New Directions in Sparse Models for Image Analysis and Restoration

by

Karthikeyan Natesan Ramamurthy

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved February 2013 by the
Graduate Supervisory Committee:

Andreas Spanias, Chair
Konstantinos Tsakalis
Pavan Turaga
Lina Karam

ARIZONA STATE UNIVERSITY

May 2013

ABSTRACT

Effective modeling of high dimensional data is crucial in information processing and machine learning. Classical subspace methods have been very effective in such applications. However, over the past few decades, there has been considerable research towards the development of new modeling paradigms that go beyond subspace methods. This dissertation focuses on the study of sparse models and their interplay with modern machine learning techniques such as manifold, ensemble and graph-based methods, along with their applications in image analysis and recovery. By considering graph relations between data samples while learning sparse models, graph-embedded codes can be obtained for use in unsupervised, supervised and semi-supervised problems. Using experiments on standard datasets, it is demonstrated that the codes obtained from the proposed methods outperform several baseline algorithms. In order to facilitate sparse learning with large scale data, the paradigm of ensemble sparse coding is proposed, and different strategies for constructing weak base models are developed. Experiments with image recovery and clustering demonstrate that these ensemble models perform better when compared to conventional sparse coding frameworks. When examples from the data manifold are available, manifold constraints can be incorporated with sparse models and two approaches are proposed to combine sparse coding with manifold projection. The improved performance of the proposed techniques in comparison to sparse coding approaches is demonstrated using several image recovery experiments.

In addition to these approaches, it might be required in some applications to combine multiple sparse models with different regularizations. In particular, combining an unconstrained sparse model with non-negative sparse coding is important in image analysis, and it poses several algorithmic and theoretical challenges. A convex and an efficient greedy algorithm for recovering combined representations are proposed. Theoretical guarantees on sparsity thresholds for exact recovery using these

algorithms are derived and recovery performance is also demonstrated using simulations on synthetic data. Finally, the problem of non-linear compressive sensing, where the measurement process is carried out in feature space obtained using non-linear transformations, is considered. An optimized non-linear measurement system is proposed, and improvements in recovery performance are demonstrated in comparison to using random measurements as well as optimized linear measurements.

To

my family and friends

who made this possible ...

ACKNOWLEDGEMENTS

This dissertation would not have been possible without the unflagging guidance and support of my mentors, teachers, friends and family. I consider myself extremely fortunate to have been in the midst of these kind and helpful people.

Among all, I would first like to thank my advisor Dr. Andreas Spanias, who placed enormous trust on me and supported me wholeheartedly throughout my PhD. His faith, confidence, and valuable advice have been a constant source of motivation for me. I am also grateful to my committee members, Dr. Lina Karam, Dr. Konstantinos Tsakalis and Dr. Pavan Turaga for their support and in helping me improve the quality of my work. Furthermore, I have had very fruitful collaborations with Dr. Pavan Turaga and Dr. David Frakes. Their energy and enthusiasm is infectious and the discussions I had with them have been very thought-provoking. My special thanks to Dr. Seungchan Kim, who mentored me during my internship at Tgen, Phoenix. His honest and forthright approach to research is a valuable example to follow. I would also like to thank Dr. Linda Hinnov, with whom I worked on the Java-DSP Earth Systems project. Her patience, attention to detail and work ethic has been a great source of inspiration to me. Jayaraman, Deepta, Prasanna, Rushil, Peter and Berkay have been excellent collaborators in my research and working with them has provided me a lot of new perspectives. They have also been very good friends. Furthermore, I am grateful to all professors, who ever taught me a class here at ASU. I also thank Dr. Joseph Palais, department chair, and the graduate college for supporting my PhD with the teaching assistantship. I also appreciate the help of all the department staff, in particular, Cynthia, Merri, Darleen, Esther, Toni, Jenna, Donna, Ginger and Farah, during my stay here.

My heartfelt gratitude goes to Harini, Vivek, Rooju, Prathap, Dilip, Supraja, Omar, Harish, Mahesh, Suhas and Lakshminarayan for being great friends. I also wish to thank all my labmates over the several years of my stay here, for making my

life at school easier. I thank ASU in general for providing me with excellent facilities and for its progressive atmosphere.

Finally, I have to acknowledge my family for their love, support and trust. Knowing that I can always bank on them has been a great source of strength for me.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Data Exploration using Signal Models

Modern day signal and information processing techniques have a strong focus on efficiently sensing, processing, representing, communicating and performing inference tasks on various types of large quantities of data. This has led to a huge demand for models that can efficiently represent the data generated by various processes, both natural and man-made.

Simple signal models are effective in representing many types of data with a great deal of success. A classic example can be found in the ubiquitous presence of the least squares algorithm [4]. This is because of the reason that a model need not necessarily describe the underlying process completely. It is only sufficient that it describes the signals/data generated by the process. More complicated processes can be described using simpler models as their building blocks. Both deterministic and probabilistic models have been quite commonly used, with their own advantages and disadvantages. We will use the term signal and data interchangeably since the theory and models that we discuss can be applied to both.

General transformations applied to signals can reveal its simple structure in a transformed space, in addition to preserving the information in the signal up to the extent required. Orthonormal transformations such as the Fourier and the Wavelet [5] reveal that the coefficients in the transformed space of many naturally occurring signals and images are sparse, i.e., a few coefficients are large and most others are small. An example of the wavelet transform of the *Lena* image is shown in Figure 1.1. The sparse structure of the coefficients in the vertical (HL), horizontal (LH) and diagonal (HH) subbands at multiple scales are clearly visible. Similar sparse coefficient structure can be seen by decomposing a speech signal using Fourier transform and small regions of natural images using the discrete cosine transform (DCT) [6].

(a) (b)

Figure 1.1: (a) *Lena* image and (b) its transform using the Haar wavelet onto multiple sub-bands at two different scales.

1.2 Low-Dimensional Signal Models

There has been an upsurge in the analysis and applications of sparse models in the past two decades. Although the nature of the signal and transformation applied to it are responsible for the sparsity of the coefficients obtained, explicitly placing sparsity constraints has a lot of advantages. It allows us to control the sparsity by placing additional constraints thereby leading to *approximation* of the signals using its coefficients and this has immense value in the field of compression [7, 8] Enforcing sparsity constraints also helps in recovering data from their incomplete/corrupted observations in applications such as inpainting, compressive recovery, denoising and super-resolution [9–14]. Furthermore, sparse models can also be extended to machine learning applications such as classification [15–18], clustering [19] and transfer learning [20]. In general, we can say that sparse models are simple models that can form fundamental building blocks of more complicated real-world signal/data models. The focus of this dissertation focuses is to study the paradigm of sparse models and their interplay with modern machine learning techniques such as manifold, ensemble

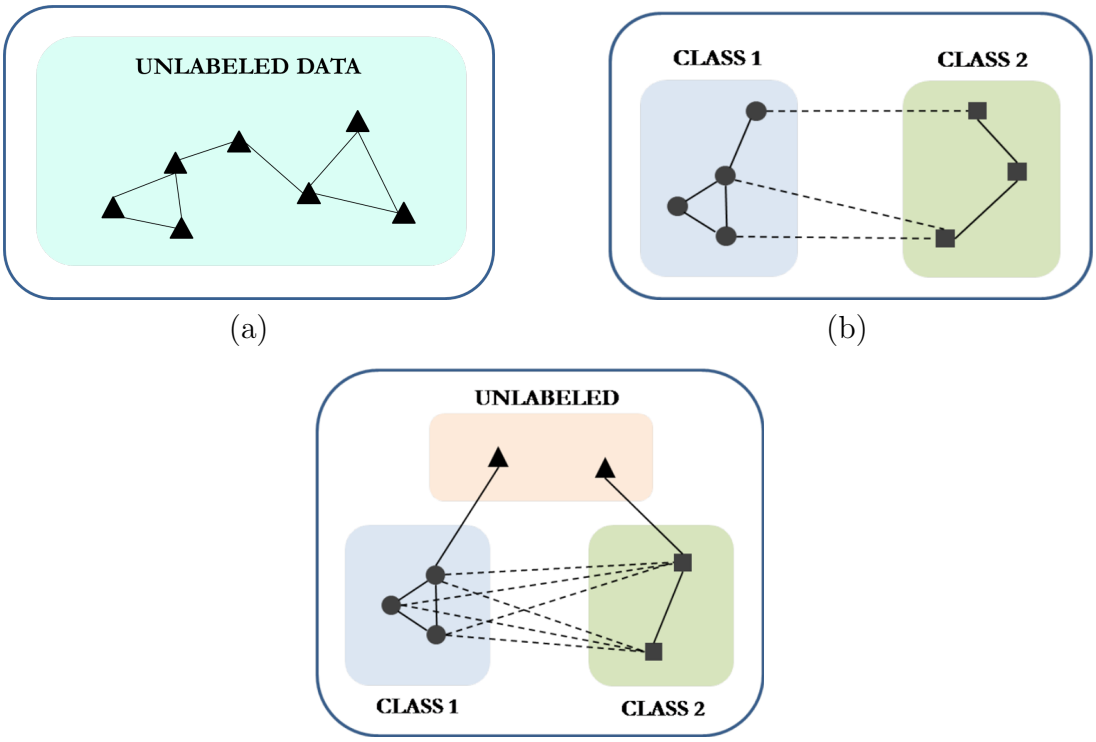
and graph-based methods. The applicability of such methods in image analysis and recovery is also considered.

Analyzing large amounts of high-dimensional data also requires that we need to focus on reducing the data dimensions, when preserving the significant information in it, as determined by the application under consideration. Several dimensionality reduction techniques have been proposed for this purpose, the most fundamental of them being principal components analysis (PCA) [21]. Although PCA is effective in several applications, it is not suited well to applications where the data either lies in a non-linear manifold, or when there is additional class-specific information that can be exploited. In both these cases, there is non-linearity induced in the structure of the data distribution. If the intrinsic structure of the manifold is mathematically defined, the problem of manifold learning reduces to the problem of identifying the intrinsic parameters. However, in the absence of such information, learning methods such as ISOMAP [22], local linear embedding (LLE) [23], Laplacian eigenmaps [24], Hessian eigenmaps [25] can be used to visualize the structure of the structure of the manifold. In fact, sparse coding is also a manifold learning approach, when the data manifold is assumed to be a union of low-dimensional subspaces. Since manifold learning is unsupervised, it is not possible to take advantage of the class-specific discriminative information. This is addressed by posing the pairwise relationship between the data as graphs and using embedding techniques to identify projections which enhance the data discrimination. Example graphs for unsupervised, supervised and semi-supervised learning applications are shown in Figure 1.2. Some of the discriminative graph-embedding techniques include linear discriminant analysis (LDA) [26], local discriminant embedding (LDE) [27], and semi-supervised discriminant analysis (SDA) [28], which can be used for supervised and semi-supervised learning on the data.

