictionary $\mathcal{A}_{t-1}$ results in the reconstruction $\hat{S}_t$, with coefficients $x_{i,t}$ for $i = 1, \ldots, K$. The atoms which are used $\{i \mid x_{i,t} > 0\}$ can be updated sequentially, and the rest of the atoms are left unchanged.

The online update using only the gradient information pertaining to the new data point can be written as:

$$A_{i,t} \leftarrow A_{i,t-1} - \alpha_t x_{i,t} \left(S_t^{-1} - \hat{S}_t^{-1} \right), \tag{4.17}$$

for a diminishing stepsize α_t, When the data comes in batches, *i.e.*, at each step t, we get a set of points $\mathcal{S}_t$ of size N_t, and their coefficients X_t are obtained by sparse coding over the current dictionary $\mathcal{A}_{t-1}$. The relevant atoms in $i \in \{1, \ldots, K\}$ are updated as:

$$A_{i,t} \leftarrow A_{i,t-1} - \alpha_t \sum_{j=1}^{N_t} x_{ij,t} \left(S_{j,t}^{-1} - \hat{S}_{j,t}^{-1} \right). \tag{4.18}$$

Using the second order information from the cost function is known to improve the online dictionary update performance [Bottou, 1998], and so we can include the online Hessian estimate for each atom i separately, accumulating over iterations $t' = 1, \ldots, t$.

$$H_{i,t} = \sum_{t'=1}^{t} \left[\sum_{j=1}^{N_{t'}} \left(x_{ij,t'} \hat{S}_{j,t}^{-1} \right) \otimes \left(x_{ij,t'} \hat{S}_{j,t}^{-1} \right) \right]. \tag{4.19}$$

$$G_{i,t} = \sum_{j=1}^{N_t} x_{ij,t} \left(S_{j,t}^{-1} - \hat{S}_{j,t}^{-1} \right). \tag{4.20}$$

$$A_{i,t} \leftarrow A_{i,t-1} - H_{i,t}^{-1} G_{i,t} \tag{4.21}$$

4.5 Time Complexity

4.5.1 Sparse Coding

When the upper cone constraint is relaed, the MAXDET problem has a time complexity of $\mathcal{O}(K^2n^2)$ per Newton iteration [Vandenberghe et al., 1998], with a worst-case complexity of $\mathcal{O}(\sqrt{n})$ Newton iterations at each step in the interior point algorithm. Our coordinate descent approach has a time complexity of $\mathcal{O}(Kn^3)$ per iteration over all the coordinates. The n^3 term comes from the computation of matrix inverses, generalized eigenvalues and their sums. Since n is pretty small in most of our applications, and the size of the dictionary is large, our specialized approach and implementation yields faster run times.

4.5.2 Dictionary Learning

The dictionary learning approach has a time complexity of $\mathcal{O}(n^3 L_{\max})$ per atom update for the gradient descent methods, and the Newton descent has a time complexity of $\mathcal{O}(n^6 L_{\max})$ since it involves solving an $n^2 \times n^2$ system of equations. $L_{\max}$ denotes the maximum number of inner iterations within each atom update step (Usually this is small in practice: ≤ 5 for initial iterations and just 1 or 2 for later iterations). The rest of the computation in each atom update step is subsumed by complexity of computing the descent direction.

4.6 Face Detection with Tensor Dictionary Learning

We apply the positive definite dictionary learning approach to detect faces in images, from the GRAZ01 *person* dataset [Opelt et al., 2004]. We trained the dictionaries on the face images of 109 people from the grayscale FERET database [Phillips et al., 2000], used in Section 3.13.2. We trained the dictionary using 7 images from each subject, for a total 763 training covariance matrices. The feature mode 6*b*) from Section 3.13.2, consisting of the pixel coordinates (x, y), image intensity I and two octaves of the Gabor filter features $g_{00}, \ldots, g_{07}, g_{10}, \ldots, g_{17}$. The resulting covariance descriptors are of dimension $n = 19$. A dictionary of size $K = 2n = 38$ atoms was learned, with $\lambda = 0.1$.

A sample set of 10 face images from the GRAZ01 dataset was used to test the face detection capability of this learned positive definite dictionary. A sliding window of a single scale of size 60×60 pixels, and stepsize 15 pixels was used, and the Gabor covariances were computed. The reconstruction error D_{ld} from sparse coding these descriptors over the face dictionary was used to compute the face score of each block.

$$\text{score} = \exp\left(-\frac{D_{\text{ld}}^2(\hat{S}, S)}{2\sigma^2}\right).$$

The parameter σ was computed as the standard deviation over all the reconstruction errors in the test image. The score is then normalized to be in the range $[0, 1]$, and smoothed with a Gaussian filter. We show the detection score images along with the original images in Figure 4.3. The ground truth bounding boxes are marked manually, but the detections (true positives, false positives, etc.) are computed pixel-wise. The precision-recall curve averaged over these 10 images is also shown.

This experiment shows the usefulness of learning positive definite matrix dictionaries for computer vision applications.

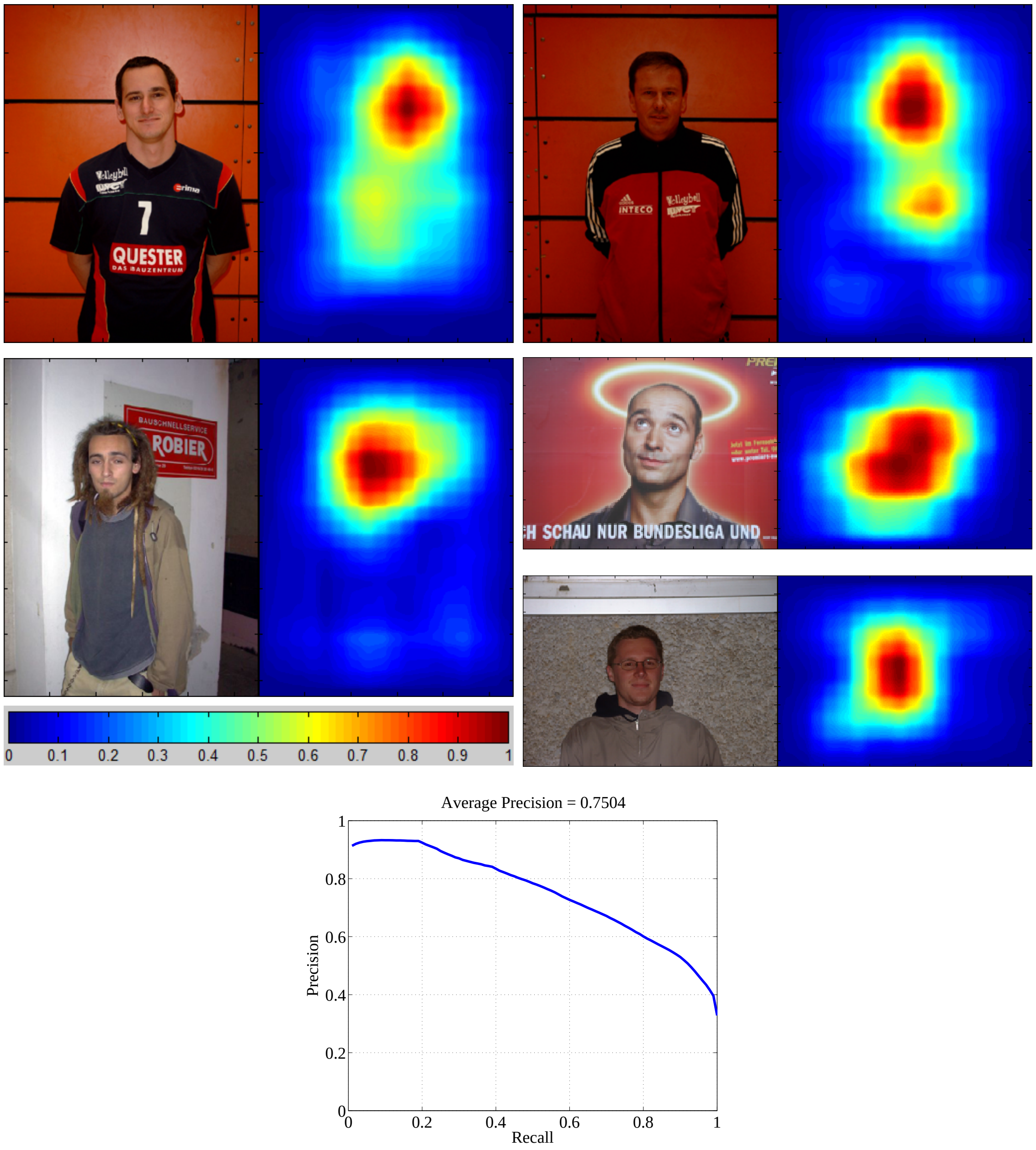


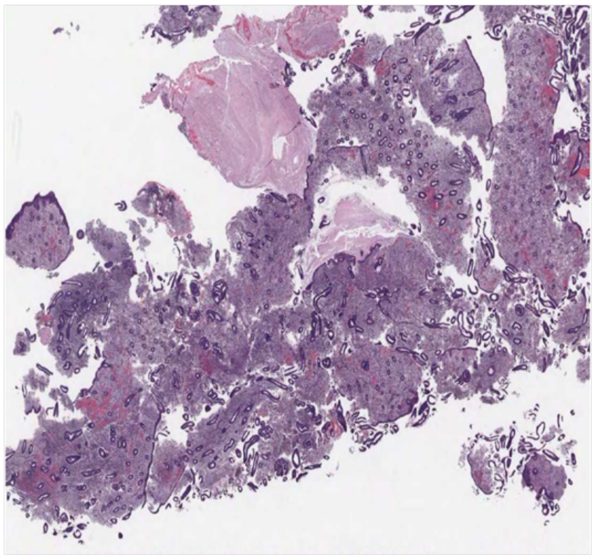
Figure 4.3: Face detection results on sample images from the GRAZ01 *person* dataset. Images and corresponding their detection scores are shown. The pixel-level precision-recall curve over the test set is also shown.

4.7 Cancer Tissue Classification with Positive Definite Dictionaries

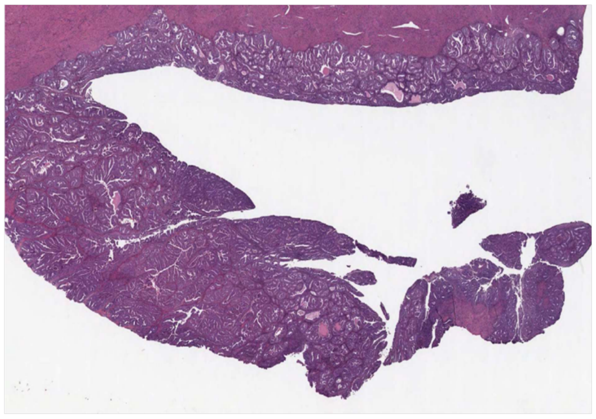
Early diagnosis of any disease is quintessential for effective treatment. This is no less true in the diagnosis and treatment of cancer. For a surgical pathologist, the most time-consuming aspect of the diagnostic process involves arduously scrutinizing tissue slides under a microscope for the evidence of disease. As a result, even a skilled pathologist is able to diagnose only a few patients every day. However, it is possible to expedite this process through computer-assisted diagnosis. Towards this end, we apply the positive definite dictionary learning algorithms towards classification of tissue image regions as cancerous or benign. We use region covariance descriptors to characterize the image blocks extracted from the tissue, since the distinction between the two classes of healthy vs. cancerous tissue is based on the architecture or texture. Our work on using region covariances for this classification, along with vector sparse dictionary learning, has been published earlier in [Sivalingam et al., 2011] and [Sivalingam et al., 2012] where we deal with endometrial and prostate cancer tissue images respectively.

We show results with positive definite dictionary learning with the endometrial tissue images from [Sivalingam et al., 2011]. Sample images from the healthy and endometrioid carcinoma tissue classes, the description of the covariance features used are shown in Figure 4.4. A combination of spatial and intensity features, with a block size of 200x200 pixels at 5x resolution was seen to give the best performance in [Sivalingam et al., 2011]. We choose 4 images each from the healthy and carcinoma classes, and sample 200 blocks from each image. We use this set of 1600 covariance descriptors and perform 4-fold cross-validation - using 1 image from each class for testing and keeping the remaining 3 for training. An ISOMAP embedding [Tenenbaum et al., 2000] showing the covariance descriptors from the two classes is also shown in Figure 4.4.

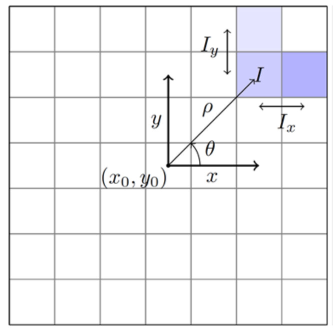
In each fold, we use the 8×8 training covariances to learn tensor dictionaries of varying



(a)

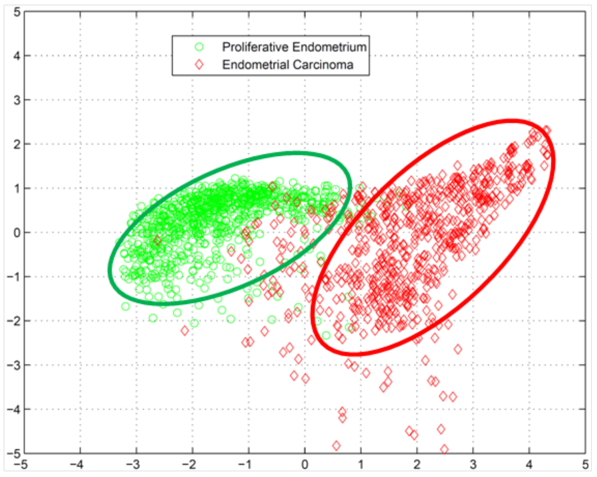


(b)



$$\mathbf{x} = \begin{bmatrix} I \\ I_x \\ I_y \\ \sqrt{I_x^2 + I_y^2} \\ x \\ y \\ \rho \\ \theta \end{bmatrix}$$

(c)



(d)

Figure 4.4: Samples from the healthy (top left) and cancerous (top right) images. The covariance feature used is described on the bottom row, and an ISOMAP embedding depicting the covariance descriptors from the two classes is also shown.

sizes, and classify the test features by the usual least reconstruction error approach. The parameter λ was set to 0.001. We compare with learned dictionaries constructed by randomly sampling the training data. We also compare the performance with a baseline K-NN classifier (with $K = 5$ chosen by cross-validation). The results are shown in Table 4.1 for different values of the dictionary size K.

Algorithm		Accuracy	
5-NN		94.31 %	
Tensor Dictionaries	K	Random	Learned
	4	90.75 %	92.75 %
	8	92.25 %	93.25 %
	16	92.44 %	93.31 %
	20	93.13 %	93.88 %
	28	93.88 %	94.63 %
	32	94.50 %	**95.38** %

Table 4.1: Average classification accuracy with 4-fold cross-validation between healthy and endometrioid cancer tissue image patches.

The dictionary learning procedure helps in improving the accuracy of dictionary-based classification, compared to randomly choosing data points for the model. We beat the baseline K-nearest-neighbor classification, while just maintaining only a few atoms derived from the data - a 32-atom dictionary stores only about 5% of the number of matrices as the K-NN classifier.

Chapter 5

Rank-One Tensor Dictionaries

The sparse coding and dictionary learning algorithms for positive definite matrices presented so far in this thesis only require that the signal S and the reconstruction $\hat{S}$ be positive definite. They do not require the dictionary atoms A_i, $i = 1, \ldots, K$ to be strictly positive definite. Hence it is possible to use rank-deficient positive semi-definite matrices as atoms in the dictionary $\mathcal{A}$.