When we are faced with the problem of insufficient training data or when we use complex hypotheses for performing inference on the data, the models used

may overfit the data and hence result in a poor performance with novel test data. In this case, instead of learning a single strong hypothesis, we may learn multiple weak hypotheses and aggregate their results to obtain our final inference. The base hypothesis or the base model needs to satisfy only weak conditions, and such ensemble models have a great applicability in supervised learning applications [29]. A basic method for creating an ensemble from multiple base models is to use a Bayesian scheme to weight each hypothesis by its posterior probability. However, methods that manipulate the training samples such as bootstrap aggregation [30] and boosting [31] have been widely used in several learning settings. In these approaches, the training data for each member in the ensemble model is obtained from the original training set either by drawing random samples based on a uniform distribution on the data or by modifying the distribution of the data samples progressively before obtaining the samples. Other approaches to create ensemble models include injecting randomness in the learning algorithm [32], and by manipulating the output targets - a standard method in boosting for regression [33].

In several applications, it may be necessary to use appropriate regularizations in sparse models or even combine several types of regularizations. Assuming non-negativity for the coefficient vector is an important constraint that is very useful in sparse representations. Although this has been used in many applications, theoretical analysis of non-negative representations has been performed only recently [34–38]. Coefficient recovery in non-negative sparse models is better than similar general sparse models, which is an incentive to incorporate non-negative constraints whenever possible. Apart from purely non-negative sparse models, combined sparse models that impose non-negative constraints only on a part of the coefficient vector have also found many applications. Some of the applications of non-negative and combined sparse models are in image inpainting [39], automatic speech recognition using exemplars [40], protein mass spectrometry [41], source separation [42] and clustering/semi-



(a) (b) (c)

Figure 1.2: Examples of graphs used to perform graph embedding, (a) unsupervised graph that considers only the neighborhood relationship between the data, (b) supervised graph that considers the neighborhood relations between data within a class as well as across as well as the classes, (c) semi-supervised graph that encodes class neighborhood information augmented by a set of unlabeled data.

supervised learning of data [19, 43], to name a few. We also note the literature in the areas of non-negative sparse coding and non-negative matrix factorization [44–47], where the idea is to decompose the data into non-negative components, possibly with sparsity constraints.

Certain classes of data cannot be represented efficiently using models in the input Euclidean space. These signals are represented well only after non-linear transformations from the input signal space to a feature space. The non-linear transformations capture the *intrinsic dimensionality* of the data which can be much lesser than its *ambient dimension*. The mathematical form of the non-linear transformation itself can be unknown, but the feature space is a Hilbert space with a well-defined

reproducing kernel. Hence all operations in the feature space can be achieved using the *kernel trick*. Kernel methods are well-known in applications such as embedding of data using kernel principal components [48], classification using kernel SVMs [49] and discriminative clustering [50]. Although the transformation from the input signal space to feature space is defined in terms of the kernel, transforming the data back to input space is usually performed using approximate methods, since such a transformation is not well-defined for all data.

1.3 Problem Statement

1.3.1 Sparse Models with Graph Embedding Constraints

Graph embedding principles [51] can be used to obtain low-dimensional dense embedding of data, which is useful in several learning and inference applications. Sparse models assume that the data lie in a union of low-dimensional linear subspaces and obtain codes assuming that the data are independent of each other. However, incorporating the graph relationship between the data can be helpful in obtaining sparse codes that are useful in several inference applications. A general class of sparse models that incorporate graph relations between the data samples as graph embedding constraints are proposed. These models result in codes that are faithful to the inherent low-dimensional linear subspace structure as well as the graph relations between the data. Since the learning problem is non-convex, two algorithms based on non-convex optimization procedures in order to obtain these codes have also been proposed.

1.3.2 Ensemble Sparse Models for Image Analysis

Ensemble methods that combine several weak hypotheses or models have been very successful in inference applications. The major motivation behind the ensemble approach is that using a single complex hypothesis on data may lead to overfitting, leading to poor generalization with novel test data. It has been well-known that although sparse models are quite powerful in data representation, when recovering

test data that has suffered severe corruption, careful regularization is necessary to obtain a good performance. Furthermore, sparse models are generally obtained using non-convex optimization procedures which lead to suboptimal models. In this dissertation, procedures for learning an ensemble of sparse models to improve the generalization performance with test observations have been proposed.

1.3.3 Combining Sparse Coding with Manifold Projections for Image Recovery

Sparse representations with predefined and learned dictionaries have been used with great success in a number of inverse problems. However, state-of-the-art denoising algorithms such as BM3D [52] and clustering-based sparse representation [53] go beyond the paradigm of sparse representations for images by grouping image patches and jointly processing them. Therefore, utilizing the additional information present in the manifold of training examples may lead to improved recovery performance in general inverse problems. Two models that perform sparse coding and utilize additional manifold regularization using examples are proposed and their performance analyzed in image inpainting and compressive recovery.

1.3.4 Theory of Combined Sparse Representations

General and non-negative sparse representations models have a number of applications in compression, recovery and inference of data. When a subset of the coefficient vector is constrained to be non-negative, it corresponds to a model that combines both non-negative and general sparse representations. For non-negative and general sparse representations, the theoretical and empirical performance guarantees of coefficient recovery are well-studied. There is a need to develop and analyze algorithms that perform coefficient recovery from combined sparse representations, and compare them with non-negative and general sparse recovery algorithms. We propose a convex and a greedy algorithm for coefficient recovery and analyze the theoretical and empirical performance guarantees of such algorithms.

1.3.5 Non-linear Compressive Sensing

Certain types of data are not well-representable using a union of subspaces model, but reveal their low-dimensionality when transformed to non-linear feature spaces. Linear compressive sensing and recovery of such data using sparse models may lead to a poor recovery performance. Non-linear compressive sensing and recovery using kernel random measurements have been proposed in [54]. Since the training data is known, in this work, a procedure to compute an optimized projection system that can further improve the recovery performance for such data has been proposed.

1.4 Contributions

In Chapter 3, the problem of incorporating graph embedding constraints in sparse models is considered. The proposed approach for graph-embedded sparse coding encompasses unsupervised, semi-supervised and supervised learning frameworks. Sparse coding using unsupervised graph embedding constraints is convex, whereas for semi-supervised and supervised models, discriminative constraints are incorporated making the problem non-convex. Two non-convex optimization procedures are developed for obtaining the sparse codes in these cases. Furthermore, an annealing approach for automatically adjusting the discrimination between classes is presented. The sparse models are learned by alternatively computing the sparse codes and adapting the dictionary. The application of these sparse models is demonstrated with several datasets in unsupervised, supervised and semi-supervised learning. Parallel and fast implementations of discriminative sparse coding procedures are coded in C++ with MATLAB interfaces. Some of the methods and results from this chapter have been published in [55].

In Chapter 4, ensemble models for representing and clustering the data using sparse coding are developed and presented. Geometric analysis of the ensemble model is performed and the reasons for better performance of the ensemble model as com-

pared to the individual models are presented and demonstrated. Two approaches for creating the ensemble - one which uses random subsets of the training data as base models, and the other that uses a boosted approach for sequentially learning the base models are presented. The proposed ensemble models are used in demonstrated in data recovery with compressive sensing and super-resolution applications. It is also demonstrated that the proposed model, when used with sparse coding based clustering frameworks, achieve improved clustering performances. The work discussed in this chapter has been reported in [56] and [57].

In Chapter 5, the problem of recovering data from incomplete/noisy observations, when the samples follow a combination of the union of subspaces model and a non-linear manifold model is considered. Since the manifold does not have a closed-form approximation, the training samples from the manifold themselves are used to approximate it. The first approach proposed regularizes the sparse codes obtained with a predefined dictionary using manifold examples. The second approach improves on the previous model in terms of computational complexity and performance by directly combining sparse coding and manifold projection is proposed for incomplete/noisy data recovery. Manifold projection is implemented using non-negative sparse coding on examples in the neighborhood of test data. Inpainting and compressive recovery of standard images is performed using the proposed models with a predefined DCT dictionary and density filtered natural image patches as manifold examples. For high undersampling/noise level, the proposed models perform better than using just sparse coding with predefined/learned dictionaries. A part of the work reported in this chapter has been published in [39].

In Chapter 6, the non-negative sparse representation model and the combined representation model are considered. The geometry of the non-negative representation model is analyzed and the sparsity threshold for unique recovery with and without the presence of an explicit sparsity regularization is derived. In the combined

representation model, there are special cases when the non-zero coefficient support of either or both the general and the non-negative components are known. These are analyzed and the corresponding thresholds for unique recovery of such representations are derived. When both the non-negative and the general coefficient supports are unknown, two algorithms that use convex and greedy procedures for coefficient recovery are proposed. The sparsity thresholds that lead to unique coefficient recovery using these procedures are derived. Finally, the empirical performances the proposed combined coefficient recovery algorithms are compared with the performances of the general and non-negative coefficient recovery procedures. It is demonstrated that incorporating knowledge about the sign of the coefficients leads to improved coefficient recovery. The algorithms, analysis, and experiments in this chapter have been reported in [58].

In Chapter 7, an optimized measurement system for non-linear compressive sensing of certain types of data, which are well-represented in a non-linearly transformed feature space, is proposed. It is evaluated for non-linear compressive sensing of handwritten digit and sculpted face datasets, and is shown to perform better than a random measurement system. Methods and results provided in this chapter have been published in [59].

1.5 Notation

Lowercase boldface letters denote column vectors and uppercase boldface denote matrices, e.g., $\mathbf{a}$ and $\mathbf{A}$ denote a vector and a matrix respectively. $\mathbf{a}_i$ indicates the i^{th} column of the matrix $\mathbf{A}$. The Moore-Penrose pseudoinverse of a matrix $\mathbf{A}$ is denoted by $\mathbf{A}^\dagger = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$. $diag(\mathbf{a}) = \mathbf{A}$ means that $\mathbf{A}$ is a diagonal matrix with the elements in the vector $\mathbf{a}$ as its diagonal elements. $|\mathbf{A}|$ refers to a matrix whose elements are the absolute values of the elements of $\mathbf{A}$ and the same notation applies to vectors also. The maximum row sum and maximum column sums of $\mathbf{A}$ are referred to as $\|\mathbf{A}\|_{\infty,\infty}$ and $\|\mathbf{A}\|_{1,1}$ respectively. A set is denoted as $\mathcal{A}$, its cardinality is given by

$|\mathcal{A}|$ and its complement by $\mathcal{A}^c$. The operator $[.]^+$ or $\max(., 0)$ returns the maximum of the argument and zero. The operator $\max(., \mathbf{0})$ will be used with vectors also, in which case, there will be an element-by-element comparison with the zero vector and the output will be a vector. $\mathbf{I}_K$ denotes an identity matrix of size $K \times K$, $\mathbf{1}_{K_1,K_2}$ is a matrix of ones with size $K_1 \times K_2$. Similar notation will for defining vectors also. When it is clear from context, the subscripts will be dropped for simplicity. The abbreviation WLOG expands to *without loss of generality*.

Chapter 2

BACKGROUND

2.1 Sparse Models

Several applications in data analysis aim to express the signals in the most parsimonious terms. Let us consider the linear generative model,

$$\mathbf{y} = \mathbf{D}\boldsymbol{\beta}, \tag{2.1}$$

where $\mathbf{y} \in \mathbb{R}^M$ is the data to be represented, $\mathbf{D} \in \mathbb{R}^{M \times K}$ is the dictionary that contains the set of elementary patterns and $\boldsymbol{\beta} \in \mathbb{R}^K$ is the coefficient vector. Sparsity can be imposed by placing the appropriate penalty on the coefficient vector $\boldsymbol{\beta}$. The straightforward function used to measure the sparsity is the ℓ_0 norm of the coefficient vector denoted as $\|\boldsymbol{\beta}\|_0$. In lieu of the exact ℓ_0 penalty, which is combinatorial, its convex surrogate, the ℓ_1 penalty can be used to compute the sparse codes. The ℓ_0 and ℓ_1 minimization problems are expressed as

$$\min_{\boldsymbol{\beta}} \|\boldsymbol{\beta}\|_0 \text{ subj. to } \mathbf{y} = \mathbf{D}\boldsymbol{\beta}, \tag{2.2}$$

$$\min_{\boldsymbol{\beta}} \|\boldsymbol{\beta}\|_1 \text{ subj. to } \mathbf{y} = \mathbf{D}\boldsymbol{\beta}. \tag{2.3}$$

We refer to the solution obtained from (2.2) as the ML_0 solution and the one obtained from (2.3) as the ML_1 solution. Since real-world data usually contains some amount of noise that is not expressed in the generative model in (2.1), the equality constraints in (2.2) and (2.3) are replaced using the constraint $\|\mathbf{y} - \mathbf{D}\boldsymbol{\beta}\|_2^2 \leq \epsilon$. Here, ϵ denotes the error goal of the representation which depends on the additive noise. In fact, any penalty function from the set $\{\|\boldsymbol{\beta}\|_p \mid 0 < p \leq 1\}$, can be shown to be promote sparsity. The shape of the unit ℓ_p balls which are level sets defined by

$$\left(\sum_{i=1}^{K} |\beta[i]|^p\right)^{1/p} = 1 \tag{2.4}$$

are shown in Figure 2.1 for various values of p. Note that the $p = 0$ cannot be used in (2.4), since the ℓ_0 norm, that counts the number of non-zero coefficients, is only a

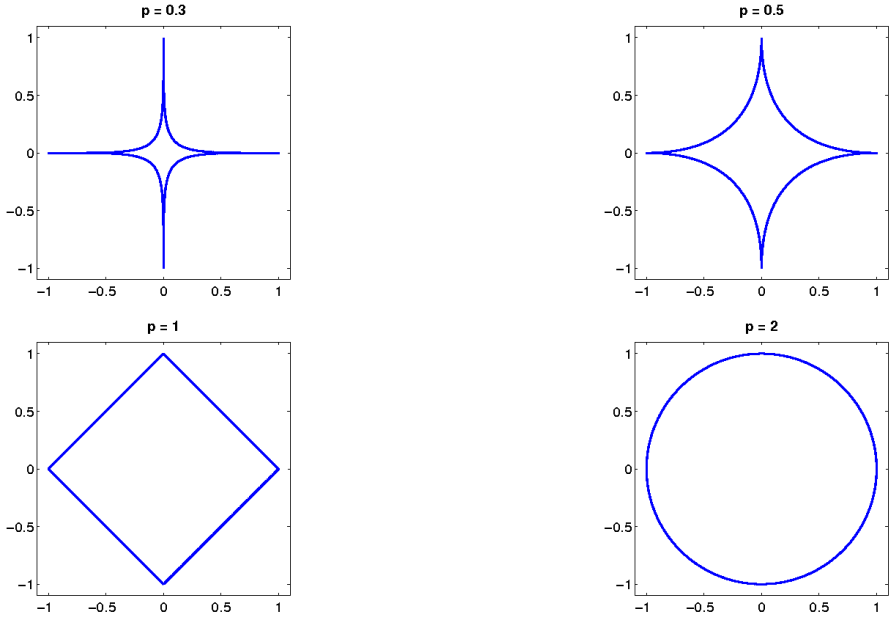


Figure 2.1: Unit ℓ_p balls for $p = 0.3, 0.5, 1, 2$ (clockwise from top left). Note that the only ℓ_p ball that is sparsity promoting and convex is the ℓ_1

pseudonorm. The optimization given in (2.3) can be visualized as the expansion the ℓ_1 ball until it touches the affine feasible set $\mathbf{y} = \mathbf{D}\boldsymbol{\beta}$. Considering the various unit balls in Figure 2.1, it can be shown that all points in the ℓ_2 ball have an equal probability of touching an arbitrary affine feasible set, and hence the solution is almost always dense. However, the balls with $p \leq 1$ have a high probability of touching the feasible set at points where most of the coordinates are zero, leading to sparse solutions with high probability. In the rest of this chapter, we restrict our discussion to ℓ_0 and ℓ_1 norms.

2.1.1 Other Sparsity Regularizations

The penalized version of the ℓ_1 minimization problem in (2.3) can be represented as

$$\min_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbf{D}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1, \tag{2.5}$$

where the parameter λ trades off the error and the sparsity of the representation. Suppose there are groups of dictionary atoms that have a high correlation, the ℓ_1

minimization chooses only one of them and ignores the others. This leads to an unstable representation, as signals that are close in space could have very different representations. Modifying the problem as

$$\min_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbf{D}\boldsymbol{\beta}\|_2^2 + \lambda_1\|\boldsymbol{\beta}\|_1 + \lambda_2\|\boldsymbol{\beta}\|_2, \tag{2.6}$$

we have the elastic net regularization that is a combination of the sparsity prior and the ridge-regression penalty [60]. This ridge regression penalty ensures that correlated dictionary atoms are picked together in the solution leading to a stabler representation. If we have the knowledge that dictionary atoms contribute to the representation in groups, this can be exploited by posing the group lasso problem [61]

$$\min_{\boldsymbol{\beta}} \|\mathbf{y} - \sum_{l=1}^{L} \mathbf{D}_l\boldsymbol{\beta}_l\|_2^2 + \lambda \sum_{l=1}^{L} \sqrt{p_l}\|\boldsymbol{\beta}_l\|_2, \tag{2.7}$$

where L is the number of groups, p_l depends on the size of the group, $\mathbf{D}_l$ is the $l^{\mathbf{th}}$ dictionary group and $\boldsymbol{\beta}_l$ is the $l^{\mathbf{th}}$ coefficient group. The summation term acts like a sparsity constraint at the group level and the all dictionary atoms in a group are either chosen or neglected together.

In the sparsity regularization considered so far,the data samples are considered independent and hence the relationship between them are not taken into account. In most cases, real-world data have a strong relationship between each other and utilizing this can be of significant help in stabilizing the representations for inverse problems. Consider the scenario where we have several data samples that are in the same subspace spanned by a group of dictionary atoms, and hence have the same sparsity support. This generalization of simple approximation to the case of several input signals is referred to as simultaneous sparse approximation (SSA). Given several input signals, we wish to approximate all the signals at once using different linear combinations of the same set of elementary signals. The coefficient vectors for sparse coding and SSA are given in Figure 2.2, where it can be seen clearly that SSA prefers

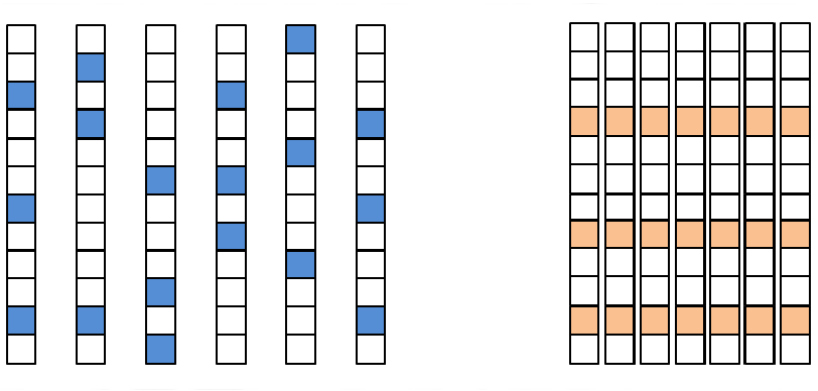


Figure 2.2: The coefficient vectors chosen in sparse coding for a group of signals (left) and the coefficient vectors for simultaneous sparse approximation (right). The shaded cells represent non-zero coefficients.

to choose the same dictionary for all data samples in the group. For a dictionary $\mathbf{D}$ and the set of input vectors $\mathbf{Y}$, the SSA can be obtained by solving

$$\min_{\mathbf{B}} \sum_{i=1}^{k} \|\boldsymbol{\beta}^i\|_q^p \text{ subj. to. } \|\mathbf{Y} - \mathbf{DB}\|_F^2 \leq \epsilon, \tag{2.8}$$

where $\boldsymbol{\beta}^i$ denotes the i^{th} row of the coefficient matrix $\mathbf{B}$. The value for the pair (p, q) is chosen as $(1, 2)$ or $(0, \infty)$, and the former leads to a convex norm, whereas the latter actually counts the number of non-zero rows in $\mathbf{B}$.