In this chapter, we deal with the extreme case when all the dictionary atoms are formed by rank-1 outer products, *i.e.*, $A_i = \mathbf{v}_i\mathbf{v}_i^T$ for $i = 1, \ldots, K$. While the structure of the atoms can be exploited in the sparse coding and dictionary learning algorithms for semidefinite atoms of any rank, the rank-1 case is tackled here in detail since we can reduce all of the computations in terms of the dictionary vectors $\mathbf{v}_i$. We present efficient versions of the sparse coding and dictionary learning algorithms in this chapter, tailored to this special case.

The time complexity of the specialized sparse coding and dictionary update algorithms are also discussed, along with timing experiments with our C++ implementation. A synthetic experiment shows the ability of the proposed approach to recover ground truth atoms from which the data was generated. As a practical application, we handle digit classification with the USPS dataset, showing the performance of rank-1 dictionaries.

Instead of representing the dictionary as the set $\mathcal{A} = \{A_i\}_{i=1}^K$, we can concisely represent it by an $n \times K$ matrix $V = [\ \mathbf{v}_1 \mid \mathbf{v}_2 \mid\ \ldots\ \mid \mathbf{v}_K\ ]$. In order to maintain the atom normalization constraint $\|A_i\|_F^2 = 1$, and noting that $\|A_i\|_F = \operatorname{tr}\left(\mathbf{v}_i \mathbf{v}_i^T\right) = \mathbf{v}_i^T \mathbf{v}_i$, we set the vectors $\mathbf{v}_i$ to have unit ℓ_2-norm $\|\mathbf{v}_i\|_2^2 = 1$. Therefore, the columns of V are constrained to be unit length.

In the rest of this work, we will interchangeably use the term *rank-1 dictionaries* to imply tensor dictionaries whose atoms are rank-1 positive semidefinite matrices.

5.1 Efficient Sparse Coding over Rank-1 Dictionaries

We extend the cyclic coordinate descent approach presented in Section 3.12 to the special case of rank-1 dictionaries in this section. The sparse coding problem over rank-1 dictionaries is given by:

$$\min_{\mathbf{x}\geq 0} \quad \operatorname{tr}\left(\sum_{i=1}^{K} x_i \mathbf{v}_i \mathbf{v}_i^T S^{-1}\right) - \log\det\left(\sum_{i=1}^{K} x_i \mathbf{v}_i \mathbf{v}_i^T S^{-1}\right) + \lambda \sum_{i=1}^{K} x_i \tag{5.1a}$$

$$\text{s.t.} \quad \sum_{i=1}^{K} x_i \mathbf{v}_i \mathbf{v}_i^T \succeq 0. \tag{5.1b}$$

The objective function $f(\mathbf{x})$ can be simplified as:

$$\begin{aligned} f(\mathbf{x}) &= \sum_{i=1}^{K} x_i \operatorname{tr}\left(\mathbf{v}_i \mathbf{v}_i^T S^{-1}\right) - \log\det\left(\sum_{i=1}^{K} x_i \mathbf{v}_i \mathbf{v}_i^T\right) + \log\det S + \lambda \sum_{i=1}^{K} x_i \\ &= \sum_{i=1}^{K} x_i \mathbf{v}_i^T S^{-1} \mathbf{v}_i - \log\det\left(\sum_{i=1}^{K} x_i \mathbf{v}_i \mathbf{v}_i^T\right) + \lambda \sum_{i=1}^{K} x_i + \text{terms independent of } \mathbf{x}. \end{aligned}$$

Setting $c_i = \mathbf{v}_i^T S^{-1} \mathbf{v}_i$ with $\mathbf{c} = [c_1, \ldots, c_K]^T$ and plugging the final expression back into the objective, we have:

$$\min_{\mathbf{x}\geq 0} \quad \mathbf{c}^T\mathbf{x} - \log\det\left(\sum_{i=1}^{K} x_i \mathbf{v}_i \mathbf{v}_i^T\right) + \lambda \sum_{i=1}^{K} x_i \tag{5.2a}$$

$$\text{s.t.} \quad \sum_{i=1}^{K} x_i \mathbf{v}_i \mathbf{v}_i^T \succeq 0. \tag{5.2b}$$

The gradient of the objective $f(\mathbf{x})$ from (5.2), with respect to the coordinate x_k being updated, is given by:

$$\frac{\partial f(\mathbf{x})}{\partial x_k} = c_k + \lambda - \operatorname{tr}\left(\left(\sum_{i=1}^{K} x_i \mathbf{v}_i \mathbf{v}_i^T\right)^{-1} \mathbf{v}_k \mathbf{v}_k^T\right)$$

$$\therefore \frac{\partial f(\mathbf{x})}{\partial x_k} = c_k + \lambda - \mathbf{v}_k^T \hat{S}^{-1} \mathbf{v}_k.$$

The gradient descent direction is therefore given by the negative of the gradient:

$$\delta x_k = -\frac{\partial f(\mathbf{x})}{\partial x_k} = \mathbf{v}_k^T \hat{S}^{-1} \mathbf{v}_k - c_k - \lambda. \tag{5.3}$$

This is a one-dimensional minimization problem along the coordinate x_k, and so the sign of δx_k is sufficient to identify the descent direction - *i.e.*, whether we should proceed along increasing x_k or decreasing x_k. The actual magnitude of the gradient descent step can be absorbed into the stepsize β (which we can find using exact line search). Therefore, the descent direction is given by:

$$\delta x_k = \text{sign}\left(\mathbf{v}_k^T \hat{S}^{-1} \mathbf{v}_k - c_k - \lambda\right), \tag{5.4}$$

and the coordinate update is given by:

$$x_k \leftarrow x_k + \beta \delta x_k. \tag{5.5}$$

Due to the nature of the update expression in the coordinate descent algorithm, the current values of $\mathbf{v}_k^T S^{-1} \mathbf{v}_k$ and $\mathbf{v}_k^T \hat{S}^{-1} \mathbf{v}_k$ are required at each iteration. While the former is constant and is already computed ($= c_k$), the later has to be updated continuously since $\hat{S}$ is a function of the current iterate of $\mathbf{x}$: $\hat{S}(\mathbf{x}) = \mathcal{A}\mathbf{x}$.

In the inner loop of the coordinate descent algorithm, when updating x_k, the reconstruction $\hat{S}$ is locally updated as:

$$\hat{S} \leftarrow \hat{S} + \left(x_k^{\text{new}} - x_k^{\text{old}}\right) A_k. \tag{5.6}$$

Replacing $A_k = \mathbf{v}_k \mathbf{v}_k^T$ and letting $\alpha = x_k^{\text{new}} - x_k^{\text{old}}$, the update becomes

$$\hat{S} \leftarrow \hat{S} + \alpha \mathbf{v}_k \mathbf{v}_k^T. \tag{5.7}$$

In order to keep a continuously updated set of values $\left\{ \mathbf{v}_i^T \hat{S}^{-1} \mathbf{v}_j \mid i, j = 1, \ldots, K \right\}$ while $\hat{S}$ is being changed, we can exploit the Sherman-Morrison-Woodbury (SMW) formula. The general form of the SMW formula is given by:

$$\left(A + \mathbf{u}\mathbf{v}^T\right)^{-1} = A^{-1} - \frac{A^{-1}\mathbf{u}\mathbf{v}^T A^{-1}}{1 + \mathbf{v}^T A^{-1} \mathbf{u}}. \tag{5.8}$$

Substituting $\mathbf{u}\mathbf{v}^T = \alpha \mathbf{v}_k \mathbf{v}_k^T$ and $A = \hat{S}$, we have, for $i, j = 1, \ldots, K$:

$$\left(\hat{S} + \alpha \mathbf{v}_k \mathbf{v}_k^T\right)^{-1} = \hat{S}^{-1} - \left(\frac{\alpha \hat{S}^{-1} \mathbf{v}_k \mathbf{v}_k^T \hat{S}^{-1}}{1 + \alpha \mathbf{v}_k^T \hat{S}^{-1} \mathbf{v}_k}\right) \tag{5.9}$$

The inner products $\mathbf{v}_i^T \hat{S}^{-1} \mathbf{v}_j$ can be updated using:

$$\mathbf{v}_i^T \left(\hat{S} + \alpha \mathbf{v}_k \mathbf{v}_k^T\right)^{-1} \mathbf{v}_j = \mathbf{v}_i^T \hat{S}^{-1} \mathbf{v}_j - \mathbf{v}_i^T \left(\frac{\alpha \hat{S}^{-1} \mathbf{v}_k \mathbf{v}_k^T \hat{S}^{-1}}{1 + \alpha \mathbf{v}_k^T \hat{S}^{-1} \mathbf{v}_k}\right) \mathbf{v}_j \tag{5.10}$$

$$= \mathbf{v}_i^T \hat{S}^{-1} \mathbf{v}_j - \frac{\alpha \left(\mathbf{v}_i^T \hat{S}^{-1} \mathbf{v}_k\right) \left(\mathbf{v}_j^T \hat{S}^{-1} \mathbf{v}_k\right)}{\left(1 + \alpha \mathbf{v}_k^T \hat{S}^{-1} \mathbf{v}_k\right)}. \tag{5.11}$$

Define the Mahalanobis Gram matrix Q:

$$Q = V^T \hat{S}^{-1} V, \tag{5.12}$$

such that $Q_{ij} = \mathbf{v}_i^T \hat{S}^{-1} \mathbf{v}_j$. Rewriting Equation (5.11) using the elements of Q,

$$Q_{ij}^{\text{new}} = Q_{ij} - \frac{\alpha Q_{ik} Q_{jk}}{1 + \alpha Q_{kk}}$$

$$\therefore Q^{\text{new}} = Q - \left(\frac{\alpha}{1 + \alpha Q_{kk}}\right) \mathbf{q}_k \mathbf{q}_k^T \tag{5.13}$$

where $\mathbf{q}_k$ denotes the k-th row (or column) of the symmetric matrix Q.

Therefore, when $\hat{S}$ is updated as in Equation (5.6), the Q matrix should be updated using Equation (5.13), enabling us to efficiently maintain an updated set of values for $\mathbf{v}_i^T \hat{S}^{-1} \mathbf{v}_j$ for $i, j = 1, \ldots, K$, during the coordinate descent iterations of the sparse coding procedure. This eliminates the need for inverting the new $\hat{S}$ at each iteration, resulting in only one $n \times n$ inversion of $\hat{S}_0 = \hat{S}(\mathbf{x}_0)$ during the initialization of the sparse coding algorithm.

The coordinate descent sparse coding algorithm for rank-1 dictionaries is presented in Algorithm 5.

Algorithm 5 Efficient Sparse Coding for Rank-1 Dictionaries

```
Input: Signal $S$, dictionary $V = [\ \mathbf{v}_1 \mid \mathbf{v}_2 \mid \ \dots \ \mid \mathbf{v}_K\ ]$, parameter $\lambda$
Output: Coefficient vector $\mathbf{x}$
  Compute $\mathbf{c}$: $c_i = \mathbf{v}_i^T S^{-1} \mathbf{v}_i$, $i = 1, \dots, K$
  Initialize $\mathbf{x} = \epsilon \mathbf{1}$, $\epsilon > 0$
  Compute $\hat{S} = \sum_{i=1}^{K} x_i \mathbf{v}_i \mathbf{v}_i^T$
  Compute $Q = V^T \hat{S}^{-1} V$
  repeat
    for $k = 1$ to $K$ do
      Compute descent direction $\delta x_k = \operatorname{sign}\left(\mathbf{v}_k^T \hat{S}^{-1} \mathbf{v}_k - c_k - \lambda\right) = \operatorname{sign}\left(Q_{kk} - c_k - \lambda\right)$
      Compute stepsize $\beta$ along $\delta x_k$ that minimizes $f\left(\mathbf{x} + \beta \delta x_k \mathbf{e}_k\right)$
      Update $x_k \leftarrow \left(x_k + \beta \delta x_k\right)_+$, where $(a)_+ = \max(a, 0)$
      Set $\alpha = x_k^{\text{new}} - x_k^{\text{old}}$
      Update $\hat{S} \leftarrow \hat{S} + \alpha \mathbf{v}_k \mathbf{v}_k^T$
      Update $Q \leftarrow Q - \left(\frac{\alpha}{1 + \alpha Q_{kk}}\right) \mathbf{q}_k \mathbf{q}_k^T$
    end for
  until convergence
```

5.2 Experiments with Rank-1 Sparse Coding

In this section, we show synthetic experiments for sparse coding over rank-1 dictionaries, comparing the interior point implementations in YALMIP+SDPT3 with our proposed coordinate descent approach of Algorithm 5.

Synthetic rank-1 dictionaries V were generated with different dimensions n and number of atoms K. $N = 100$ random sparse vectors $X^* = \{\mathbf{x}_j^*, j = 1, \ldots, N\}$ with sparsity $k = n$ were generated. These coefficients were used as ground-truth and the signals $\mathcal{S} = \{S_j, j = 1, \ldots, N\}$ were synthesized. The signals were sparse-coded with the dictionary with Algorithm 5, setting the parameter $\lambda = 0$. The coefficient estimation error $\|\mathbf{x} - \mathbf{x}^*\|_2$ and the objective function value $D_{\text{ld}}\left(\sum_{i=1}^{K} x_i \mathbf{v}_i \mathbf{v}_i^T, S\right)$ are shown in Figure 5.1 and Figure 5.2 respectively.

The experiment is repeated for different values of (n, K), each averaged over 20 iterations. The performance of the proposed algorithm and the interior point implementation (both in MATLAB) are compared, and our method provides improved accuracy in both measured quantities.

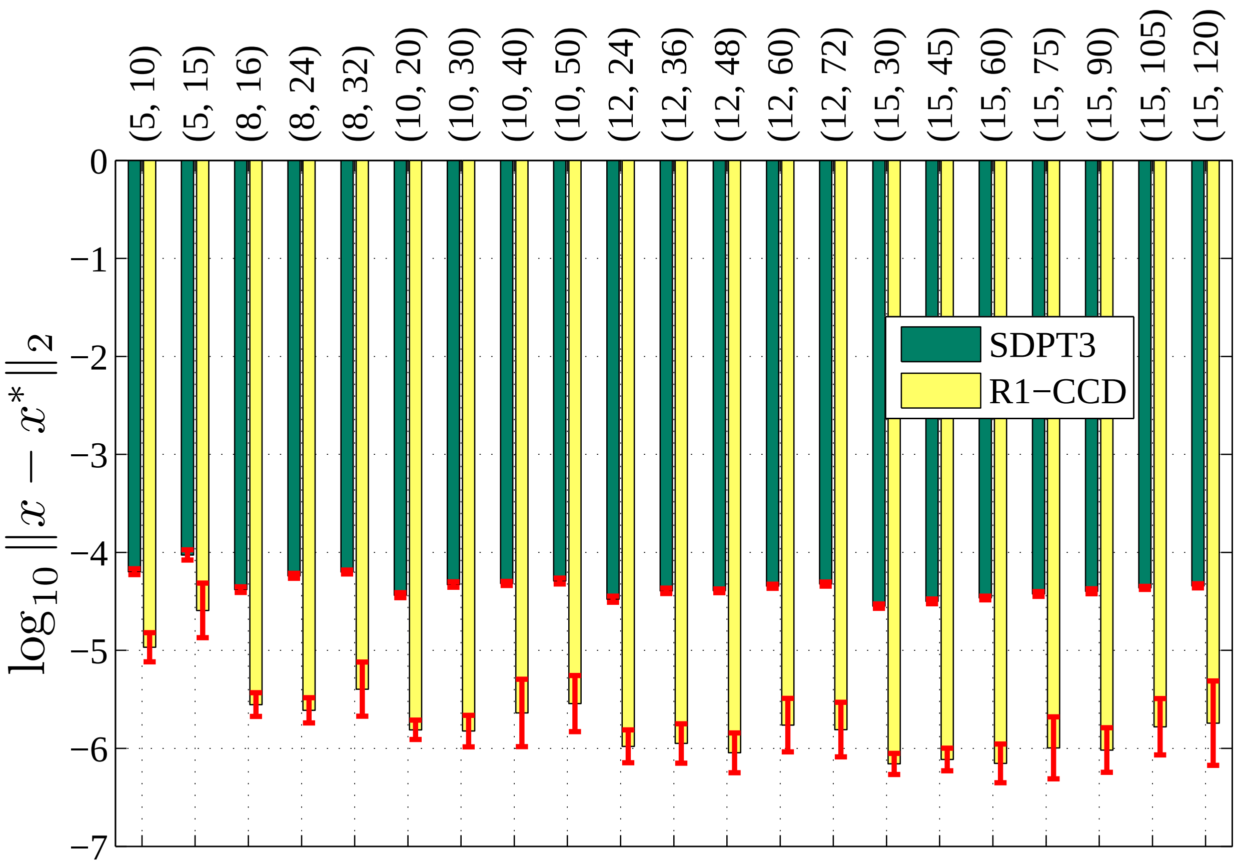


Figure 5.1: Average coefficient estimation error for sparse coding over rank-1 dictionaries, with different (n, K) values indicated in the X-axis. *SDPT3* indicates the interior-point algorithm implemented in YALMIP+SDPT3, while *R1-CCD* denotes our proposed coordinate descent method in Algorithm 5. We show an improvement in the coefficient error by 1-2 orders of magnitude. 1σ bars are also shown.