2.1.2 Sparse Representations on Pairs of Dictionaries

When a dictionary $\mathbf{D}$ is a combination of two different sub-dictionaries $\mathbf{D}_1 \in \mathbb{R}^{M \times K_1}$ and $\mathbf{D}_2 \in \mathbb{R}^{M \times K_2}$ such that $\mathbf{D} = [\mathbf{D}_1 \ \mathbf{D}_2]$, the representation of the data $\mathbf{y}$ is given by

$$\mathbf{y} = \mathbf{D}_1\boldsymbol{\beta}_1 + \mathbf{D}_2\boldsymbol{\beta}_2, \tag{2.9}$$

where $\boldsymbol{\beta}_1 \in \mathbb{R}^{K_1}$ and $\boldsymbol{\beta}_2 \in \mathbb{R}^{K_2}$ are the coefficient vectors. Let us denote the coefficient vector for $\mathbf{D}$ as $\boldsymbol{\beta}$. Several recent papers [62, 63] have focussed on analyzing the sparsity thresholds for unique recovery of the coefficients from the data $\mathbf{y}$, when $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$ are general sparse vectors. By considering the coherence parameters of $\mathbf{D}_1$ and $\mathbf{D}_2$ separately, an improvement up to a factor of two can be achieved in the sparsity

threshold when compared to considering $\mathbf{D}_1$ and $\mathbf{D}_2$ together as a single dictionary. Apart from deterministic sparsity thresholds, probabilistic or robust thresholds, i.e., thresholds that hold for most sparsity patterns and non-zero values of $\boldsymbol{\beta}$ have also been derived [62]. In addition, bounds on coefficient recovery error in cases of approximate sparsity and noisy observations have also been derived.

2.1.3 Non-negative Sparse Representations

The representations considered so far had no constraints on the signs of the coefficients. However there are several applications, where we require the model to be *strictly additive*, i.e., the coefficients need to be of the same sign. The underdetermined system of linear equations with the constraint that the solution is non-negative can be expressed as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\alpha}, \text{ such that } \boldsymbol{\alpha} \geq 0, \tag{2.10}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^{K_x}$ is the non-negative coefficient vector, and $\mathbf{X} \in \mathbb{R}^{M \times K_x}$ is the dictionary with $K_x > M$. Under the constraint that the solution is sparse the solution can be computed using an optimization program similar to (2.2) with the additional non-negativity constraint incorporated. We will refer to the solution obtained using this procedure as the $ML_0 - NN$ solution. Similar to (2.3), the convex program used to solve for the non-negative coefficient vector is given as

$$\min_{\boldsymbol{\alpha}} \mathbf{1}^T \boldsymbol{\alpha} \text{ subject to } \mathbf{y} = \mathbf{X}\boldsymbol{\alpha}, \boldsymbol{\alpha} \geq \mathbf{0}. \tag{2.11}$$

If the set

$$\{\boldsymbol{\alpha} | \mathbf{y} = \mathbf{X}\boldsymbol{\alpha}, \boldsymbol{\alpha} \geq 0\} \tag{2.12}$$

contains only one solution, any variational function on $\boldsymbol{\alpha}$ can be used to obtain the solution [35, 36] and ℓ_1 minimization is not required.

2.1.4 Geometrical Interpretation

The generative model indicated in (2.1) with sparsity constraints is a non-linear model, because the set of all $S-$sparse vectors is not closed under addition. The

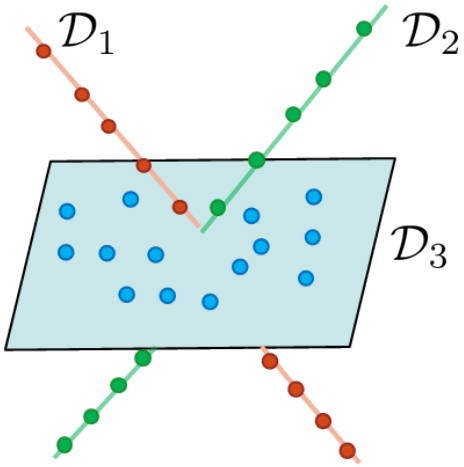


Figure 2.3: The union of subspaces model; the data lies in the union of two $1-$D subspaces ($\mathcal{D}_1$ and $\mathcal{D}_2$), and one $2-$D subspace ($\mathcal{D}_3$).

sum of two $S-$ sparse vectors generally results in a $2S-$sparse vector. An example of the sparse model in an example $3-$dimensional case is given in Figure 2.3, where it can be seen that the sum of data samples from $\mathcal{D}_1$ and $\mathcal{D}_2$, does not lie in a $1-$dimensional subspace. Clearly, sparse models are generalizations of linear subspace models since each sparse pattern represents a subspace, and the union of all patterns represent a union of subspaces. Considering $S-$sparse coefficient vectors obtained from a dictionary of size $M \times K$, the data samples $\mathbf{y}$ obtained using the model (2.1) lie in a union of $\binom{K}{S}$ $S-$dimensional subspaces. In the case of non-negative sparse model given by (2.10), for $S-$sparse representations, the data samples lie in a union of $\binom{K}{S}$ simplical cones. Given the a subset $\mathbf{D}_\Omega$ of dictionary atoms, where Ω is the set of S indices corresponding to the non-zero coefficients, the simplical cone generated by the atoms is given by

$$\left\{ \sum_{j \in \Omega} \alpha[j] \mathbf{x}_j \mid \alpha[j] > 0 \right\}. \tag{2.13}$$

Note that a simplical cone is a subset of the subspace spanned by the same atoms.

It is instructive to compare the combinatorial complexity, pertaining to ℓ_0 minimization, for computing general and non-negative sparse representations. For a

general representation, we need to identify both the support and the sign pattern, whereas for a non-negative representation, identification of support alone is sufficient. The complexity of identifying the support alone for a $S-$sparse representation is $\binom{K}{S}$, and identification of sign pattern along with the support incurs a complexity of $\binom{2K-S}{S}$. This is because, there are $2K$ signs to choose from and we cannot choose both positive and negative signs for the same coefficient. The comparison of these complexities for general and non-negative sparse representations is shown in Figure 2.4. It is clear that the complexity increases as the number of coefficients increase and the non-negative sparse representation is far less complex compared to the general representation, in terms of absolute complexity. Although this gives us an idea about the combinatorial complexity, the general representation actually incurs lesser complexity both with practical convex and greedy optimization procedures. The reason for this is that including additional constraints such as non-negativity in an optimization problem usually increases the computational complexity for arriving at an optimal solution.

Although the geometric interpretation presented above considers the combinatorial complexity, in practical cases, we need to ensure that the ℓ_1 solution obtained using (2.3) is equivalent to the ℓ_0 solution given by (2.2) [64, 65]. This equivalence completely depends on the properties of the dictionary $\mathbf{D}$. Assuming that the dictionary atoms are normalized to unit ℓ_2 norm, let us define the Gram matrix for the dictionary as $\mathbf{G} = \mathbf{D}^T\mathbf{D}$ and denote the coherence as the maximum magnitude of the off-diagonal elements

$$\mu = \max_{i \neq j} |g_{i,j}|. \tag{2.14}$$

It can be inferred that the representation obtained from ℓ_0 minimization is unique and equivalent to ℓ_1 minimization if

$$\|\mathbf{a}\|_0 \leq \frac{1}{2}\left(1 + \frac{1}{\mu}\right). \tag{2.15}$$

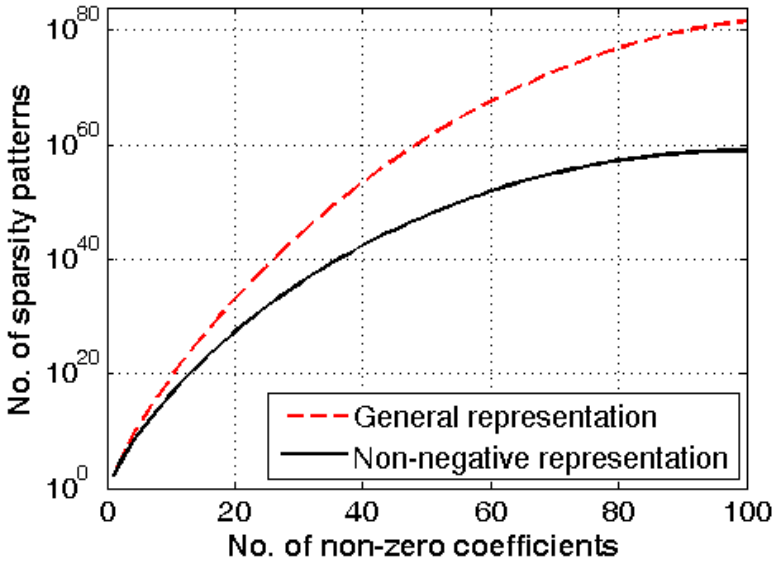


Figure 2.4: The number of patterns to be searched for if ℓ_0 minimization is used to compute sparse representations. Size of the dictionary is 100×200. Computing non-negative representations always incurs lesser complexity compared to general representations.

2.1.5 Sparsity Thresholds

This is referred to as the *deterministic sparsity threshold*, since it holds true for all sparsity patterns and non-zero values in the coefficient vectors. This threshold is illustrated in Figure 2.5 for various values of μ. Since the general representation encompasses the non-negative case also, the same bound holds true for non-negative sparse representations as well. Furthermore, the threshold is the same for ℓ_1 minimization as well as greedy recovery algorithms such as the Orthogonal Matching Pursuit (OMP). The deterministic sparsity threshold scale at best as $\sqrt{M}$ as M increases. Probabilistic or Robust sparsity thresholds, on the other hand scale in the order of $M/\log K$ [66] and break the square-root bottleneck. However, the trade-off is that the unique recovery using ℓ_1 minimization is only assured with high probability, and robust sparsity thresholds for unique recovery using OMP are also unknown.