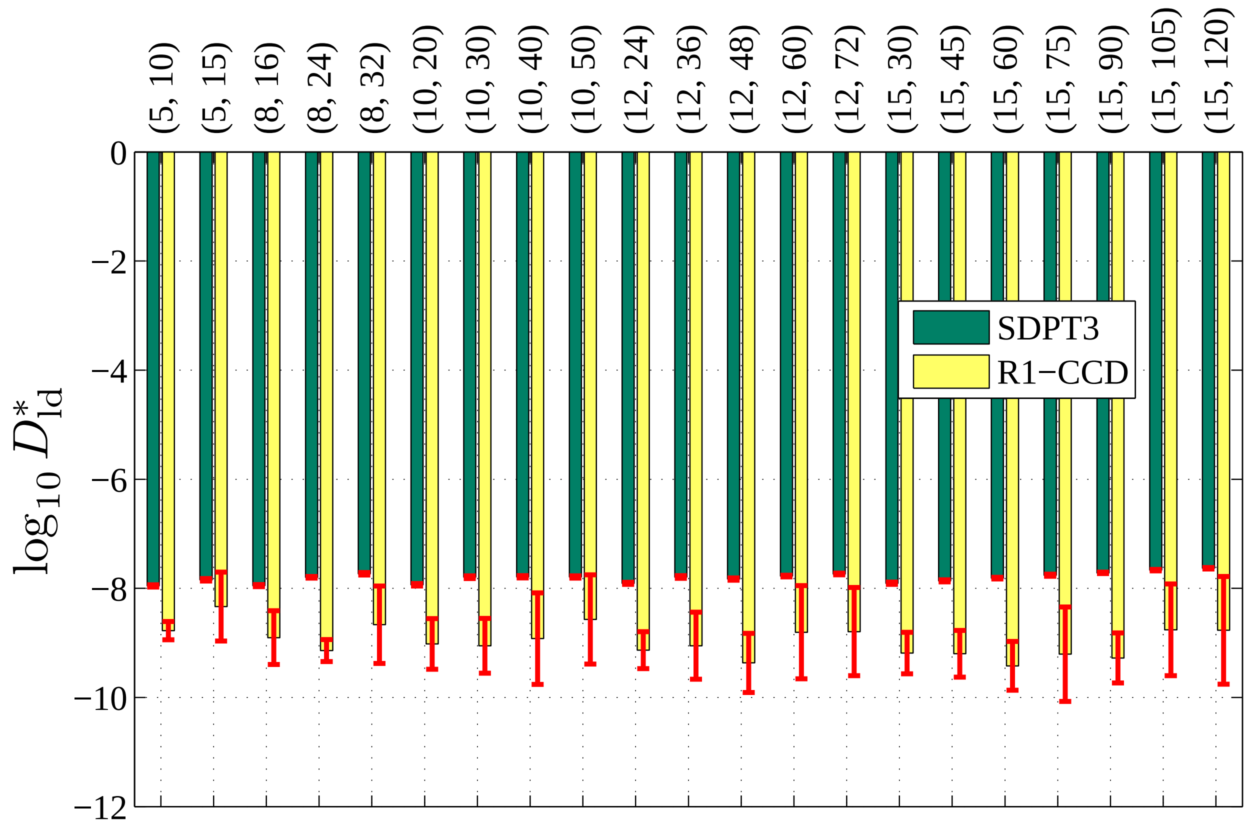


Figure 5.2: Average objective function value for sparse coding over rank-1 dictionaries, with different (n, K) values indicated in the X-axis. *SDPT3* indicates the interior-point algorithm implemented in YALMIP+SDPT3, while *R1-CCD* denotes our proposed coordinate descent method in Algorithm 5. We show an consistent improvement in the converged objective function value, up to 2 orders of magnitude. 1σ bars are also shown.

5.3 Learning Rank-1 Dictionaries

The dictionary learning problem for rank-1 dictionary atoms is given by:

$$\min_{V,X} \quad \sum_{j=1}^{N} \left\{ \mathrm{tr}\left(\sum_{i=1}^{K} x_{ij} \mathbf{v}_i \mathbf{v}_i^T S_j^{-1} \right) - \log\det\left(\sum_{i=1}^{K} x_{ij} \mathbf{v}_i \mathbf{v}_i^T S_j^{-1} \right) + \lambda \sum_{i=1}^{K} x_{ij} \right\} \tag{5.14a}$$

$$\begin{aligned} \text{s.t.} \quad & x_{ij} \geq 0 && \text{for } i = 1, \dots, K,\ j = 1, \dots, N \\ & \textstyle\sum_{i=1}^{K} x_{ij} \mathbf{v}_i \mathbf{v}_i^T \succeq 0 && \text{for } j = 1, \dots, N \\ & \|\mathbf{v}_i\|_2^2 \leq 1 && \text{for } i = 1, \dots, K \end{aligned} \tag{5.14b}$$

Similar to the positive definite dictionary learning problem from Chapter 4, we also use the alternating minimization approach to learn the dictionary V. The dictionary V^0 is initialized by sampling a random subset of K signals from $\mathcal{S}$ and computing their principal eigenvectors. The sparse coding is performed efficiently using the algorithm presented Section 5.1. The dictionary learning algorithm is presented in Algorithm 6.

The dictionary update step involves sequentially updating one atom at a time. To update atom $\mathbf{v}_i$, the optimization sub-problem (over the active set ω_i) is:

$$\min_{\mathbf{v}_i : \mathbf{v}_i^T \mathbf{v}_i \leq 1} \quad \sum_{j \in \omega_i} \left\{ x_{ij} \mathrm{tr}\left(\mathbf{v}_i \mathbf{v}_i^T S_j^{-1} \right) - \log\det\left(\sum_{i'=1}^{K} x_{i'j} \mathbf{v}_{i'} \mathbf{v}_{i'}^T S_j^{-1} \right) \right\}. \tag{5.15}$$

Define the objective in (5.15) as $f(\mathbf{v}_i)$, ignoring the terms which are independent of $\mathbf{v}_i$:

$$f(\mathbf{v}_i) = \sum_{j \in \omega_i} \left\{ x_{ij} \mathbf{v}_i^T S_j^{-1} \mathbf{v}_i - \log\det\left(\sum_{i'=1}^{K} x_{i'j} \mathbf{v}_{i'} \mathbf{v}_{i'}^T \right) \right\}. \tag{5.16}$$

Algorithm 6 **Rank-1 Dictionary Learning**

```
Input: Data $\mathcal{S} = \{S_j\}_{j=1}^{N}$, dictionary size $K$, sparsity parameter $\lambda$
Output: $V = [\mathbf{v}_i], i = 1, \ldots, K$
  $k = 0$
  Initialize $V^0$ by sampling from principal eigenvectors of data points in $\mathcal{S}$
  repeat
    $k \leftarrow k + 1$
    Given $\mathcal{S}$ and $V^{k-1}$, compute the sparse coefficients $X^k$ using Algorithm 5
    for $i = 1$ to $K$ do
      Update atom $\mathbf{v}_i^{k-1}$ to $\mathbf{v}_i^k$, along with the corresponding coefficients in $X^k$ (Algorithm 7)
    end for
  until convergence
```

Algorithm 7 **Rank-1 Atom Update**

```
Input: $\mathbf{v}_i$, $\left\{x_{ij}, S_j, \hat{S}_j \mid j \in \omega_i\right\}$
Output: $\mathbf{v}_i$, $\left\{x_{ij}, \hat{S}_j \mid j \in \omega_i\right\}$
  repeat
    Compute descent direction $\Delta\mathbf{v}_i$ using (5.18) or (5.22)
    Choose stepsize $\alpha$ by line search
    $\mathbf{v}_i^{new} \leftarrow \mathbf{v}_i + \alpha\Delta\mathbf{v}_i$
    $\hat{S}_j \leftarrow \hat{S}_j + x_{ij}\left(\mathbf{v}_i^{new}\left(\mathbf{v}_i^{new}\right)^T - \mathbf{v}_i\mathbf{v}_i^T\right) \quad \forall j \in \omega_i$
    $t = \max\left\{\left\|\mathbf{v}_i^{new}\right\|_2, 1\right\}$
    $\mathbf{v}_i \leftarrow \mathbf{v}_i^{new}/t$
    $x_{ij} \leftarrow t^2\, x_{ij} \quad \forall j \in \omega_i$
  until convergence
```

5.3.1 Gradient Descent

Taking the gradient of $f(\mathbf{v}_i)$ w.r.t. $\mathbf{v}_i$,

$$\frac{\partial f(\mathbf{v}_i)}{\partial \mathbf{v}_i} = \sum_{j\in\omega_i} \left\{ 2x_{ij} S_j^{-1}\mathbf{v}_i - 2x_{ij}\left(x_{ij}\mathbf{v}_i\mathbf{v}_i^T + \sum_{i'\neq i} x_{i'j}\mathbf{v}_{i'}\mathbf{v}_{i'}^T \right)^{-1} \mathbf{v}_i \right\}$$

$$= 2\sum_{j\in\omega_i} x_{ij}\left\{ S_j^{-1} - \hat{S}_j^{-1} \right\} \mathbf{v}_i \tag{5.17}$$

The gradient descent direction $\Delta\mathbf{v}_i^g$ is given by the negative of the gradient:

$$\Delta\mathbf{v}_i^g = 2\sum_{j\in\omega_i} x_{ij}\left\{ \hat{S}_j^{-1} - S_j^{-1} \right\} \mathbf{v}_i \tag{5.18}$$

The gradient descent update algorithm is, therefore,

$$\mathbf{v}_i^k \leftarrow \mathbf{v}_i^{k-1} + \alpha\Delta\mathbf{v}_i^g, \tag{5.19}$$

with stepsize $\alpha \geq 0$ determined using line search techniques, such as *exact* or *backtracking* (*Armijo*) line search.

5.3.2 Newton Descent

The Hessian of $f(\mathbf{v}_i)$ w.r.t. $\mathbf{v}_i$ is given by:

$$\frac{\partial^2 f(\mathbf{v}_i)}{\partial \mathbf{v}_i^2} = 2\sum_{j\in\omega_i} x_{ij}\left\{ S_j^{-1} - \hat{S}_j^{-1} + x_{ij}\hat{S}^{-1}\mathbf{v}_i\mathbf{v}_i^T\hat{S}^{-1} + x_{ij}\left(\mathbf{v}_i^T\hat{S}^{-1}\mathbf{v}_i\right)\hat{S}^{-1} \right\} \tag{5.20}$$

$$= 2\sum_{j\in\omega_i} x_{ij}\left\{ S_j^{-1} + \left[x_{ij}\mathbf{v}_i^T\hat{S}_j^{-1}\mathbf{v}_i - 1 \right]\hat{S}_j^{-1} + x_{ij}\left(\hat{S}_j^{-1}\mathbf{v}_i\right)\left(\hat{S}_j^{-1}\mathbf{v}_i\right)^T \right\} \tag{5.21}$$

The Newton descent direction $\Delta\mathbf{v}_i^N$ is obtained by solving:

$$\frac{\partial^2 f(\mathbf{v}_i)}{\partial \mathbf{v}_i^2}\,\Delta\mathbf{v}_i^N = -\frac{\partial f(\mathbf{v}_i)}{\partial \mathbf{v}_i}, \tag{5.22}$$

and the Newton descent update algorithm is, therefore,

$$\mathbf{v}_i^k \leftarrow \mathbf{v}_i^{k-1} + \alpha\Delta\mathbf{v}_i^N, \tag{5.23}$$

with stepsize $\alpha \geq 0$.

5.3.3 Reconstruction Update

When the atom $\mathbf{v}_i$ is modified, the reconstructions $\hat{S}_j$, $j \in \omega_i$ are also updated, according to Algorithm 7:

$$\hat{S}_j \leftarrow \hat{S}_j + x_{ij} \left(\mathbf{v}_i^{new} \left(\mathbf{v}_i^{new} \right)^T - \mathbf{v}_i \mathbf{v}_i^T \right) \quad \forall j \in \omega_i. \tag{5.24}$$

Since this is a rank-2 update, we can compute the modified inverses $\hat{S}_j^{-1}$ using the low-rank version of the Sherman-Morrison-Woodbury formula:

$$(A + UCV)^{-1} = A^{-1} - A^{-1} U \left(C^{-1} + V A^{-1} U \right)^{-1} V A^{-1}, \tag{5.25}$$

by setting

$$A = \hat{S}_j, \quad C = \left[\begin{array}{cc} x_{ij} & 0 \\ 0 & -x_{ij} \end{array} \right], \quad U = V^T = [\ \mathbf{v}_i^{\text{new}} \mid \mathbf{v}_i\ ]\,. \tag{5.26}$$

This is equivalent to two iterations of the rank-1 update (5.8) and therefore has the same order-complexity.

5.4 Time Complexity

5.4.1 Sparse Coding

The sparse coding algorithm for rank-1 dictionaries has a cost of $\mathcal{O}(n^2)$ for updating each coordinate. For each outer iteration, therefore, the cost is $\mathcal{O}(n^2K)$. If the maximum number of outer iterations is denoted by $L_{\max}$, the complexity of sparse coding one signal S over a dictionary of K $n \times n$ rank-1 atoms is $\mathcal{O}(n^2KL_{\max})$.

We show sparse coding computation times on an AMD Phenom II X6-1090T 3.2 GHz 64-bit 6-core desktop computer with 8GB RAM. Figure 5.3 shows the average sparse coding time per signal for different values of (n, K), averaged over 50 runs of 1000 signals each. The parameter λ is also varied in $\{0, 0.1, 1, 10\}$. When $K < 3n$, the sparse coding is very fast (much faster than the general sparse coding algorithm), but when $K \geq 3n$, the algorithm takes longer to converge to the specified precision.

5.4.2 Dictionary Update

The Newton direction can be computed in $\mathcal{O}(n^3 + n^2N)$ time, accounting for the $n \times n$ matrix inversion and computing sums of matrices over the active set ω_i, $|\omega_i| \leq N$. The stepsize estimation using line search involves at most one generalized eigenvalue computation, costing $\mathcal{O}(n^3)$. The reconstruction update takes $\mathcal{O}(n^2N)$ time. Bounding the number of inner iterations in the atom update by $M_{\max}$, the complexity of one atom update is $\mathcal{O}((n^3 + n^2N)\, M_{\max})$. Since the number of training samples is usually large compared to the data dimension $N \gg n$ - for *e.g.*, in the case of region covariance descriptors, this can be approximated as $\mathcal{O}(n^2NM_{\max})$. Therefore one iteration of the dictionary update stage has a time complexity of $\mathcal{O}(n^2KNM_{\max})$.

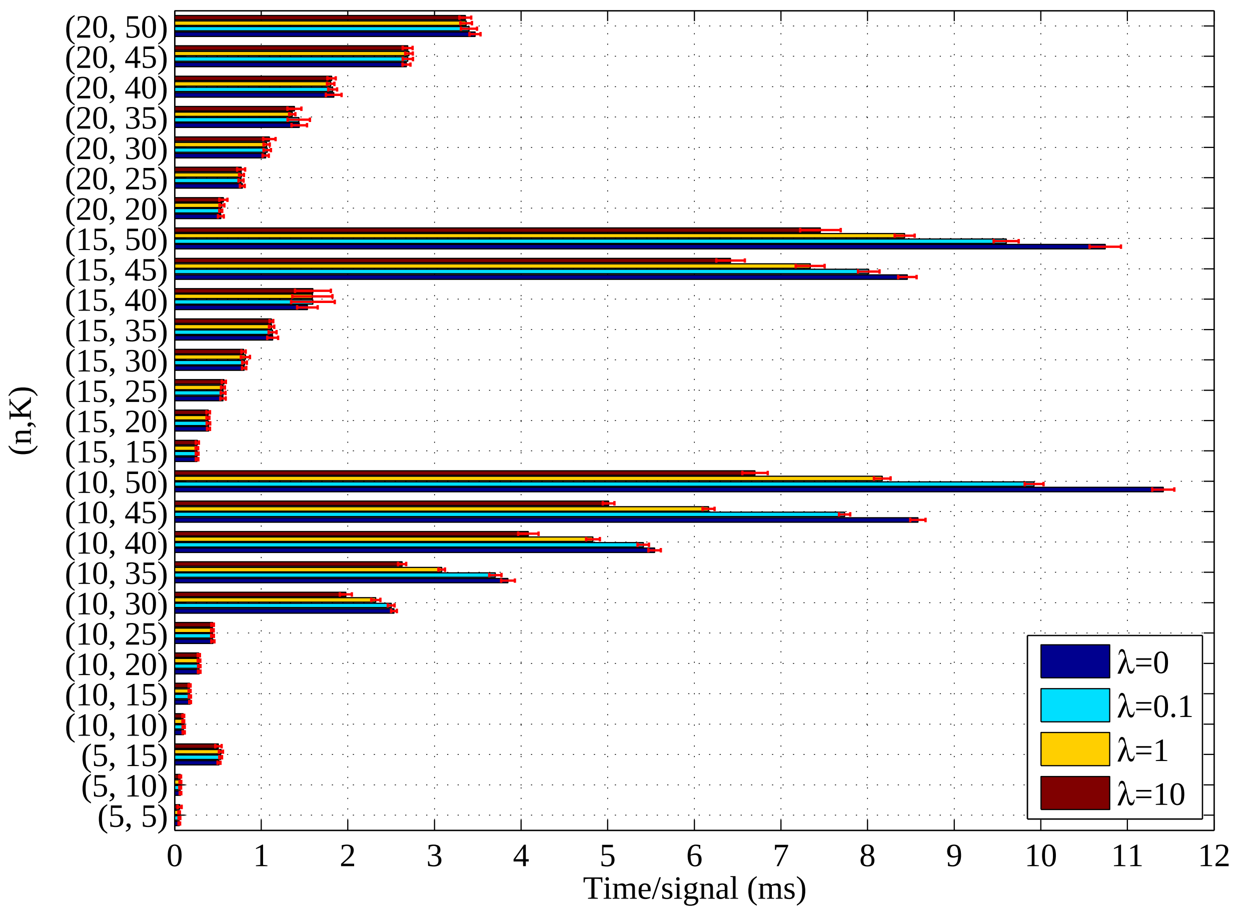


Figure 5.3: Average sparse coding times over rank-1 dictionaries, for different (n, K) value pairs. The parameter λ is also varied. Results are averaged over 50 runs, and 1σ bars are also shown.

5.5 Atom Recovery

We test the rank-1 dictionary learning approach by generating a ground-truth synthetic dictionary $V^* = [\ \mathbf{v}_1^* \mid \mathbf{v}_2^* \mid \ \dots \ \mid \mathbf{v}_K^*\ ]$ of size $n \times K$, with unit-norm columns. Random sparse vectors $\mathbf{x}$ are generated and signals S are synthesized ($N = 4nK$). The dictionary learning approach from Algorithm 6 is applied with Newton descent for a maximum of 50 iterations, setting $\lambda = 0$. We test the number of atoms correctly recovered in the learned dictionary $\hat{V}$ from the original dictionary V^*, as in [Aharon et al., 2006]. The atoms are matched using a threshold of $|\langle \mathbf{v}_i^*, \hat{\mathbf{v}}_i \rangle| \geq 0.99$. The percentage of atoms correctly recovered for different values of (n, K) is shown in Figure 5.4. It can be seen that for small values of K, the atoms are recovered almost exactly.

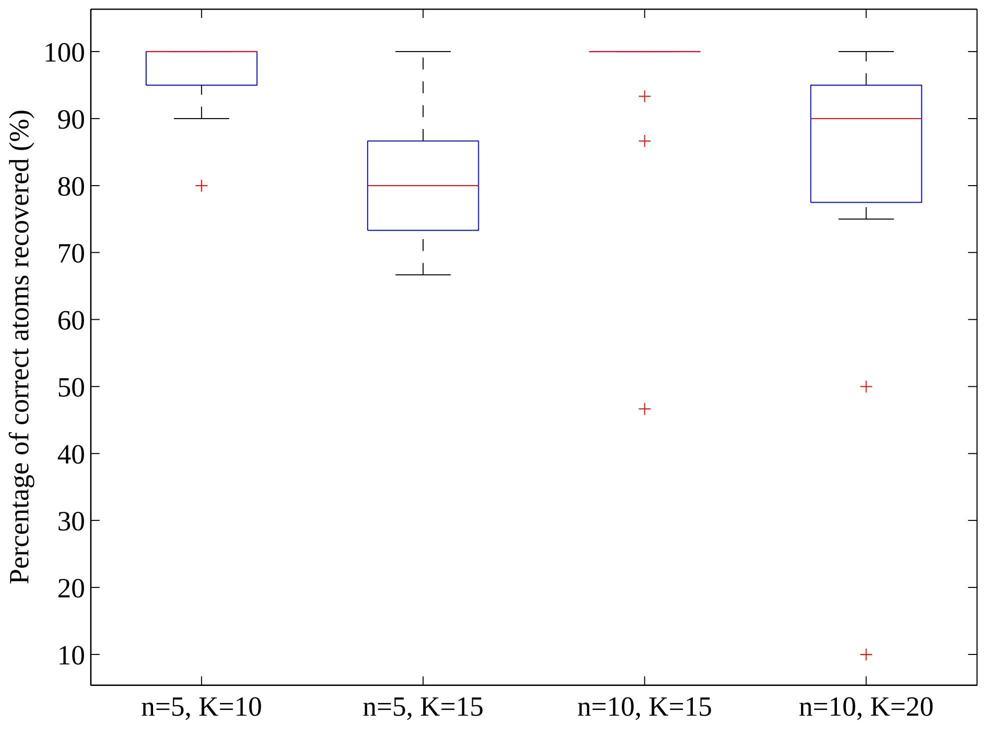


Figure 5.4: Boxplot of the percentage of atoms correctly recovered for different values of (n, K), over 20 trials, with $N = 4nK$ training samples.

5.6 USPS Digits Classification with Rank-1 Dictionaries

In this section, we apply the dictionary learning algorithm for rank-1 positive semidefinite dictionary atoms to classification of handwritten digits from the USPS dataset. We use a subset of 1100 samples from each class, split into 1000 training and 100 test samples. From each 16×16 digit image, a 5×5 region covariance descriptor is computed using the grayscale intensity and absolute values of the first- and second-order spatial derivatives, $\{I, |I_x|, |I_y|, |I_{xx}|, |I_{yy}|\}$. Some sample images from class are shown in Figure 5.5.

Figure 5.5: Sample images from each class in the USPS digit recognition dataset.

Dictionaries $V_0, V_1, \ldots, V_9$ each of size $K = 10$ were learned independently for each digit class $0, 1, \ldots, 9$, and then used for classifying the test data based on the class dictionary yielding the minimum reconstruction error. The classification accuracy was tested at each iteration in the dictionary training process, as a measure of how well the rank-1 dictionary learning process models the positive definite signals from each class. This is shown in Figure 5.6. The initial dictionaries are randomly sampled from the data, and in a few training iterations the accuracy shows a marked improvement.

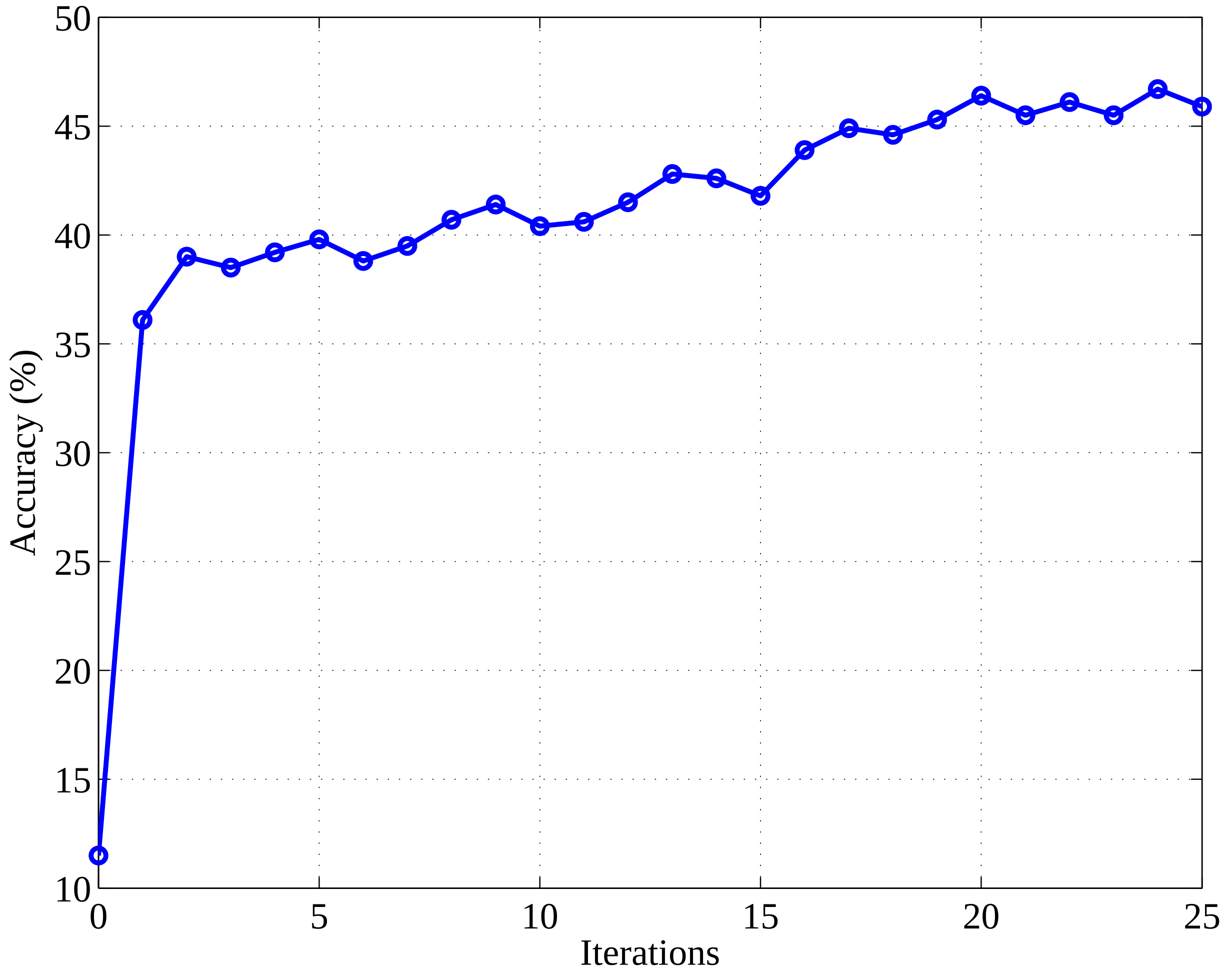


Figure 5.6: Classification accuracy of USPS digits vs. number of dictionary learning iterations. The initial dictionary is randomly sampled from the data. In a few learning iterations, the dictionaries model the individual classes and show a marked increase in the classification accuracy.

Chapter 6

Discriminative Dictionary Learning

Sparse models have been used extensively to classify or cluster data. Learning dictionaries for each class independently without information from the other classes can be compared to *generative modeling*, which may not be able to classify or cluster data with sufficient accuracy when there is significant overlap in the feature space. Such a scenario calls for the use of *discriminative modeling*, where the learning should promote discrimination between the sparse models of each class. In other words, the dictionary learned for a certain class should provide good reconstruction for the signals from that class, and poor reconstruction for signals that do not belong to that class. Conversely, a signal from a certain class should be reconstructed best by a dictionary of the same class, compared to all other class dictionaries.

In the vector sparse modeling literature, [Mairal et al., 2008, Ramirez et al., 2010] have used different formulations to solve the dictionary learning problem while increasing the discriminative power of the learned dictionaries. [Mairal et al., 2008] use a logistic loss term in their objective function that penalizes for misclassification of signals. In [Ramirez et al., 2010], however, the discrimination is learned in terms of the incoherence between atoms of different class dictionaries. We follow the latter approach in learning discriminative positive definite dictionaries.

6.1 Discriminative Dictionary Learning Formulation

Given training data from C different classes, we will attempt to learn the dictionary for each class $c = 1, \ldots, C$. The sizes of the training data from each class c is given by N_c. The training data from class c is specified as $\mathcal{S}^{(c)} = \left\{ S_j^{(c)} \right\}$, $j = 1, \ldots, N_c$, and the dictionary learned to model this data is denoted by $\mathcal{A}^{(c)} = \left\{ A_i^{(c)} \right\}$, $i = 1, \ldots, K_c$, K_c being the dictionary size for class c.

The discriminative power of the dictionaries is induced by including a term which promotes incoherence between the dictionaries of different classes - *i.e.*, between each class c dictionary $\mathcal{A}^{(c)}$ and all other dictionaries $\mathcal{A}^{(c')}$, $c' \neq c$. This is motivated by the work of [Ramirez et al., 2010] in learning discriminative dictionaries for classification and clustering. Similar to their work, we will use our definition of atom coherence from Section 3.7 and penalize for the coherence between atoms from dictionaries of different classes.