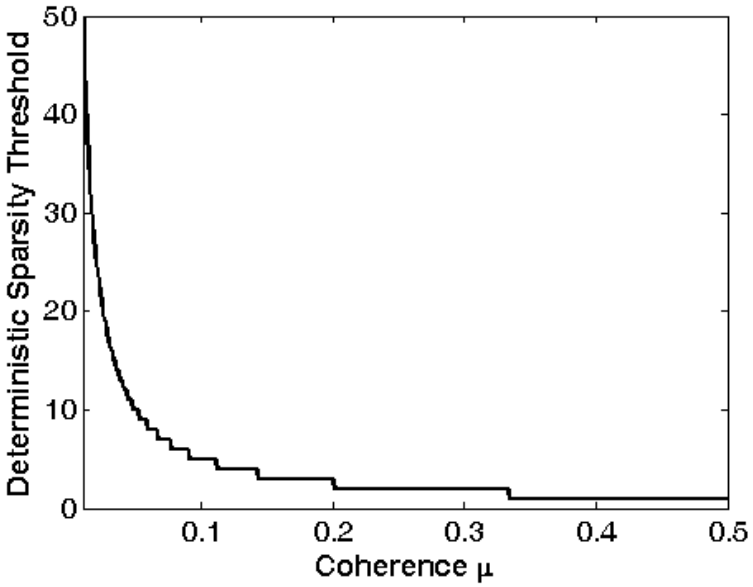


Figure 2.5: Deterministic sparsity threshold with respect to the coherence of the dictionary.

2.1.6 Phase Transitions

Deterministic thresholds are too pessimistic, and in reality the performance of sparse recovery is much better than that predicted by the theory. Robust sparsity thresholds are better but still restrictive as they are not available for several greedy sparse recovery algorithms. Phase transition diagrams describe sparsity thresholds at which the recovery algorithm transitions from a high probability of success to high probability of failure for various values of the ratio M/K (undersampling factor) ranging from 0 to 1. For random dictionaries and coefficient vectors whose entries are realized from various probability distributions, empirical phase transitions can be computed by finding the points at which fraction of success for sparse recovery is 0.5 with respect to a finely spaced grid of sparsity and undersampling factors [67].

Asymptotic phase transitions can be computed based on the theory of polytopes [35,37,67,68]. For $K \to \infty$ when the dictionary entries are derived from $\mathcal{N}(0,1)$ and the non-zero coefficients are signs (± 1) asymptotic phase transitions are shown in

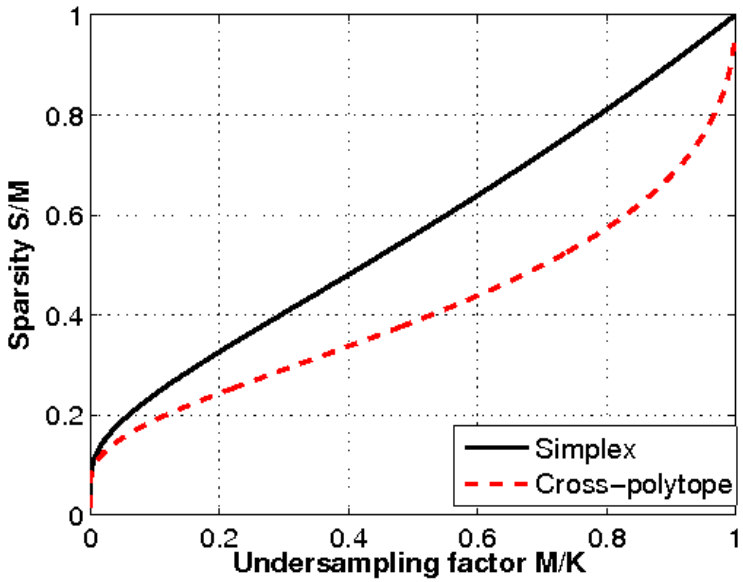


Figure 2.6: Asymptotic phase transitions for simplex and cross-polytope when the dictionary elements are derived from i.i.d. Gaussian $\mathcal{N}(0,1)$ and non-zero coefficients are signs.

Figure 2.6 for ℓ_1 minimization algorithms. It can be seen that imposing non-negativity constraint gives an improved phase transition when compared to not having it. Furthermore, empirical phase transitions computed for Rademacher, partial Hadamard, Bernoulli and random Ternary ensembles are similar to those computed for Gaussian i.i.d. ensembles [67]. The phase transitions of several greedy recovery algorithms have been analyzed and presented in [69] and optimal tuning of sparse approximation algorithms that use iterative thresholding have been performed by studying their phase transition characteristics [70].

2.1.7 Learning the Sparse Model

When the dictionary $\mathbf{D}$ that represents the data itself is unknown and only a set of T training data samples $\mathbf{Y} = [\mathbf{y}_i]_{i=1}^{T}$ are known, the dictionary can be learned from the training data. Given a sufficient number of training examples, the learned dictionary is a good representative of key features present in the data. This is an unsupervised

problem and the cost function used promotes dictionaries which generate a representation having a small residual error with the training data. We will state the general problem of dictionary learning in terms of minimizing the representation error as

$$\min_{\mathbf{D},\mathbf{B}} \|\mathbf{Y} - \mathbf{D}\mathbf{B}\|_F^2 \text{ s.t. } \|\boldsymbol{\beta}_i\|_0 \leq S_d, \|\mathbf{d}_j\|_2 = 1, \forall i, j \tag{2.16}$$

Several dictionary learning algorithms have been proposed in the literature [11–13, 71–77], some of them tailored to specific applications but the fundamental goal is to identify the basis for the union of subspaces where the data resides. This leads to improved performances in data compression, recovery and inference, where sparse models are widely used.

2.2 Low-dimensional Manifold Models

Low dimensional manifold models find applications in representation and processing of natural signals and images. Intrinsic low-dimensionality of data is necessary to learn simple models with less number of parameters, which can in-turn help in efficient data representation, recovery and inference. Here, we use the term manifold to refer to low-dimensional surfaces in a high-dimensional space, that can be triangulated. This is in contrast to the sparse coding model, where the data was assumed to be lying in a union of subspaces.

2.2.1 Principal Components Analysis

Before jumping into the various manifold learning approaches, we will first describe the well-known principal components analysis method, which will provide us with valuable insights into various data modeling approaches. In PCA, a set of orthogonal projection directions are computed such that the projected data is decorrelated [21]. This is achieved by performing an eigen analysis on the covariance matrix and choosing the projection matrix as the maximum variance directions. For the data samples given by $\mathbf{Y} = [\mathbf{y}_1 \dots \mathbf{y}_T]$, let us denote the mean as $\overline{\mathbf{y}}$, and the covariance matrix as $\mathbf{C}$. The d orthogonal projection directions are given by the matrix $\mathbf{V}$, which are

chosen as the eigenvectors corresponding to the d largest eigenvalues of $\mathbf{V}$. For a test data sample denoted as $\mathbf{u}$, the PCA coefficients are given as $\mathbf{V}^T(\mathbf{u}-\overline{\mathbf{y}})$, and the projection is given by $\mathbf{V}\mathbf{V}^T(\mathbf{u}-\overline{\mathbf{y}})+\overline{\mathbf{y}}$. Note that mean of the training samples must be removed from test data before projection and added back after projection. From the discussion above, it can be inferred that PCA is a *global* model that identifies the maximum variance subspace, such that the isometry of the data is approximately preserved. PCA assumes that the data is generated from a Gaussian distribution. Gaussian is a maximum entropy prior distribution only when we have do not have any idea about the data generating process. However, in many cases we do have some knowledge about the data generating process. As it can be seen in the discussions that follow, even a very primitive knowledge that the data is distributed in a locally linear fashion, leads to much more sophisticated and useful modeling paradigms.

2.2.2 Manifold Learning and Dimensionality Reduction

We will start with the assumption that the data is sampled out of a smooth and embeddable manifold which means that locally the manifold can be treated like an Euclidean space. Furthermore, locally the manifold is assumed to be of much lower dimensions compared to the ambient dimension M. We will not venture into exact mathematical definitions and [78] is an excellent reference for this. Since manifold learning is must take into account the local structure of the manifold, there is a need for us to move away from the PCA model. The learning paradigm must take into account the local similarities between the data samples. Typically, we do not have a complete description of the manifold, rather we only have a finite number of samples obtained from it. Let us denote the T samples from the manifold $\mathcal{M}$ by the columns of the matrix $\mathbf{Y}$, which are also referred to as examples.

One of the first approaches that attempted to create an embedding of data considering their pairwise dissimilarities is multidimensional scaling. Let us denote

the dissimilarities between two training samples i and j as $\delta_{i,j}$, and let the embedding of a sample $\mathbf{y}_i$ be $\mathbf{q}_i$. Let $\mathbf{\Delta}$ be the matrix containing the pairwise dissimilarities $\delta_{i,j}$ and $\mathbf{\Delta}_Q$ be the matrix containing the pairwise distances between embeddings, such that $\mathbf{\Delta}_Q(i,j) = \|\mathbf{q}_i - \mathbf{q}_j\|_2$. MDS is attempts to find the embeddings by minimizing the following objective,

$$\|\mathbf{\Delta} - \mathbf{\Delta}_Q\|_F^2, \tag{2.17}$$

where $\|.\|_F^2$ denotes the Frobenius norm of the matrix. Denoting the matrix containing the top d eigenvectors of the matrix $\mathbf{\Delta}$ to be $\mathbf{V}$, the global minimum for (2.17) is obtained by setting $\mathbf{q}_i$ as the i^{th} row of $\mathbf{V}$.

ISOMAP [22] is a global manifold learning algorithm that uses MDS to obtain a low-dimensional embedding. The embedding is obtained such that the intrinsic manifold distances between the samples in the manifold are preserved in the embedding. In other words, the pairwise dissimilarity $\delta_{i,j}$ is set as the shortest distance between two points as we traverse along the manifold, which is also referred to as the *geodesic distance*. The ISOMAP algorithm expresses the manifold as a graph constructed by connecting the neighboring data samples and uses the Djikstra's algorithm to estimate the geodesic distance. The MDS cost given in (2.17) is then minimized to obtain the embedding. Note that this algorithm attempts to preserve *all* pairwise geodesic distances and hence can be termed as global.