We define the coherence between two dictionaries $\mathcal{A}^{(c)}$ and $\mathcal{A}^{(c')}$ by

$$\mathcal{Q}\left(\mathcal{A}^{(c)}, \mathcal{A}^{(c')}\right) = \sum_{i=1}^{K_c} \sum_{i'=1}^{K_{c'}} \left\langle A_i^{(c)}, A_{i'}^{(c')} \right\rangle . \tag{6.1}$$

The discriminative dictionary learning problem is given by:

$$\min_{\mathcal{A}^{(1)},\ldots,\mathcal{A}^{(C)}} \quad \sum_{c=1}^{C} \left[\sum_{j=1}^{N_c} \left\{ \min_{\substack{\mathbf{x} \geq 0 \\ \mathcal{A}^{(c)}\mathbf{x} \succeq 0}} \quad D_{\text{ld}}\left(\mathcal{A}^{(c)}\mathbf{x}, S_j^{(c)}\right) + \lambda \left\|\mathbf{x}\right\|_1 \right\} + \eta \sum_{c' \neq c} \mathcal{Q}\left(\mathcal{A}^{(c)}, \mathcal{A}^{(c')}\right) \right] \tag{6.2a}$$

$$\begin{aligned} \text{s.t.} \qquad A_i^{(c)} &\succeq 0 && \text{for } i = 1, \ldots, K_c,\ c = 1, \ldots, C \\ \left\| A_i^{(c)} \right\|_F^2 &\leq 1 && \text{for } i = 1, \ldots, K_c,\ c = 1, \ldots, C \end{aligned} \tag{6.2b}$$

This coherence term is convex (in fact, linear) in one argument, given the other fixed. Therefore, while updating the class c dictionary $\mathcal{A}^{(c)}$, all other class dictionaries are fixed.

The alternating minimization between the sparse coding and dictionary update stages is the same as in the usual dictionary learning approach.

Writing out the coherence term $\mathcal{Q}$ in (6.2),

$$\begin{aligned}
\sum_{c' \neq c} \mathcal{Q}\left(\mathcal{A}^{(c)}, \mathcal{A}^{(c')}\right) &= \sum_{c' \neq c} \sum_{i=1}^{K_c} \sum_{i'=1}^{K_{c'}} \operatorname{tr}\left(A_i^{(c)} A_{i'}^{(c')}\right) \\
&= \sum_{i=1}^{K_c} \operatorname{tr}\left(A_i^{(c)} \left(\sum_{c' \neq c} \sum_{i'=1}^{K_{c'}} A_{i'}^{(c')}\right)\right) \\
&= \sum_{i=1}^{K_c} \operatorname{tr}\left(A_i^{(c)} M^{(c)}\right) \qquad \text{where } M^{(c)} = \sum_{c' \neq c} \sum_{i'=1}^{K_{c'}} A_{i'}^{(c')}.
\end{aligned}$$

While updating the dictionary from class c, the factor $M^{(c)}$ encompasses the influence of all the other class dictionary atoms. This is independent of the atom number i in dictionary $\mathcal{A}^{(c)}$. The linear penalty $\operatorname{tr}\left(A_i^{(c)} M^{(c)}\right)$ merely adds an $\eta M^{(c)}$ term to the gradient expression for the dictionary learning problem in Equation (4.7). Denoting the objective in (6.2) by $f\left(\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(C)}\right)$, the gradient w.r.t. $A_i^{(c)}$ is given by:

$$\frac{\partial}{\partial A_i^{(c)}} f\left(\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(C)}\right) = \sum_{j \in \omega_i^{(c)}} x_{ij}^{(c)} \left[\left(S_j^{(c)}\right)^{-1} - \left(\hat{S}_j^{(c)}\right)^{-1}\right] + \eta M^{(c)}. \tag{6.3}$$

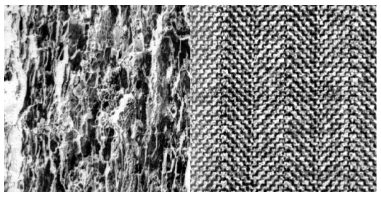
The Hessian from the dictionary learning problem in Equation (4.12) does not change since the coherence term $\mathcal{Q}$ is linear.

The discriminative atom update in Algorithm 8 can be performed using either the gradient descent or Newton descent methods in Section 4.3.

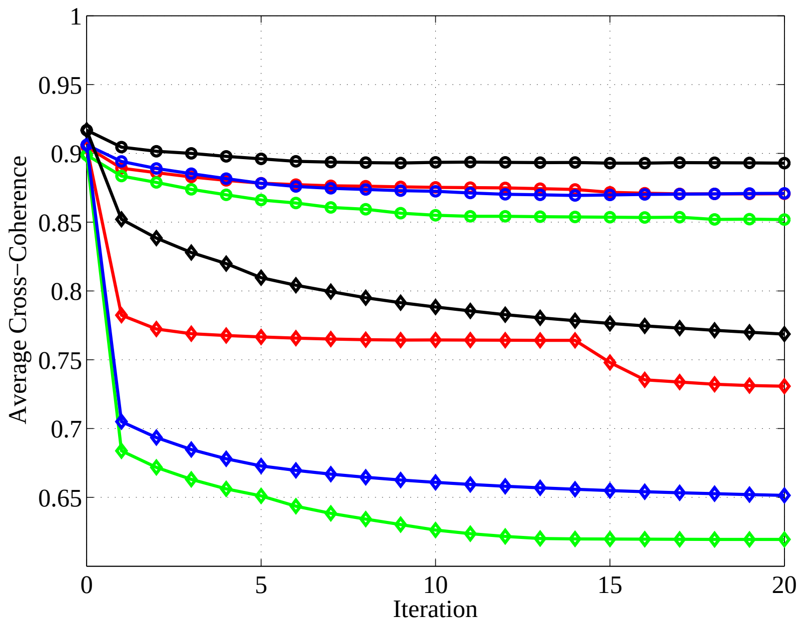
Algorithm 8 Discriminative Dictionary Learning

```
Input: Data $\mathcal{S}^{(c)} = \left\{S_j^{(c)}\right\}_{j=1}^{N_c}$, $c = 1, \dots, C$, dictionary size $K$, sparsity parameter $\lambda$,
       incoherence parameter $\eta$
Output: $\mathcal{A}^{(c)} = \left\{A_i^{(c)}\right\}_{i=1}^{K}$, $c = 1, \dots, C$
  $k = 0$
  for $c = 1$ to $C$ do
    Initialize $\mathcal{A}_0^{(c)}$ sampled from $\mathcal{S}^{(c)}$
  end for
  repeat
    $k \leftarrow k + 1$
    for $c = 1$ to $C$ do
      Given $\mathcal{S}^{(c)}$ and $\mathcal{A}_{k-1}^{(c)}$, compute the sparse coefficients $X_k^{(c)}$
    end for
    for $c = 1$ to $C$ do
      Given $\mathcal{S}^{(c)}$, $X_k^{(c)}$, and other class dictionaries $\left\{\mathcal{A}_k^{(1)}, \dots, \mathcal{A}_k^{(c-1)}, \mathcal{A}_{k-1}^{(c+1)}, \dots, \mathcal{A}_{k-1}^{(C)}\right\}$,
      compute the updated dictionary $\mathcal{A}_k^{(c)}$
    end for
  until convergence
```

To show that the discriminative dictionary learning approach presented here does reduce the coherence between the dictionaries, we pick the two textures from a mosaic in the Brodatz dataset [Randen and Husøy, 1999], shown in Figure 6.1(a). The texture descriptor is computed as in Section 3.13.3 producing 5×5 covariances. Individual dictionaries of varying sizes ($K = 10, 15, 20, 50$) are learned for each class, with ($\eta = 5$) and without ($\eta = 0$) the discriminative penalty term. The decrease in the average cross-coherence between atoms of the two dictionaries is shown in Figure 6.1(b). The use of the coherence penalty term shows a clear improvement in the resultant average cross-coherence between the trained dictionaries. Next we will show improvements in the classification accuracy for practical applications.



(a)



(b)

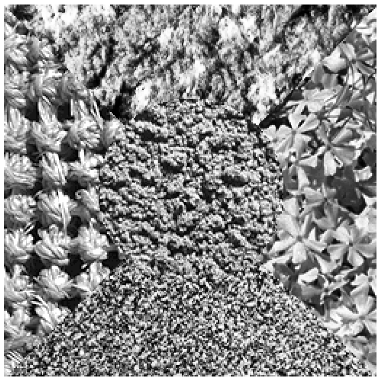
Figure 6.1: Plot of the average cross-coherence between atoms of two dictionaries for the texture (a) against the number of discriminative dictionary learning iterations. The dictionary size is varied as $K = 10$ (red), $K = 15$ (green), $K = 20$ (blue) and $K = 50$ (black). The symbol $\circ$ denotes dictionary learning without the discriminative term $\eta = 0$, and $\Diamond$ denotes discriminative dictionary learning, with $\eta = 5$. Discriminative learning reduces the average coherence between the dictionary atoms of the different classes.

6.2 Discriminative Dictionaries for Brodatz Textures

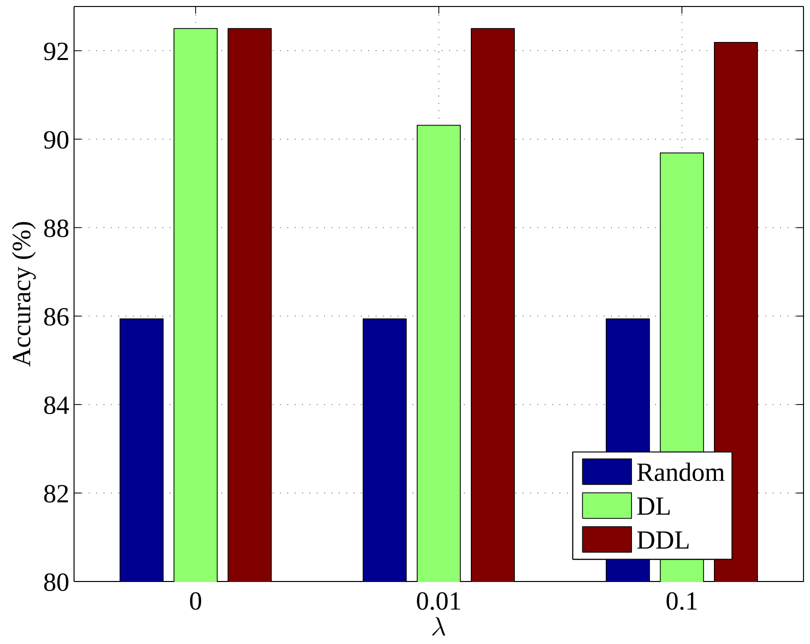
We apply the discriminative dictionary learning algorithm from Algorithm 8 to classify two example textures from the Brodatz texture mosaics dataset [Randen and Husøy, 1999]. Each of these examples have 5 different texture classes. The same procedure as in Section 3.13.3 was applied, but with different types of dictionaries:

1. Randomly sampled from the data
2. Learned from the data independently in each class (denoted as *DL*)
3. Learned from the data discriminatively with the coherence penalty (denoted as *DDL*)

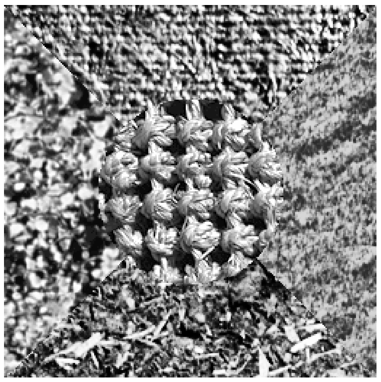
We chose a dictionary size of $K = 4$, and varied the sparsity regularizer λ. The value of η was set to be 0.1. The improvement of accuracy in the texture classification is shown in Figure 6.2. The learned dictionary improves the classification performance, and the discriminative training proves a further boost to the accuracy, sometimes substantially.



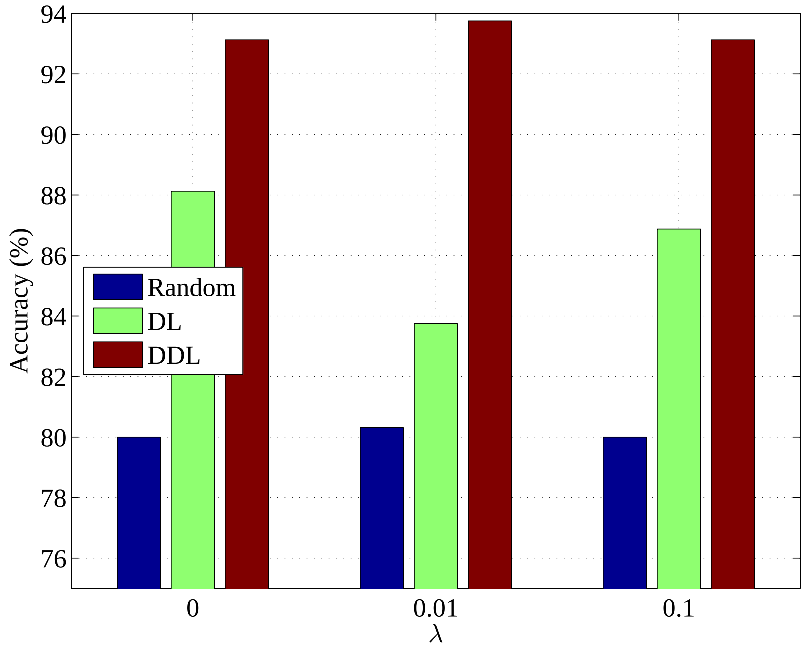
(a)



(b)



(c)



(d)

Figure 6.2: Comparison of accuracy between randomly initialized dictionaries (*random*) and dictionaries learned with (*DDL*) and without (*DL*) the discriminative penalty. The corresponding textures are shown on the left.

6.3 USPS Digits Classification with Discriminative Dictionary Learning

The USPS dataset from Section 5.6, with 1000 training and 100 test examples from each class $\{0, \ldots, 9\}$, was used to learn discriminative dictionaries for digit recognition. We denote the number of classes by C - $C = 10$. Two dictionary sizes $K = 15$ and $K = 30$ were tested, and λ was set to 0.1. The discriminative dictionary learning was applied for 10 iterations each, with the dictionary initialization obtained through K-means clustering of the individual data classes.

Table 6.1 shows the classification accuracy over the test set, comparing the use of independently and discriminatively trained dictionaries, with $\eta = 0$ and $\eta = 1/C$ respectively. The improvement in accuracy Δ shows that the use of discriminative dictionary learning definitely provides a benefit in the classification setting. Arbitrarily increasing η, however, does not always give improved performance - when we set $\eta = 10$ in this data set, the accuracy dropped by around 30%.

K	Accuracy		Δ
	$\eta = 0$	$\eta = \frac{1}{C}$	
15	60.8 %	67.2 %	+ 6.4 %
30	59.5 %	67.0 %	+ 7.5 %

Table 6.1: Classification accuracy on the USPS dataset with our tensor discriminative dictionary learning approach. The improvement Δ in the accuracy is shown on the right. The discriminative training improves the classification accuracy by a significant margin.