Since smooth manifolds can be thought of as locally similar to Euclidean spaces, it is possible to embed them with low-distortion considering only the local linear structure. Locally linear embedding [23] is a dimensionality reduction approach that computes an embedding using this idea. It proceeds in three steps: (a) building a neighborhood for each sample, (b) finding the weights that linearly approximate the sample using its neighbors and (c) finding the low-dimensional coordinates that are best reconstructed using those weights. Given the desired number of dimensions

d, the algorithm can be formally described as:

1. Find the k nearest neighbors $\mathcal{N}_k$ for $\mathbf{y}_i$,

2. Find the weight matrix $\mathbf{W}$ that results in a low-distortion reconstruction of $\mathbf{y}_i$ from its neighbors by optimizing,

$$\min_{\mathbf{W}} \sum_{i=1}^{T} \|\mathbf{y}_i - \sum_{j \neq i} w_{ij} \mathbf{y}_j\|_2^2, \tag{2.18}$$

   where $w_{ij} = 0$ if $j \notin \mathcal{N}_k(i)$, and $\sum_j w_{ij} = 1$.

3. Find the embedding that minimizes the reconstruction error using the weights obtained,

$$\min_{\mathbf{Q}} \sum_{i=1}^{T} \|\mathbf{q}_i - \sum_{j \neq i} w_{ij} \mathbf{q}_j\|_2^2, \tag{2.19}$$

   such that the embedding is centered, and has a unit covariance.

Once the weights are computed, the embedding can be obtained by performing an eigen decomposition of the matrix $(\mathbf{I}-\mathbf{W})^T(\mathbf{I}-\mathbf{W})$ and choosing the k bottom eigen vectors, ignoring the last which is a constant $\mathbf{1}$ vector. If we denote this eigen vector set by the matrix $\mathbf{V}$, the embedding $\mathbf{q}_i$ corresponds to the i^{th} row of this matrix.

A closely related method to LLE that attempts to compute low-dimensional embeddings using the locality information is the Laplacian eigenmaps [24]. We begin by considering the neighbors of each data sample, obtained either using the $k-$neighborhood or the $\epsilon-$neighborhood and create a binary affinity matrix, whose entries are given by

$$w_{ij} = \begin{cases} 1 & \text{if } i \in \mathcal{N}_k(j) \text{ OR } j \in \mathcal{N}_k(i), \\ 0 & \text{otherwise.} \end{cases} \tag{2.20}$$

Note that the weights for the neighbors can also be computed using the heat kernel

$$w_{ij} = \exp^{-\gamma \|\mathbf{y}_i - \mathbf{y}_j\|_2^2}, \tag{2.21}$$

where γ controls the scale of this function. Note that the weight matrix is also the adjacency matrix of the graph G obtained from the samples $\mathbf{Y}$. Since we consider only an undirected graph here, the weight matrix can be made symmetric. The diagonal degree matrix $\mathbf{\Delta}$ is computed such that $\delta_{ii} = \sum_j w_{ij}$. The Laplacian matrix is now obtained as $\mathbf{L} = \mathbf{D} - \mathbf{W}$, and it is a symmetric positive semidefinite matrix. The embedding is now obtained by performing the generalized eigen decomposition such that

$$\mathbf{L}\mathbf{v} = \lambda\mathbf{\Delta}\mathbf{v}. \tag{2.22}$$

Similar to the case of LLE, we obtain the embedding $\mathbf{Q}$ choosing the d bottom eigen vectors and leaving out the last corresponding to an eigen value of 0. It can be shown that this is the optimal solution to the following objective

$$\min_{\mathbf{Q}} \sum_{ij} \|\mathbf{q}_i - \mathbf{q}_j\|_2^2 w_{ij}, \tag{2.23}$$

along with the constraint that $\text{trace}(\mathbf{Q}\mathbf{D}\mathbf{Q}^T) = \mathbf{I}$. The objective mandates that if two high-dimensional data samples $\mathbf{y}_i$ and $\mathbf{y}_j$ are neighbors, then the low-dimensional embeddings $\mathbf{q}_i$ and $\mathbf{q}_j$ must be close to each other. For a novel data sample $\mathbf{z}$ assumed to be lying close to the manifold, the embedding an be computed without re-performing the eigen decomposition. The embedding $\mathbf{q}$ can be obtained using an out-of-sample extension procedure, by optimizing

$$\min_{\mathbf{q}} \sum_{i} \|\mathbf{q} - \mathbf{q}_i\|_2^2 w_i. \tag{2.24}$$

Here w_i is fixed as 1 if $\mathbf{z}$ is in the neighborhood of the training sample $\mathbf{y}_i$, else it is set as 0.

2.2.3 Manifold Projection

When modeling low-dimensional manifolds, one useful approach is to consider the manifold as a union of convex sets, where each convex set is formed using a small subset of examples [78, 79]. This amounts to a piecewise linear approximation of the

manifold using low dimensional simplices, similar to piecewise linear approximation of one dimensional curves using lines. Furthermore, this interpretation can generalize well to cases where we have data clustered in disjoint unions of convex sets rather than a continuous submanifold. A useful application of this generalization is in the case of high contrast natural image patches, which have been shown to lie in clusters and non-linear low dimensional submanifolds [80]. Since the manifolds that we assumed are triangulable, they can be approximated using local linear models in inverse problems.

In several learning problems, it is necessary to project a data sample onto a known manifold, which is a generalization of projecting a sample onto a known subspace. Denoting the test data as $\mathbf{z}$ and the manifold as $\mathcal{M}$, projection can be performed by selecting a small set of samples $\mathbf{Y}_\Omega = \{\mathbf{y}_i\}_{i\in\Omega}$ in the neighborhood of $\mathbf{z}$ and finding the closest point of $\mathbf{z}$ in the simplex spanned by $\mathbf{Y}_\Omega$. Ω is the index set of manifold samples in the neighborhood of $\mathbf{z}$. Usually the neighborhood is chosen using an ℓ_2 distance measure, and this is similar to the approach used by LLE for manifold learning. An illustration of manifold projection for a sample case in is given in Figure 2.7. When the neighborhood Ω is known, manifold projection involves estimating the weights $\boldsymbol{\alpha}_\Omega$,

$$\hat{\boldsymbol{\alpha}}_\Omega = \underset{\boldsymbol{\alpha}_\Omega}{\operatorname{argmin}} \|\mathbf{z} - \mathbf{Y}_\Omega\boldsymbol{\alpha}_\Omega\|_2^2 \text{ subject to } \sum_{i\in\Omega}\alpha_i = 1, \alpha_i \geq 0 \; i \in \Omega, \tag{2.25}$$

from which the projection can be estimated as $\hat{\mathbf{z}} = \mathbf{Y}_\Omega\boldsymbol{\alpha}_\Omega$.

The selection of the neighborhood and projection of the test sample onto the low dimensional simplex can be posed as a weighted sparse coding problem,

$$\begin{aligned}\hat{\boldsymbol{\alpha}} = \quad & \underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \|\mathbf{z} - \mathbf{Y}\boldsymbol{\alpha}\|_2^2 + \lambda\sum_{i=1}^{T}\|\mathbf{z} - \mathbf{y}_i\|_2^2|\alpha_i| \\ & \text{subject to } \sum_{i=1}^{T}\alpha_i = 1, \alpha_i \geq 0 \; \forall i,\end{aligned} \tag{2.26}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^T$ are coefficients for the projection of $\mathbf{z}$ on $\mathbf{Y}$. The constraints on $\boldsymbol{\alpha}$ ensure that the test sample is projected onto the convex hull (simplex) spanned by the

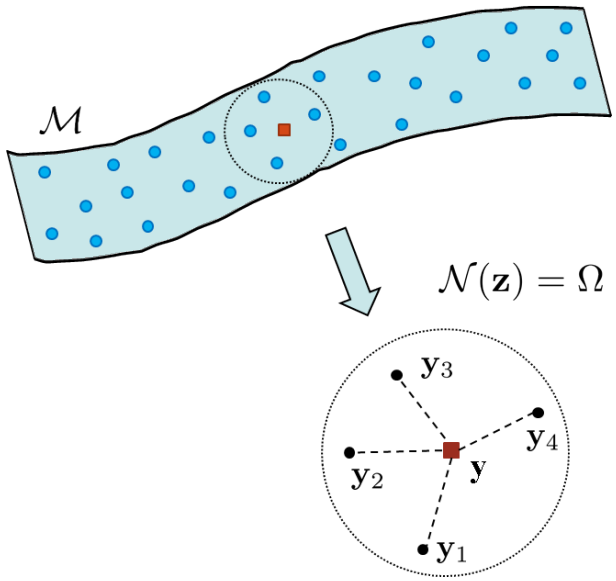


Figure 2.7: Manifold projection of test data using four nearest neighbor samples from $\mathcal{M}$.

chosen manifold samples. Nearest neighbor and weighted sparse coding based models have been used for local dictionaries from data which have been very successful for object classification applications [16].

2.3 Graph-Embedding Methods

In several learning algorithms, the pairwise relationship between the data samples $\mathbf{Y}$ can be encoded using graphs. This can be observed in the case of manifold learning methods in Section 2.2.2. These graphs can be then used to perform dimensionality reduction for the data. The formulation of graph embedding [51] provides a unified framework under which such dimensionality reduction schemes can be analyzed. Furthermore, manifold learning schemes are typically unsupervised and do not take into account the additional label information that may be available for the training samples. Whereas, graph embedding techniques can be applied successfully for dimensionality reduction with unsupervised, supervised or semi-supervised cases. The examples of such graphs are given in Figure 1.2 graphs.

2.3.1 Locality Preserving Projections

An example of an unsupervised graph embedding approach is Locality preserving projections (LPP) [81], which uses a graph similar to that used by Laplacian eigenmaps. However, unlike Laplacian eigenmaps, it computes orthonormal projection directions that form the basis set for the embedding subspace. Therefore, embedding for both training samples and test samples can be obtained by projecting them on to the embedding subspace. Constructing the affinity matrix using (2.20), the orthonormal projection directions $\mathbf{V} \in \mathbb{R}^{M \times d}$, can be computed by optimizing

$$\min_{\text{trace}(\mathbf{V}^T\mathbf{Y}\boldsymbol{\Delta}\mathbf{Y}^T\mathbf{V})=\mathbf{I}} \text{trace}(\mathbf{V}^T\mathbf{Y}\mathbf{L}\mathbf{Y}^T\mathbf{V}). \tag{2.27}$$

The embedding for any data sample $\mathbf{y}$ can be obtained as $\mathbf{V}^T\mathbf{y}$. Similar to the case of Laplacian eigenmaps, it can be shown that LPP computes the projection directions by minimizing the objective

$$\min_{\mathbf{V}} \sum_{i,j=1}^{T} \|\mathbf{V}^T\mathbf{y}_i - \mathbf{V}^T\mathbf{y}_j\|_2^2 w_{ij} \text{ subj. to } \sum_{i=1}^{T} \|\mathbf{V}^T\mathbf{y}_i\|_2^2 \delta_{i,i} = 1. \tag{2.28}$$

This optimization ensures that the embedding preserves the neighborhood structure of the graph.