6.4 Action Recognition with Discriminative Learned Dictionaries

We apply the discriminative dictionary learning approach to classify actions from the KTH dataset from [Schuldt et al., 2004]. There are 6 different actions performed about 4 times each by 25 subjects (for a total of 598 sequences). We use the covariance feature representation from [Guo et al., 2010a], using the optical flow of the video frames. The features used are:

$$\phi(x,y,t) = [x, y, t, I_t, u, v, u_t, v_t, Div, Vor, Gten, Sten]^T ,$$

where (x, y, t) are the spatio-temporal coordinates of the corresponding pixels, and (u, v) are the optical flow values. I_t, u_t, v_t are the temporal gradient of the image intensity and optical flow features respectively. Div, Vor, $Gten$, and $Sten$ are the features derived from the optical flow, from [Guo et al., 2010a]:

$$Div(x,y,t) = \frac{\partial u(x,y,t)}{\partial x} + \frac{\partial v(x,y,t)}{\partial y},$$

$$Vor(x,y,t) = \frac{\partial v(x,y,t)}{\partial x} - \frac{\partial u(x,y,t)}{\partial y},$$

$$Gten(x,y,t) = \frac{1}{2}\left\{\mathrm{tr}\left(\nabla\mathbf{u}(x,y,t)\right)^2 - \mathrm{tr}\left(\nabla\mathbf{u}(x,y,t)^2\right)\right\},$$

$$\text{where } \nabla\mathbf{u}(x,y,t) = \begin{bmatrix} \frac{\partial u(x,y,t)}{\partial x} & \frac{\partial u(x,y,t)}{\partial y} \\ \frac{\partial v(x,y,t)}{\partial x} & \frac{\partial v(x,y,t)}{\partial y} \end{bmatrix},$$

$$Sten(x,y,t) = \frac{1}{2}\left\{\mathrm{tr}\left(S(x,y,t)\right)^2 - \mathrm{tr}\left(S(x,y,t)^2\right)\right\},$$

$$\text{where } S(x,y,t) = \frac{1}{2}\left(\nabla\mathbf{u}(x,y,t) + \nabla\mathbf{u}(x,y,t)^T\right).$$

These features are combined to form 12×12 covariance descriptors, computed over each subsequence in each action video, the start- and stop-frames of which are indicated in the

KTH dataset. In each frame, only the pixels with magnitude of optical flow above a certain threshold are used in the construction of the optical flow covariance descriptors.

We use the 8 training and 9 test subjects indicated in the dataset, and test our discriminative dictionary learning approach. We compare this with the baseline K-nearest-neighbors classification (best $K = 6$), as well as the vectorized-log-covariance sparse coding approach from [Guo et al., 2010a] (best sparsity $k = 2$ with our implementation). We only compare with these two methods using optical flow-based region covariance descriptors, and the classification accuracy is shown in Table 6.2. The procedure for our dictionary-based classification is the same as in the previous chapters. DL implies dictionary learning without discrimination ($\eta = 0$), and DDL denotes discriminative dictionary learning ($\eta \neq 0$). K, the number of dictionary atoms for each class dictionary, was also varied. λ was set to 0.1. Note that in the first two approaches, the entire training data is available to the classifier during test time, which is not the case in our approach. The learned dictionary models the features from the different classes, and the discriminative term improves the overall classification accuracy.

These experiments show that including a discriminative penalty term in the training procedure produces better results than learning the different class dictionaries independently.

Classification Approach	Accuracy
K-NN ($K = 2$)	82.50 %
[Guo et al., 2010a]	83.89 %
DL ($K = 15$)	83.43 %
DDL ($K = 15$)	83.78 %
DL ($K = 30$)	84.01 %
DDL ($K = 30$)	84.59 %
DL ($K = 60$)	85.75 %
DDL ($\mathbf{K} = \mathbf{60}$)	$\mathbf{86.37}$ %

Table 6.2: Classification accuracy on the KTH dataset, comparing the tensor dictionary learning approaches with K-nearest neighbor and Euclidean sparse coding over vectorized log-covariances. The discriminative dictionary learning approach provides the best accuracy in this experiment.

Chapter 7

TeSLa: Tensor Sparse Library

As part of the contributions of this thesis, we provide a software package called *Tensor Sparse Library*, or *TeSLa*, containing efficient and optimized C++ implementations of the main tensor sparse coding and dictionary learning algorithms presented here. The software uses the *Eigen* library [Guennebaud et al., 2010] with OpenMP, and has been tested in Windows and Mac environments.

MATLAB scripts are provided for reading and writing data, dictionary and coefficient files in the format compatible with the C++ software. These are written as separate text files whose names are then passed as arguments to the different programs. The data is encoded by vectorizing the upper triangular part of each matrix, in a column-major format. The data dimensions n and number of training points N are provided in the header. The dictionary atoms are formatted as complete $n \times n$ matrices, along with a header indicating n and dictionary size K. The coefficients $\mathbf{x}_j$, $j = 1, \ldots, N$, are written out as a $K \times N$ matrix.

7.1 Sparse Coding

The program `TensorSC` solves the tensor sparse coding using the coordinate descent algorithm inAlgorithm 1. It takes the data $\mathcal{S} = \{S_j\}_{j=1}^N$, $S_j \in \mathbf{S}_{++}^n$ and dictionary $\mathcal{A} = \{A_i\}_{i=1}^K$, $A_i \in \mathbf{S}_+^n$ as inputs and returns the sparse coefficients $X \in \mathbf{R}_+^{K\times N}$. The regularization parameter λ can be specified, or it defaults to $\lambda = 0$. The program automatically determines if the dictionaries are rank-1 outer products, and switches to the efficient rank-1 sparse coding algorithm from Algorithm 5.

Arguments:

`-input`	`< inputFilename >`	Data input filename
`-dictionary`	`< dictionaryFilename >`	Dictionary input filename
`-lambda`	`< lambdaValue >`	λ input sparsity parameter
`-output`	`< outputFilename >`	Coefficient output filename

We show sparse coding computation times on an AMD Phenom II X6-1090T 3.2 GHz 64-bit 6-core desktop computer with 8GB RAM. Figure 7.1 shows the average sparse coding time per signal for different values of (n, K), averaged over 50 runs of 1000 signals each. The typical dimension of region covariance descriptors is $n = 5$, and it can be seen that for reasonable dictionary sizes $K \leq 50$ the sparse coding takes *under 1 millisecond.*

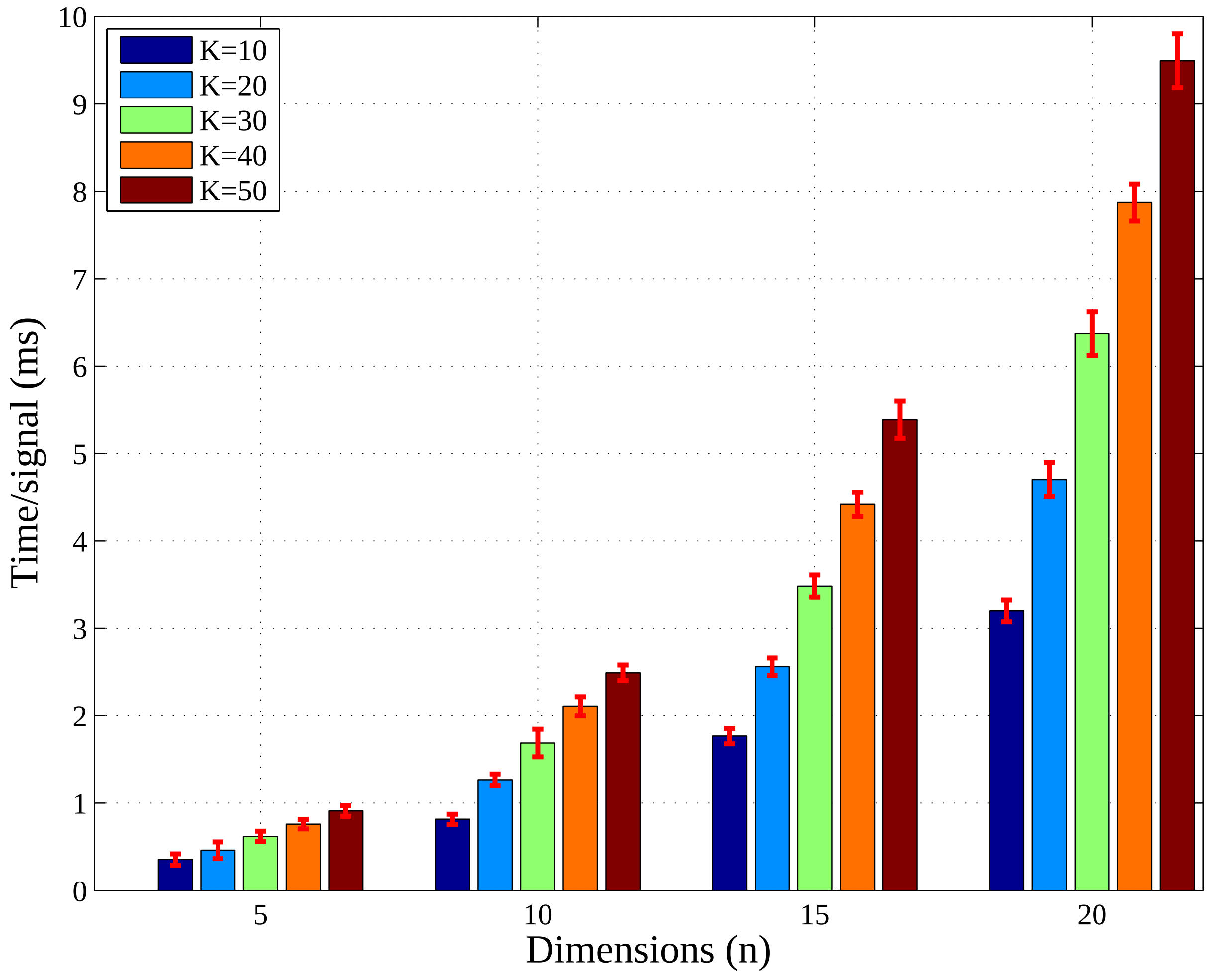


Figure 7.1: Average sparse coding times per $n \times n$ positive definite signal S over a dictionary $\mathcal{A}$ of size K, using the coordinate descent algorithm ($\lambda = 0$). 3σ bars are also shown.

7.2 Dictionary Learning

The program `TensorDL` solves the tensor dictionary learning problem using alternative minimization. It alternates between the sparse coding and dictionary update stages, for a specified number of iterations. Both the gradient atom update and Newton atom update strategies are implemented, along with the rank-1 dictionary update mode. The training data set $\mathcal{S} = \{S_j\}_{j=1}^{N}$ is passed as input to the program, along with the dictionary size K, regularization parameter λ, and the number of iterations *nIters*. The choice of learning rank-1 dictionaries can be specified by setting a Boolean flag *rank1*= true, which is false by default. The sparse coefficients are not returned in this program.

Arguments:

`-input`	`< inputFilename >`	Data input filename
`-K`	`< Kvalue >`	Dictionary size parameter
`-lambda`	`< lambdaValue >`	λ input sparsity parameter
`-nIters`	`< numberOfIters >`	Number of iterations
`-rank1`	`< flag >`	Flag specifying use of rank-1 dictionaries
`-dictionary`	`< dictionaryFilename >`	Dictionary output filename

7.3 Discriminative Dictionary Learning

The program `TensorDDL` solves the tensor discriminative dictionary learning problem. The number of classes C is provided as an input, and the training data set is provided in a set of C different input files with a common prefix - *i.e.*, *filename1*, *filename2*, . . . , *filenameC*. Only the prefix *filename* needs to be specified. Similarly, the learned dictionaries are written to C different files with the specified dictionary filename prefix. The sparsity regularization parameter λ and incoherence regularization parameter η are provided as inputs, or use default values of $\lambda = 0$, $\eta = 0$. When $\eta = 0$, the dictionaries are not learned discriminatively, rather the dictionaries for the different classes are learned independently of each other.

Arguments:

`-input`	`< inputFilenamePrefix >`	Data input filename prefix
`-C`	`< Cvalue >`	Number of classes
`-K`	`< Kvalue >`	Dictionary size parameter
`-lambda`	`< lambdaValue >`	λ input sparsity parameter
`-eta`	`< etaValue >`	η input incoherence parameter
`-nIters`	`< numberOfIters >`	Number of iterations
`-rank1`	`< flag >`	Flag for using rank-1 dictionaries
`-dictionary`	`< dictionaryFilenamePrefix >`	Dictionary output filename prefix

Chapter 8

Summary and Contributions

In this thesis, we have proposed novel sparse modeling techniques for positive definite matrices. The vast success of vector sparse modeling approaches in computer vision and machine learning served as the motivation to develop sparse models for the special case of positive definite matrices, which appear in many areas including probability, statistics, control theory, and medical imaging. We present sparse coding and dictionary learning techniques, with efficient implementations for each. The effects of the various quantities involved in the modeling are explored to get a deeper understanding of the presented methods. Special extensions to dictionaries with rank-one matrices and discriminative models are developed and applied to practical scenarios.

8.1 Contributions of the Thesis

- We designed a new tensor sparse coding technique from the ground up, from the choice of an appropriate distortion measure to the convex formulation belonging to a rich class of MAXDET optimization problems. This is augmented by an efficient algorithm and implementation for solving the positive definite sparse coding problem.

- We explore the effects of atom normalization, sparsity regularization, dictionary size, and data dimension in the sparse coding problem. We show an empirical phase transition diagram, analogous to that used in the vector sparse recovery literature, showing the unique recoverability of the sparse coefficients from a signal processing and compressive sensing viewpoint.

- A novel sparse modeling approach for learning dictionaries of positive definite atoms is presented, and solved using alternating minimization. Iterative descent methods for updating the atoms of the dictionary are shown, and computational improvements are suggested to handle cases of large dimensions.

- Both the sparse coding and dictionary learning methods are extended and efficiently reduced to the case when the dictionary atoms are rank-1 outer products, simplifying many of the computations.

- An approach to learn positive definite sparse models for classification is presented, by incorporating cross-coherence between the atoms of different class dictionaries, analogous to similar vector sparse modeling techniques.

- A software package has been developed, comprising optimized C++ implementations of the algorithms in this thesis, for use by other researchers in our field.

8.2 Future Directions

- The sparsity model used in this thesis can be extended further to incorporate group sparsity and hierarchical sparsity in the coefficients.

- The dictionary learning approach can be used to learn vocabularies for Bag-of-Words (BoW) type features, where the features are positive definite matrices such as region covariances, yielding Bag-of-Covariance-Words (BoCW) features. The sparse coefficients can be summed up and used as the feature vector for further classification with powerful classifiers like SVMs.

- Throughout this thesis, we have assumed the dictionary size to be a user-defined parameter. While heuristic methods can be applied to prune dictionaries based on coherence, it would be of great value to have a way to automatically determine the number of dictionary atoms to model each data class.

- To tackle the scalability of the sparse coding approach to extremely large dictionaries, greedy and/or heuristic techniques to quickly narrow down a subset of dictionary atoms are essential. This would override the need to visit each atom at every iteration of the coordinate descent procedure, improving scalability. Hashing-type approaches based on the atom coherence defined here would be interesting to explore further.

- Just as vector sparse models are used to denoise image patches, the sparse dictionary models from this thesis can be applied to denoising positive definite tensor fields, such as those arising in diffusion tensor imaging.

- Sparse regression on covariance models can be applied towards prediction of dynamic covariances, such as those occurring in the stock market, climate modeling, etc.

- Feature selection is a critical task in many computer vision/image processing applications. While this has been explored in vector descriptors, not much has been done for feature selection in covariance descriptors. Sparse positive definite atoms in the dictionary could enable us to perform feature selection in covariance descriptor data.

References

[Aharon et al., 2006] Aharon, M., Elad, M. and Bruckstein, A. (2006). K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation. IEEE Transactions on Signal Processing *54*, 4311–4322.

[Alahi et al., 2008] Alahi, A., Marimon, D., Bierlaire, M. and Kunt, M. (2008). A master-slave approach for object detection and matching with fixed and mobile cameras. In 15th IEEE International Conference on Image Processing 2008 pp. 1712–1715,.

[Arif and Vela, 2009] Arif, O. and Vela, P. (2009). Kernel covariance image region description for object tracking. In 16th IEEE International Conference on Image Processing, 2009 pp. 865–868,.

[Arsigny et al., 2006] Arsigny, V., Fillard, P., Pennec, X. and Ayache, N. (2006). Log-Euclidean Metrics for Fast and Simple Calculus on Diffusion Tensors. Magnetic Resonance in Medicine *56*, 411–421.