2.3.2 Linear Discriminant Analysis

The unsupervised embedding algorithms considered so far use a single graph, which can be constructed with just the data and a distance function. We will introduce a discriminative embedding algorithm that will incorporate the label information provided for the training data and construct an embedding that improves the separation between classes.

The Linear discriminant analysis (LDA) algorithm [26] computes a set of d discriminant directions $\mathbf{V}$, so that the class separability of the projected data $\mathbf{V}^T\mathbf{Y}$ is improved. The projection directions are computed using the within-class and

between-class scatter matrices. The within-class scatter matrix is obtained as,

$$\mathbf{S}_w = \sum_{c=1}^{C} \sum_{i=1}^{T} \mathbb{I}(l_i = c) \left(\mathbf{y}_i - \boldsymbol{\mu}_c\right) \left(\mathbf{y}_i - \boldsymbol{\mu}_c\right)^T, \tag{2.29}$$

and it computes the scatter of data samples around their respective class mean vectors denoted by $\boldsymbol{\mu}_c$ for classes $c = \{1, \ldots, C\}$. The label for the sample i is given by l_i. Note that $\mathbb{I}(.)$ is the indicator function which returns the value of 1 if the argument condition is true and zero otherwise. When all data samples are centered to zero mean, the between-class scatter is given by

$$\mathbf{S}_b = \sum_{c=1}^{C} T_c \boldsymbol{\mu}_c \boldsymbol{\mu}_c^T, \tag{2.30}$$

where T_c denotes the total number of samples in the class c. The embedding dimensions are set to $d = C - 1$ and the objective for computing $\mathbf{V}$ is

$$\max_{\mathbf{V}} \frac{\mathrm{Tr}[\mathbf{V}^T \mathbf{S}_b \mathbf{V}]}{\mathrm{Tr}[\mathbf{V}^T \mathbf{S}_w \mathbf{V}]}. \tag{2.31}$$

This trace ratio minimization is usually converted to minimization of the ratio trace $\mathrm{Tr}[(\mathbf{V}^T \mathbf{S}_w \mathbf{V})^{-1} \mathbf{V}^T \mathbf{S}_b \mathbf{V}]$, for which the greedy solution is obtained using the generalized eigen decomposition [82].

LDA directions can be alternatively computed by maximizing the ratio of $\mathrm{Tr}[\mathbf{V}^T \mathbf{S}_b \mathbf{V}]$ and $\mathrm{Tr}[\mathbf{V}^T (\mathbf{S}_w + \mathbf{S}_b) \mathbf{V}]$. This can be formulated as a graph embedding problem with the intra- and inter-class defined as

$$w_{ij} = \begin{cases} 1/T_{l_i} & \text{if } l_i = l_j, \\ 0 & \text{otherwise,} \end{cases} \tag{2.32}$$

$$w'_{ij} = 1/T. \tag{2.33}$$

Using these affinity matrices we define the diagonal degree matrices $\boldsymbol{\Delta}$ and $\boldsymbol{\Delta}'$, whose elements are computed as $\delta_{ii} = \sum_j w_{ij}$ and $\delta'_{ii} = \sum_j w'_{ij}$ respectively. The Laplacian

matrices are obtained as $\mathbf{L} = \mathbf{\Delta} - \mathbf{W}$ and $\mathbf{L}' = \mathbf{\Delta}' - \mathbf{W}'$. Now the projection directions $\mathbf{V}$ can be obtained by performing the generalized eigen decomposition

$$\mathbf{Y}\mathbf{L}'\mathbf{Y}^T = \lambda \mathbf{Y}\mathbf{L}\mathbf{Y}^T \tag{2.34}$$

and choosing the d top eigen vectors.

2.3.3 Local Discriminant Embedding

Although LDA is a discriminative embedding, it incorporates only the class label information and completely ignores the local structure of the data. Local discriminant embedding (LDE) considers both the locality information as well as the class label information in order to create a discriminative embedding [27]. Let $\mathcal{N}_k(i)$ denote the $k-$nearest neighbor set for the data sample i. The affinity matrices in this case are given by

$$w_{ij} = \begin{cases} 1 & \text{if } l_i = l_j \text{ AND } [i \in \mathcal{N}_k(j) \text{ OR } j \in \mathcal{N}_k(i)], \\ 0 & \text{otherwise,} \end{cases} \tag{2.35}$$

$$w'_{ij} = \begin{cases} 1 & \text{if } l_i \neq l_j \text{ AND } [i \in \mathcal{N}_{k'}(j) \text{ OR } j \in \mathcal{N}_{k'}(i)], \\ 0 & \text{otherwise.} \end{cases} \tag{2.36}$$

The Laplacians $\mathbf{L}$ and $\mathbf{L}'$ are computed and projection directions $\mathbf{V}$ are obtained similar to the case of LDA. The embedding for any data sample $\mathbf{z}$ can be obtained as $\mathbf{V}^T\mathbf{z}$. An example of graphs used by LDE is given in Figure 1.2 (b).

2.3.4 Semi-Supervised Discriminant Analysis

In semi-supervised learning, the information available in the labeled training set is augmented by an unlabeled training set. If the unlabeled data is carefully used, it can be used to improve the discrimination between classes. The unlabeled data is denoted as $\mathbf{Y}_u \in \mathbb{R}^{M \times T_u}$, and the labeled data is given as $\mathbf{Y}_\ell \in \mathbb{R}^{M \times T_\ell}$, and hence the total training set is given by $\mathbf{Y} = [\mathbf{Y}_\ell \ \mathbf{Y}_u]$. Semisupervised discriminant analysis

(SDA) computes discriminative embedding directions incorporating both unlabeled and labeled training set. The entries in the affinity matrices are given by [28]

$$w_{ij} = \begin{cases} 1/T_{l_i} + \alpha s_{ij} & \text{if } l_i = l_j, \\ \alpha s_{ij} & \text{otherwise,} \end{cases} \tag{2.37}$$

$$w'_{ij} = \begin{cases} 1/T_\ell & \text{if both } \mathbf{y}_i \text{ and } \mathbf{y}_j \text{ are labeled,} \\ 0 & \text{otherwise,} \end{cases} \tag{2.38}$$

where

$$s_{ij} = \begin{cases} 1 & \text{if } i \in \mathcal{N}_k(j) \text{ OR } j \in \mathcal{N}_k(i), \\ 0 & \text{otherwise.} \end{cases} \tag{2.39}$$

Note that α is the parameter that adjusts the relative importance of labeled and unlabeled data. An example of graphs used by SDA is given in Figure 1.2 (c).

2.4 Ensemble Methods in Supervised Learning

In standard supervised learning problems, the training examples given to the learning system are denoted by $\mathcal{T} = \{(\mathbf{y}_1, l_1), (\mathbf{y}_2, l_2), \ldots, (\mathbf{y}_T, l_T)\}$. The unknown underlying function corresponding to the features is denoted by $l = f(\mathbf{y})$, where $\mathbf{x}$ is usually a vector of features which represent the different attributes of the data. If the l_i values are drawn from a discrete set $\{1, \ldots, C\}$, the problem is referred to as classification, whereas in regression the values are drawn from a real line. We will consider only the case of classification, and here the goal is to learn a classifier such that given a test data it outputs the estimated labels. Since learning the classifier is an attempt to approximate the target function, we will consider he expected error of the learning algorithm with respect to a target function and training set size. The expected error has three components, as described in [83]: (a) *bias*, which measures the how the close the average classifier is to the target function, (b) *variance*, which measures the

variation of the predictions around the mean prediction, and (c) minimum error that can be achieved using a Bayes optimal classifier for the target function.

Ensemble methods attempt to combine the decisions of multiple classifiers in order to classify new examples. As it has been argued in [84], using ensembles may reduce the bias and variance of the learning algorithms. Intuitively, it can be seen that if on an average a classifier performs better than random guessing, and if different classifiers make different errors on each sample, combining the decisions of those classifiers will give a better accuracy compared to each of them. In fact, the necessary and sufficient condition for an ensemble to perform better than each of its members is that the classifiers are diverse and accurate [85]. Diversity means that the errors made by each of the classifiers is uncorrelated and accuracy means that each classifier performs better than random guessing. We will two ensemble based classification systems, bagging [30] and boosting [31] that have been widely used in supervised learning. The L individual classifiers used in our discussions are denoted by $h_1, \ldots, h_L$.

2.4.1 Bootstrap Aggregation

Usually the single training set $\mathcal{T}$ is used to arrive at a single classifier h. Suppose we obtain L learning sets $\{\mathcal{T}_l\}$ each consisting of T independent observations from the underlying process. Assuming that we learn a classifier from each of the sets, the output of the final classifier will be the expected value of the individual outputs. Since we are dealing with classifiers, expectation can be replaced by a weighted voting mechanism, where each classifier gives a weighted vote, its weight determined by the probability of the learning set.

However, usually we are supplied with only a single training set $\mathcal{T}$ and do not access to the underlying distribution. Still we can obtain the learning sets $\{\mathcal{T}_l\}$, each of which is a *bootstrap replicate* of the set $\mathcal{T}$. The boostrap replicates are obtained

by drawing T samples at random, with replacement, from the set $\mathcal{T}$. Each bootstrap replicate contains, on an average 63.2% of samples from the original training set, with several samples repeated in each set. The final classifier h, will be the average

$$h = \frac{1}{L}\sum_{l=1}^{L} h_l. \tag{2.40}$$

For bagging to succeed the base classifier must be *unstable*, i.e., it must be substantially different for each of the bootstrapped training sets. Else, the averaging using (2.40) will not be of much help. It has been shown in [86] empirically that bagging and other randomized methods help in addressing the statistical problem in learning, which occurs when the amount of training data is small compared to the complexity of the hypothesis space. Although it is intuitive that bagging will reduce the variance term in the expected error of the learning algorithm, it has been argued that it may also reduce the bias portion of the error for some datasets [87].