[Artner et al., 2009] Artner, N., Ion, A. and Kropatsch, W. (2009). Coarse-to-Fine Tracking of Articulated Objects Using a Hierarchical Spring System. In Computer Analysis of Images and Patterns, (Jiang, X. and Petkov, N., eds), vol. 5702, of Lecture Notes in Computer Science pp. 1011–1018. Springer Berlin / Heidelberg.

[Austvoll and Kwolek, 2010] Austvoll, I. and Kwolek, B. (2010). Region Covariance Matrix-Based Object Tracking with Occlusions Handling. In Computer Vision and Graphics, (Bolc, L., Tadeusiewicz, R., Chmielewski, L. and Wojciechowski, K., eds), vol. 6374, of Lecture Notes in Computer Science pp. 201–208. Springer Berlin / Heidelberg.

[Banerjee et al., 2005] Banerjee, A., Merugu, S., Dhillon, I. S. and Ghosh, J. (2005). Clustering with Bregman Divergences. J. Mach. Learn. Res. *6*, 1705–1749.

[Bar and Sapiro, 2010] Bar, L. and Sapiro, G. (2010). Hierarchical dictionary learning for invariant classification. In IEEE International Conference on Acoustics Speech and Signal Processing 2010 pp. 3578–3581,.

[Barzilai and Borwein, 1988] Barzilai, J. and Borwein, J. M. (1988). Two-Point Step Size Gradient Methods. IMA Journal of Numerical Analysis *8*, 141–148.

[Bhatia, 2007] Bhatia, R. (2007). Positive Definite Matrices. Princeton University Press, Princeton, NJ, USA.

[Bottou, 1998] Bottou, L. (1998). Online learning in neural networks. chapter Online learning and stochastic approximations, pp. 9–42. Cambridge University Press New York, NY, USA.

[Bregman, 1967] Bregman, L. M. (1967). 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*, 200–217.

[Cai et al., 2010] Cai, Y., Takala, V. and Pietikainen, M. (2010). Matching Groups of People by Covariance Descriptor. In 20th International Conference on Pattern Recognition 2010 pp. 2744–2747,.

[Cargill et al., 2009] Cargill, P. C., Rius, C. U., Quiroz, D. M. and Soto, A. (2009). Performance Evaluation of the Covariance Descriptor for Target Detection. In Intl. Conf. of the Chilean Computer Science Society 2009 pp. 133–141,.

[Castrodad and Sapiro, 2011] Castrodad, A. and Sapiro, G. (2011). Sparse modeling of human actions from motion imagery. Technical Report 2378 Institute for Mathematics and its Applications, University of Minnesota.

[Chen et al., 2001] Chen, S. S., Donoho, D. L. and Saunders, M. A. (2001). Atomic Decomposition by Basis Pursuit. SIAM Rev. *43*, 129–159.

[Davis et al., 1997] Davis, G., Mallat, S. and Avellaneda, M. (1997). Adaptive greedy approximations. Constructive Approximation *13*, 57–98.

[Davis et al., 2007] Davis, J. V., Kulis, B., Jain, P., Sra, S. and Dhillon, I. S. (2007). Information-theoretic metric learning. In 24th Intl. Conf. on Machine learning ICML '07 pp. 209–216, ACM, New York, NY, USA.

[Ding et al., 2010] Ding, X., Huang, C., Huang, F., Xu, L. and fang Li, X. (2010). Region covariance based object tracking using Monte Carlo method. In 8th IEEE International Conference on Control and Automation 2010 pp. 1802–1805,.

[Donoho and Stodden, 2004] Donoho, D. and Stodden, V. (2004). When Does Non-Negative Matrix Factorization Give a Correct Decomposition into Parts? In Advances in Neural Information Processing Systems 16. MIT Press Cambridge, MA.

[Donoho and Tanner, 2009] Donoho, D. and Tanner, J. (2009). Observed universality of phase transitions in high-dimensional geometry, with implications for modern data analysis and signal processing. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences *367*, 4273–4293.

[Donoho and Tanner, 2010] Donoho, D. and Tanner, J. (2010). Precise Undersampling Theorems. Proceedings of the IEEE *98*, 913–924.

[Donoho et al., 2006] Donoho, D., Tsaig, Y., Drori, I. and Starck, J. L. (2006). Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit. Tech. Report. 2006-2 Department of Statistics, Stanford University.

[Donoho and Tanner, 2005] Donoho, D. L. and Tanner, J. (2005). Sparse nonnegative solution of underdetermined linear equations by linear programming. Proceedings of the National Academy of Sciences of the United States of America *102*, 9446–9451.

[Duarte-Carvajalino and Sapiro, 2009] Duarte-Carvajalino, J. and Sapiro, G. (2009). Learning to Sense Sparse Signals: Simultaneous Sensing Matrix and Sparsifying Dictionary Optimization. IEEE Transactions on Image Processing *18*, 1395–1408.

[Efron et al., 2004] Efron, B., Hastie, T., Johnstone, I. and Tibshirani, R. (2004). Least Angle Regression. The Annals of Statistics *32*, 407–451.

[Elad and Aharon, 2006] Elad, M. and Aharon, M. (2006). Image Denoising Via Sparse and Redundant Representations Over Learned Dictionaries. IEEE Transactions on Image Processing *15*, 3736–3745.

[Engan et al., 1999] Engan, K., Aase, S. and Hakon Husoy, J. (1999). Method of optimal directions for frame design. In IEEE International Conference on Acoustics, Speech, and Signal Processing 1999 vol. 5, pp. 2443–2446,.

[Ge and Yu, 2008a] Ge, Y. and Yu, J. (2008a). A scene recognition algorithm based on covariance descriptor. In IEEE Conference on Cybernetics and Intelligent Systems, 2008 pp. 838–842,.

[Ge and Yu, 2008b] Ge, Y. and Yu, J. (2008b). Robust visual tracking using incremental appearance descriptor update. In IEEE International Conference on Automation and Logistics, 2008. pp. 316–321,.

[Gelman et al., 2003] Gelman, A., Carlin, J. B., Stern, H. S. and Rubin, D. S. (2003). Bayesian Data Analysis, Second Edition. Chapman & Hall/CRC, Boca Raton.

[Golub and Loan, 1996] Golub, G. H. and Loan, C. F. V. (1996). Matrix Computations. 3rd edition, The Johns Hopkins University Press.

[Gualdi et al., 2009] Gualdi, G., Prati, A. and Cucchiara, R. (2009). Covariance descriptors on moving regions for human detection in very complex outdoor scenes. In Third ACM/IEEE International Conference on Distributed Smart Cameras, 2009. pp. 1–8,.

[Gualdi et al., 2010] Gualdi, G., Prati, A. and Cucchiara, R. (2010). Perspective and appearance context for people surveillance in open areas. In IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops 2010 pp. 13–18,.

[Guennebaud et al., 2010] Guennebaud, G., Jacob, B. et al. (2010). Eigen v3. http://eigen.tuxfamily.org.

[Guo et al., 2010a] Guo, K., Ishwar, P. and Konrad, J. (2010a). Action Recognition Using Sparse Representation on Covariance Manifolds of Optical Flow. In Seventh IEEE International Conference on Advanced Video and Signal Based Surveillance 2010 pp. 188–195,.

[Guo et al., 2010b] Guo, K., Ishwar, P. and Konrad, J. (2010b). Action change detection in video by covariance matching of silhouette tunnels. In IEEE International Conference on Acoustics Speech and Signal Processing 2010 pp. 1110–1113,.

[Han et al., 2009] Han, Y., Sun, Z., Tan, T. and Hao, Y. (2009). Palmprint Recognition Based on Regional Rank Correlation of Directional Features. In Advances in Biometrics, (Tistarelli, M. and Nixon, M., eds), vol. 5558, of Lecture Notes in Computer Science pp. 587–596. Springer Berlin / Heidelberg.

[Hu et al., 2008] Hu, H., Qin, J., Lin, Y. and Xu, Y. (2008). Region covariance based probabilistic tracking. In 7th World Congress on Intelligent Control and Automation 2008 pp. 575–580,.

[Huo and Feng, 2010] Huo, H. and Feng, J. (2010). Face Recognition via AAM and Multi-features Fusion on Riemannian Manifolds. In Computer Vision ACCV 2009, (Zha, H., Taniguchi, R.-i. and Maybank, S., eds), vol. 5996, of Lecture Notes in Computer Science pp. 591–600. Springer Berlin / Heidelberg.

[Hussein et al., 2009] Hussein, M., Porikli, F. and Davis, L. (2009). A Comprehensive Evaluation Framework and a Comparative Study for Human Detectors. IEEE Transactions on Intelligent Transportation Systems *10*, 417 –427.

[Karasev et al., 2008] Karasev, P., Malcolm, J. and Tannenbaum, A. (2008). Kernel-based high-dimensional histogram estimation for visual tracking. In 15th IEEE International Conference on Image Processing 2008 pp. 2728–2731,.

[Kilic et al., 2010] Kilic, N., Kursun, O. and Ucan, O. (2010). Classification of the Colonic Polyps in CT-Colonography Using Region Covariance as Descriptor Features of Suspicious Regions. Journal of Medical Systems *34*, 101–105.

[Kulis et al., 2006] Kulis, B., Sustik, M. and Dhillon, I. (2006). Learning low-rank kernel matrices. In Proc. 23rd Intl. Conf. on Machine learning ICML '06 pp. 505–512, ACM, New York, NY, USA.

[Kulis et al., 2009] Kulis, B., Sustik, M. A. and Dhillon, I. S. (2009). Low-Rank Kernel Learning with Bregman Matrix Divergences. J. Mach. Learn. Res. *10*, 341–376.

[Kuo et al., 2010] Kuo, C.-H., Huang, C. and Nevatia, R. (2010). Inter-camera Association of Multi-target Tracks by On-Line Learned Appearance Affinity Models. In Computer Vision ECCV 2010, (Daniilidis, K., Maragos, P. and Paragios, N., eds), vol. 6311, of Lecture Notes in Computer Science pp. 383–396. Springer Berlin / Heidelberg.

[Kwolek, 2009] Kwolek, B. (2009). Object Tracking via Multi-region Covariance and Particle Swarm Optimization. In Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance, 2009 pp. 418–423,.

[Kwolek, 2010] Kwolek, B. (2010). Multi Camera-Based Person Tracking Using Region Covariance and Homography Constraint. In Seventh IEEE International Conference on Advanced Video and Signal Based Surveillance 2010 pp. 294–299,.

[Kwon and Park, 2008] Kwon, J. and Park, F. (2008). Visual tracking via particle filtering on the affine group. In International Conference on Information and Automation, 2008 pp. 997–1002,.

[Lee and Seung, 2000] Lee, D. D. and Seung, H. S. (2000). Algorithms for Non-negative Matrix Factorization. In Advances in Neural Information Processing Systems 16, (Leen, T., Dietterich, T. and Tresp, V., eds), vol. 13, pp. 556–562. MIT Press Cambridge, MA.

[Li et al., 2008] Li, X., Hu, W., Zhang, Z. and Zhang, X. (2008). Robust Visual Tracking Based on an Effective Appearance Model. In Computer Vision ECCV 2008, (Forsyth, D., Torr, P. and Zisserman, A., eds), vol. 5305, of Lecture Notes in Computer Science pp. 396–408. Springer Berlin / Heidelberg.

[Löfberg, 2004] Löfberg, J. (2004). YALMIP : A Toolbox for Modeling and Optimization in MATLAB. In Proceedings of the CACSD Conference, Taipei, Taiwan.

[Lu et al., 2009] Lu, J., Zhao, Y. and Hu, J. (2009). Enhanced gabor-based region covariance matrices for palmprint recognition. Electronics Letters *45*, 880 –881.

[Luo and Tseng, 1992] Luo, Z. Q. and Tseng, P. (1992). On the convergence of the coordinate descent method for convex differentiable minimization. Journal of Optimization Theory and Applications *72*, 7–35.

[Ma et al., 2007] Ma, Y., Miller, B. and Cohen, I. (2007). Video Sequence Querying Using Clustering of Objects Appearance Models. In Advances in Visual Computing vol. 4842, of Lecture Notes in Computer Science pp. 328–339. Springer Berlin / Heidelberg.

[Mairal et al., 2009] Mairal, J., Bach, F., Ponce, J. and Sapiro, G. (2009). Online dictionary learning for sparse coding. In Proceedings of the 26th Annual International Conference on Machine Learning ICML '09 pp. 689–696, ACM, New York, NY, USA.

[Mairal et al., 2010] Mairal, J., Bach, F., Ponce, J. and Sapiro, G. (2010). Online Learning for Matrix Factorization and Sparse Coding. J. Mach. Learn. Res. *11*, 19–60.

[Mairal et al., 2008] Mairal, J., Bach, F., Ponce, J., Sapiro, G. and Zisserman, A. (2008). Discriminative learned dictionaries for local image analysis. In IEEE Conference on Computer Vision and Pattern Recognition, 2008 pp. 1–8,.

[Mairal et al., 2009] Mairal, J., Bach, F., Ponce, J., Sapiro, G. and Zisserman, A. (2009). Non-local sparse models for image restoration. In 12th IEEE International Conference on Computer Vision, 2009 pp. 2272–2279,.

[Mairal et al., 2008a] Mairal, J., Elad, M. and Sapiro, G. (2008a). Sparse Representation for Color Image Restoration. IEEE Transactions on Image Processing *17*, 53 –69.

[Mairal et al., 2008b] Mairal, J., Leordeanu, M., Bach, F., Hebert, M. and Ponce, J. (2008b). Discriminative Sparse Image Models for Class-Specific Edge Detection and Image Interpretation. In Proceedings of the 10th European Conference on Computer Vision: Part III pp. 43–56, Springer-Verlag, Berlin, Heidelberg.

[Mairal et al., 2007] Mairal, J., Sapiro, G. and Elad, M. (2007). Multiscale Sparse Image Representation with Learned Dictionaries. In IEEE International Conference on Image Processing, 2007 vol. 3, pp. III –105 –III –108,.

[Mallat and Zhang, 1993] Mallat, S. and Zhang, Z. (1993). Matching pursuits with time-frequency dictionaries. IEEE Transactions on Signal Processing *41*, 3397–3415.

[Martelli et al., 2010] Martelli, S., Tosato, D., Farenzena, M., Cristani, M. and Murino, V. (2010). An FPGA-based Classification Architecture on Riemannian Manifolds. In Workshop on Database and Expert Systems Applications 2010 pp. 215–220,.

[Opelt et al., 2004] Opelt, A., Fussenegger, M., Pinz, A. and Auer, P. (2004). Weak Hypotheses and Boosting for Generic Object Detection and Recognition. In Computer Vision - ECCV 2004, (Pajdla, T. and Matas, J., eds), vol. 3022, of Lecture Notes in Computer Science pp. 71–84. Springer Berlin / Heidelberg.

[Osborne et al., 2000] Osborne, M., Presnell, B. and Turlach, B. (2000). A new approach to variable selection in least squares problems. IMA Journal of Numerical Analysis *20*, 389–403.

[Osman, 2009a] Osman, H. (2009a). Covariance-based recognition using an incremental learning approach. Artificial Life and Robotics *14*, 233–236.

[Osman, 2009b] Osman, H. (2009b). Hardware-Based Solutions Utilizing Random Forests for Object Recognition. In Advances in Neuro-Information Processing, (Kppen, M., Kasabov, N. and Coghill, G., eds), vol. 5507, of Lecture Notes in Computer Science pp. 760–767. Springer Berlin / Heidelberg.

[Paisitkriangkrai et al., 2007] Paisitkriangkrai, S., Shen, C. and Zhang, J. (2007). An Experimental Evaluation of Local Features for Pedestrian Classification. In 9th Biennial Conference of the Australian Pattern Recognition Society on Digital Image Computing Techniques and Applications pp. 53–60,.

[Paisitkriangkrai et al., 2008a] Paisitkriangkrai, S., Shen, C. and Zhang, J. (2008a). Performance evaluation of local features in human classification and detection. IET Computer Vision *2*, 236–246.

[Paisitkriangkrai et al., 2008b] Paisitkriangkrai, S., Shen, C. and Zhang, J. (2008b). Fast Pedestrian Detection Using a Cascade of Boosted Covariance Features. IEEE Transactions on Circuits and Systems for Video Technology *18*, 1140 –1151.

[Paisitkriangkrai et al., 2008c] Paisitkriangkrai, S., Shen, C. and Zhang, J. (2008c). An experimental study on pedestrian classification using local features. In IEEE International Symposium on Circuits and Systems, 2008 pp. 2741–2744,.

[Palaio and Batista, 2008] Palaio, H. and Batista, J. (2008). Multi-object tracking using an adaptive transition model particle filter with region covariance data association. In 19th Intl. Conf. on Pattern Recognition, 2008. pp. 1–4,.

[Palaio and Batista, 2009a] Palaio, H. and Batista, J. (2009a). A kernel particle filter multi-object tracking using gabor-based region covariance matrices. In 16th IEEE International Conference on Image Processing 2009 pp. 4085–4088,.

[Palaio and Batista, 2009b] Palaio, H. and Batista, J. (2009b). Kernel Based Multi-object Tracking Using Gabor Functions Embedded in a Region Covariance Matrix. In Pattern Recognition and Image Analysis, (Araujo, H., Mendona, A., Pinho, A. and Torres, M., eds), vol. 5524, of Lecture Notes in Computer Science pp. 72–79. Springer Berlin / Heidelberg.

[Palaio et al., 2009] Palaio, H., Maduro, C., Batista, K. and Batista, J. (2009). Ground plane velocity estimation embedding rectification on a particle filter multi-target tracking. In IEEE International Conference on Robotics and Automation, 2009 pp. 825–830,.

[Pang et al., 2008] Pang, Y., Yuan, Y. and Li, X. (2008). Gabor-Based Region Covariance Matrices for Face Recognition. IEEE Trans. Circuits Syst. Video Technol. *18*, 989–993.

[Pati et al., 1993] Pati, Y., Rezaiifar, R. and Krishnaprasad, P. (1993). Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. In Twenty-Seventh Asilomar Conference on Signals, Systems and Computers 1993 vol. 1, pp. 40–44,.

[Pennec et al., 2006] Pennec, X., Fillard, P. and Ayache, N. (2006). A Riemannian Framework for Tensor Computing. Int. J. Comput. Vision *66*, 41–66.

[Pfander et al., 2008] Pfander, G., Rauhut, H. and Tanner, J. (2008). Identification of Matrices Having a Sparse Representation. IEEE Trans. Signal Process. *56*, 5376–5388.

[Phillips et al., 2000] Phillips, P., Moon, H., Rizvi, S. and Rauss, P. (2000). The FERET evaluation methodology for face-recognition algorithms. IEEE Trans. Pattern Anal. Mach. Intell. *22*, 1090–1104.

[Porikli, 2010] Porikli, F. (2010). Learning on Manifolds. In Structural, Syntactic, and Statistical Pattern Recognition, (Hancock, E., Wilson, R., Windeatt, T., Ulusoy, I. and Escolano, F., eds), vol. 6218, of Lecture Notes in Computer Science pp. 20–39. Springer Berlin / Heidelberg.

[Porikli and Kocak, 2006] Porikli, F. and Kocak, T. (2006). Robust License Plate Detection Using Covariance Descriptor in a Neural Network Framework. In IEEE Intl. Conf. on Video and Signal Based Surveillance, 2006. pp. 107–107,.

[Porikli and Tuzel, 2006] Porikli, F. and Tuzel, O. (2006). Fast Construction of Covariance Matrices for Arbitrary Size Image Windows. In IEEE Intl. Conf. on Image Processing, 2006 pp. 1581–1584,.

[Porikli et al., 2006] Porikli, F., Tuzel, O. and Meer, P. (2006). Covariance Tracking using Model Update Based on Lie Algebra. In IEEE Comp. Soc. Conf. on Computer Vision and Pattern Recognition 2006 vol. 1, pp. 728–735,.

[Prakash et al., 2007] Prakash, C., Paluri, B., Nalin Pradeep, S. and Shah, H. (2007). Fragments Based Parametric Tracking. In Computer Vision ACCV 2007, (Yagi, Y., Kang, S., Kweon, I. and Zha, H., eds), vol. 4843, of Lecture Notes in Computer Science pp. 522–531. Springer Berlin / Heidelberg.

[Ramirez et al., 2010] Ramirez, I., Sprechmann, P. and Sapiro, G. (2010). Classification and clustering via dictionary learning with structured incoherence and shared features. In IEEE Conference on Computer Vision and Pattern Recognition 2010 pp. 3501–3508,.

[Randen and Husøy, 1999] Randen, T. and Husøy, J. H. (1999). Filtering for Texture Classification: A Comparative Study. IEEE Trans. Pattern Anal. Mach. Intell. *21*, 291–310.

[Rubinstein et al., 2010a] Rubinstein, R., Bruckstein, A. and Elad, M. (2010a). Dictionaries for Sparse Representation Modeling. Proceedings of the IEEE *98*, 1045–1057.

[Rubinstein et al., 2010b] Rubinstein, R., Zibulevsky, M. and Elad, M. (2010b). Double Sparsity: Learning Sparse Dictionaries for Sparse Signal Approximation. IEEE Transactions on Signal Processing *58*, 1553–1564.

[Ruta et al., 2009] Ruta, A., Porikli, F., Watanabe, S. and Li, Y. (2009). In-vehicle camera traffic sign detection and recognition. Machine Vision and Applications *1*, 1–17.

[Schuldt et al., 2004] Schuldt, C., Laptev, I. and Caputo, B. (2004). Recognizing Human Actions: A Local SVM Approach. In 17th International Conference on Pattern Recognition ICPR '04 pp. 32–36,.

[Sharif et al., 2008a] Sharif, M., Ihaddadene, N. and Djeraba, C. (2008a). Covariance Matrices for Crowd Behaviour Monitoring on the Escalator Exits. In Advances in Visual Computing vol. 5359, of Lecture Notes in Computer Science pp. 470–481. Springer Berlin / Heidelberg.

[Sharif et al., 2008b] Sharif, M. H., Martinet, J. and Djeraba, C. (2008b). Object Tracking in Video Using Covariance Matrices. In Encyclopedia of Multimedia, (Furht, B., ed.), pp. 676–679. Springer US.

[Shinohara et al., 2010] Shinohara, Y., Masuko, T. and Akamine, M. (2010). Covariance clustering on Riemannian manifolds for acoustic model compression. In IEEE International Conference on Acoustics Speech and Signal Processing, 2010 pp. 4326–4329,.

[Sivalingam et al., 2009] Sivalingam, R., Morellas, V., Boley, D. and Papanikolopoulos, N. (2009). Metric learning for semi-supervised clustering of Region Covariance Descriptors. In Proc. 3rd ACM/IEEE Intl. Conf. on Distributed Smart Cameras 2009 pp. 1–8,.

[Sivalingam et al., 2012] Sivalingam, R., Somasundaram, G., Li, X., Kaplan, A., Henriksen, J., Banerjee, A., Morellas, V., Papanikolopoulos, N., and Truskinovsky, A. (2012). Diagnosing Adenocarcinoma of the Prostate by Computer Vision Methods. In Annual Meeting of the United States and Canadian Academy of Pathology (USCAP).

[Sivalingam et al., 2011] Sivalingam, R., Somasundaram, G., Ragipindi, A., Banerjee, A., Morellas, V., Papanikolopoulos, N. and Truskinovsky, A. (2011). Diagnosing Endometrial Carcinoma via Computer-Assisted Image Analysis. In Annual Meeting of the United States and Canadian Academy of Pathology (USCAP).

[Sprechmann and Sapiro, 2010] Sprechmann, P. and Sapiro, G. (2010). Dictionary learning and sparse coding for unsupervised clustering. In IEEE International Conference on Acoustics Speech and Signal Processing 2010 pp. 2042–2045,.

[Sra and Cherian, 2011] Sra, S. and Cherian, A. (2011). Generalized dictionary learning for symmetric positive definite matrices with application to nearest neighbor retrieval. In Proc. 2011 European Conf. on Machine Learning and Knowledge Discovery in Databases - Volume Part III pp. 318–332, Springer-Verlag, Berlin, Heidelberg.

[Tenenbaum et al., 2000] Tenenbaum, J. B., Silva, V. and Langford, J. C. (2000). A Global Geometric Framework for Nonlinear Dimensionality Reduction. Science *290*, 2319–2323.

[Tibshirani, 1996] Tibshirani, R. (1996). Regression Shrinkage and Selection via the Lasso. Journal of the Royal Statistical Society (Series B) *58*, 267–288.

[Tou et al., 2009] Tou, J., Tay, Y. and Lau, P. (2009). Gabor Filters as Feature Images for Covariance Matrix on Texture Classification Problem. In Advances in Neuro-Information Processing, (Kppen, M., Kasabov, N. and Coghill, G., eds), vol. 5507, of Lecture Notes in Computer Science pp. 745–751. Springer Berlin / Heidelberg.

[Tropp, 2004] Tropp, J. (2004). Greed is good: algorithmic results for sparse approximation. IEEE Transactions on Information Theory *50*, 2231–2242.

[Tropp, 2006] Tropp, J. (2006). Just relax: convex programming methods for identifying sparse signals in noise. IEEE Trans. Inf. Theory *52*, 1030–1051.

[Tropp and Gilbert, 2007] Tropp, J. and Gilbert, A. (2007). Signal Recovery From Random Measurements Via Orthogonal Matching Pursuit. IEEE Trans. Inf. Theory *53*, 4655–4666.

[Tutuncu et al., 2003] Tutuncu, R. H., Toh, K. C. and Todd, M. J. (2003). Solving semidefinite-quadratic-linear programs using SDPT3. Math. Program. *95*, 189–217.

[Tuzel et al., 2006] Tuzel, O., Porikli, F. and Meer, P. (2006). Region Covariance: A Fast Descriptor for Detection and Classification. In Computer Vision ECCV 2006, (Leonardis, A., Bischof, H. and Pinz, A., eds), vol. 3952, of Lecture Notes in Computer Science pp. 589–600. Springer Berlin / Heidelberg.

[Tuzel et al., 2007] Tuzel, O., Porikli, F. and Meer, P. (2007). Human Detection via Classification on Riemannian Manifolds. In IEEE Conference on Computer Vision and Pattern Recognition, 2007 pp. 1–8,.

[Tuzel et al., 2008] Tuzel, O., Porikli, F. and Meer, P. (2008). Pedestrian Detection via Classification on Riemannian Manifolds. IEEE Trans. Pattern Anal. Mach. Intell. *30*, 1713–1727.

[Vandenberghe et al., 1998] Vandenberghe, L., Boyd, S. and Wu, S.-P. (1998). Determinant Maximization with Linear Matrix Inequality Constraints. SIAM J. Matrix Anal. Appl. *19*, 499–533.

[Wang et al., 2009] Wang, G., Liu, Y. and Shi, H. (2009). Covariance Tracking via Geometric Particle Filtering. In Second International Conference on Intelligent Computation Technology and Automation, 2009 vol. 1, pp. 250–254,.

[Wang et al., 2010] Wang, H., Banerjee, A. and Boley, D. (2010). Modeling Time Varying Covariance Matrices in Low Dimensions. Technical Report TR-10-017 Dept. of Computer Science and Engineering, University of Minnesota.

[Wang and Wu, 2010] Wang, J. and Wu, Y. (2010). Visual Tracking via Incremental Covariance Model Learning. In Second Intl. Conf. on Computer Modeling and Simulation 2010 vol. 1, pp. 277–280,.

[Wang and Yagi, 2008] Wang, J. and Yagi, Y. (2008). Switching local and covariance matching for efficient object tracking. In 19th International Conference on Pattern Recognition, 2008 pp. 1–4,.

[Wang et al., 2012] Wang, L., Li, Y., Jia, J., Sun, J., Wipf, D. and Rehg, J. M. (2012). Learning sparse covariance patterns for natural scenes. To appear in CVPR 2012.

[Wildenauer et al., 2007] Wildenauer, H., Mičušík, B. and Vincze, M. (2007). Efficient texture representation using multi-scale regions. In Proc. 8th Asian Conf. on Computer vision - Volume Part I ACCV'07 pp. 65–74,.

[Wishart, 1928] Wishart, J. (1928). The generalized product moment distribution in samples from a normal multivariate population. Biometrika *20A*, 32–52.

[Wright et al., 2010] Wright, J., Ma, Y., Mairal, J., Sapiro, G., Huang, T. and Yan, S. (2010). Sparse Representation for Computer Vision and Pattern Recognition. Proc. IEEE *98*, 1031–1044.

[Wright et al., 2009] Wright, J., Yang, A., Ganesh, A., Sastry, S. and Ma, Y. (2009). Robust Face Recognition via Sparse Representation. IEEE Trans. Pattern Anal. Mach. Intell *31*, 210–227.

[Wu et al., 2009a] Wu, Y., Wang, J. and Lu, H. (2009a). Robust Bayesian tracking on Riemannian manifolds via fragments-based representation. In IEEE International Conference on Acoustics, Speech and Signal Processing, 2009 pp. 765–768,.

[Wu et al., 2009b] Wu, Y., Wang, J. and Lu, H. (2009b). Real-Time Visual Tracking via Incremental Covariance Model Update on Log-Euclidean Riemannian Manifold. In Chinese Conference on Pattern Recognition, 2009. pp. 1–5,.

[Wu et al., 2008] Wu, Y., Wu, B., Liu, J. and Lu, H. (2008). Probabilistic tracking on Riemannian manifolds. In 19th International Conference on Pattern Recognition, 2008 pp. 1–4,.

[Ye et al., 2008] Ye, C., Liu, J., Chen, C., Song, M. and Bu, J. (2008). Speech Emotion Classification on a Riemannian Manifold. In Advances in Multimedia Information Processing - PCM 2008, (Huang, Y.-M., Xu, C., Cheng, K.-S., Yang, J.-F., Swamy, M., Li, S. and Ding, J.-W., eds), vol. 5353, of Lecture Notes in Computer Science pp. 61–69. Springer Berlin / Heidelberg.

[Yinghui and Jianjun, 2009] Yinghui, G. and Jianjun, Y. (2009). Multi-target tracking using mixed spatio-temporal features learning model. In IEEE International Conference on Automation and Logistics, 2009 pp. 799–803,.

[Yuan et al., 2010] Yuan, C., Hu, W., Li, X., Maybank, S. and Luo, G. (2010). Human Action Recognition under Log-Euclidean Riemannian Metric. In Computer Vision ACCV 2009, (Zha, H., Taniguchi, R.-i. and Maybank, S., eds), vol. 5994, of Lecture Notes in Computer Science pp. 343–353. Springer Berlin / Heidelberg.

[Zheng et al., 2009] Zheng, S., Qiao, H., Zhang, B. and Zhang, P. (2009). The application of intrinsic variable preserving manifold learning method to tracking multiple people with occlusion reasoning. In IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems 2009. pp. 2993–2998,.

[Zheng et al., 2010] Zheng, W., Tang, H., Lin, Z. and Huang, T. S. (2010). Emotion Recognition from Arbitrary View Facial Images. In Computer Vision ECCV 2010, (Daniilidis, K., Maragos, P. and Paragios, N., eds), vol. 6316, of Lecture Notes in Computer Science pp. 490–503. Springer Berlin / Heidelberg.