2.4.2 Boosting

Boosting is also an ensemble based classification method that combines several moderately accurate classifiers to produce an accurate prediction rule. The fundamental difference between bagging and boosting is that boosting learns a set of classifiers sequentially, where each classifier focuses on the samples wrongly classified by the previous ones. Although several methods for boosting exist, we will focus only the AdaBoost algorithm [31], because of its simplicity and wide applicability.

To begin with, let us consider only the two-class AdaBoost algorithm, which uses an ensemble of weak classifiers $\{h_l\}$, each of which is the map $h_l : \mathcal{Y} \to \{-1, +1\}$. Here $\mathcal{Y}$ denotes the set of training samples. Each weak hypothesis has an associated error defined as

$$\epsilon_l = \sum_{i=1}^{T} \mathbb{I}(l_i \neq h_l(\mathbf{y}_i)) p_t(\mathbf{y}_i) \tag{2.41}$$

where $p_l(.)$ denotes the probability masses on the training examples at round l. The complete AdaBoost algorithm [31] is:

1. Initialize the probability masses, $p_l(\mathbf{y}_i) = 1/T$, $i = 1, 2, \ldots, T$

2. For $l = 1$ to L:

   a) Fit a classifier h_l to the training data using weights $p_l(\mathbf{y}_i)$.

   b) Compute error ϵ_l using (2.41).

   c) Compute

$$\alpha_l = \log \frac{1 - \epsilon_l}{\epsilon_l}. \tag{2.42}$$

   d) Update the probability masses

$$p_{l+1}(\mathbf{y}_i) \leftarrow p_{l+1}(\mathbf{y}_i) \exp\left(\alpha_l \mathbb{I}\left(l_i \neq h_l(\mathbf{y}_i)\right)\right), \ i = \{1, 2, \ldots, T\} \tag{2.43}$$

   e) Renormalize $p_{l+1}(\mathbf{y}_i)$.

3. Output final class of data $\mathbf{y}$ as

$$\underset{c}{\operatorname{argmax}} \sum_{l=1}^{L} \alpha_l \mathbb{I}(h_l(\mathbf{y}_i) = l_i). \tag{2.44}$$

Using the weak hypothesis h_l in each iteration, AdaBoost chooses the parameter α_l, which measures the importance assigned to h_l. Note that it is required to an accurate classifier, i.e., $\epsilon_l < 1/2$, in each round to have a valid $\alpha_l > 0$. In other words, $\alpha_l < 0$ will signal that the particular weak classifier may reduce the performance of the ensemble. The update of the probability masses is performed such that misclassified samples get higher weights in the next round. The final hypothesis is a weighted vote of the L weak hypothesis.

It can be shown that statistically the two-class AdaBoost computes the final hypothesis using forward stagewise additive modeling and an exponential loss function [88]. In order to perform boosting in multiclass classification, an algorithm based on multiclass exponential loss function has been proposed in [89].

Boosting tries to reduce the classification error aggressively and hence may result in better performance compared to bagging, with weak classifiers. However, in the presence of classification noise, i.e., when there are mislabeled training samples, bagging tends to perform better than boosting [90], when there are mislabeled training samples, i.e., in the presence of classification noise. This is because, boosting may provide very high weights for mislabeled training samples in several rounds, leading to a poor generalization with test data. Whereas, the diversity of bagging classifiers is improved in the presence of classification noise. Other ensemble based algorithms include Bayesian voting algorithms, algorithms that manipulate the output targets and those that randomize the internal decisions, such as randomized decision trees [86, 90].

2.5 Kernel Models

For certain classes of signals, the low-dimensional structure is revealed after applying non-linear transformations. The underlying generative processes of these signals have very low degrees of freedom that cannot be modeled using unions of subspaces and local linearity may be a restrictive assumption as well. We refer to such a space obtained after the non-linear transformation as the *feature space*. Let the data $\mathbf{Y}$ in the input space $\mathcal{Y}$, be transformed to the feature space $\mathcal{F}$, using the non-linear transformation $\phi : \mathcal{Y} \mapsto \mathcal{F}$. The feature space is an inner product space with a reproducing kernel $K(.,.)$defined (reproducing kernel Hilbert space). Since the dimension of $\mathcal{F}$ can be very high and sometimes infinite depending on $\phi(.)$, given two signals $\mathbf{y}_i$ and $\mathbf{y}_j$, the operations in $\mathcal{F}$ can be efficiently performed exclusively using the inner product $\langle \phi(\mathbf{y}_i), \phi(\mathbf{y}_j) \rangle_{\mathcal{F}} = K(\mathbf{y}_i, \mathbf{y}_j)$ [49]. Another important property of the feature space $\mathcal{F}$ is that the transformed data $\phi(\mathbf{Y})$ span the space and hence any element of the space can be expressed as a linear combination of $\phi(\mathbf{Y})$. Commonly used kernels can be broadly divided into three categories, projective, radial and user-defined. Pro-

jective kernels are of the form $K(\mathbf{y}_i, \mathbf{y}_j) = f(\langle \mathbf{y}_i, \mathbf{y}_j \rangle_{\mathcal{Y}})$, and some examples of these are polynomial, exponential and sigmoid kernels. Radial kernels depend on the ℓ_2 distance between the data and hence can be written as $K(\mathbf{y}_i, \mathbf{y}_j) = f(\|\mathbf{y}_i - \mathbf{y}_j\|^2_{\mathcal{Y}})$. User-defined kernels can either have a mathematical form, or it can just be a matrix of feature space similarities between the data samples. The only condition for the user-defined kernel to be valid is that the kernel matrix should be symmetric positive-semidefinite as specified by Mercer's theorem [91].

Principal component analysis (PCA) can be performed in the feature space using the kernel trick [48]. Discriminant subspaces of the feature space can also be computed using generalized discriminant analysis (GDA) [92] such that the projected data in the feature space may be well-separated as required by certain inference tasks such as classification and clustering. The inverse of the map $\phi : \mathcal{Y} \mapsto \mathcal{F}$ is not defined for all vectors in $\mathcal{F}$. Hence, the approximate preimage of an arbitrary vector in $\mathcal{F}$ can be obtained using techniques based on multi-dimensional scaling (MDS) [93], conformal mapping [94] or local isomorphism [95].

Chapter 3

GRAPH-EMBEDDED SPARSE REPRESENTATIONS

3.1 Introduction

As discussed in the previous chapters, several supervised, semi-supervised, and unsupervised machine learning schemes can be unified under the general framework of graph embedding. Incorporating graph embedding principles into sparse representation based learning schemes can provide an improved performance in several learning tasks. In this chapter, we propose dictionary learning procedures for computing sparse codes that obey graph embedding constraints. Sparse codes for unlabeled graphs can be obtained by minimizing a convex objective, and this can be efficiently implemented by modifying the existing LARS algorithm. However, obtaining discriminative sparse codes is a highly non-convex problem and we propose two methods to obtain supervised graph-embedded sparse codes (GE-SC). The first procedure integrates a modified version of the sequential quadratic programming (SQP) procedure [96] with the feature sign search (FSS) method [97]. The second approach uses the concave-convex procedure (CCCP) to linearize the concave part of the objective function, and performs an iterative optimization to obtain the GE-SCs. Experiments demonstrating that the proposed graph embedded sparse codes perform better than several baseline methods in unsupervised clustering, supervised and semi-supervised classification, are also presented. Some of the methods and results in this chapter have been published in [55].

Sparse coding aims to obtain a parsimonious representation for the data using the basis functions in a given dictionary. However, the coding process does not explicitly consider the underlying graph structure of the data. Assuming that the data $\mathbf{Y}$ can be sparsely represented using a dictionary, $\mathbf{\Psi} \in \mathbb{R}^{M \times K}$, the authors in [98, 99] proposed an algorithm to obtain sparse codes, $\mathbf{X} \in \mathbb{R}^{K \times T}$, that preserve

the neighborhood structure of the unsupervised graph G by solving

$$\min_{\mathbf{\Psi},\mathbf{X}} \|\mathbf{Y} - \mathbf{\Psi}\mathbf{X}\|_F^2 + \gamma \sum_i \|\mathbf{x}_i\|_1 + \beta \text{trace}(\mathbf{X}\mathbf{L}\mathbf{X}^T) \quad \text{subj. to } \forall j, \|\boldsymbol{\psi}_j\|_2 \leq 1, \tag{3.1}$$

where $\mathbf{L}$ is the Laplacian of the graph G. Here $\mathbf{x}_i$ is the sparse code for the sample $\mathbf{x}_i$ and $\boldsymbol{\psi}_j$ is the j^{th} column of $\mathbf{\Psi}$. We will refer to this as the unsupervised GE-SC.

Graph embedding can also be applied to supervised and semi-supervised learning problems. Local discriminant embedding (LDE) [27] is one such supervised embedding scheme that incorporates intra- and inter-class relationships by respectively defining two graphs G and G'. Denoting the graph Laplacian of G' by $\mathbf{L}'$, the optimal directions for linear embedding in LDE can be obtained by solving

$$\min_{\text{trace}(\mathbf{V}^T\mathbf{Y}\mathbf{L}'\mathbf{Y}^T\mathbf{V})=\mathbf{I}} \text{trace}(\mathbf{V}^T\mathbf{Y}\mathbf{L}\mathbf{Y}^T\mathbf{V}). \tag{3.2}$$

Furthermore, when only a subset of the training data is labeled, a semi-supervised graph embedding approach, referred to as semi-supervised discriminant analysis (SDA) has been developed [28].

Approaches such as local sparse coding incorporate the graph or manifold structure of the data indirectly by considering the neighbor relation between the training data and the dictionary [16], and this has been useful in classification and image retrieval [100]. The unsupervised GE-SC method, that uses unsupervised graphs, has been used in classification and clustering of images [99]. The authors in [19] proposed to use an ℓ_1-graph, that describes the neighborhood relation between the training data using sparse codes for clustering, subspace learning, and semi-supervised learning.

In this chapter, graph embedded sparse coding and dictionary learning is performed using an alternating optimization procedure. The GE-SC are obtained using a modified version of SQP in the FSS algorithm. Dictionary learning is performed using the Lagrangian dual method by fixing the sparse codes. We use the proposed

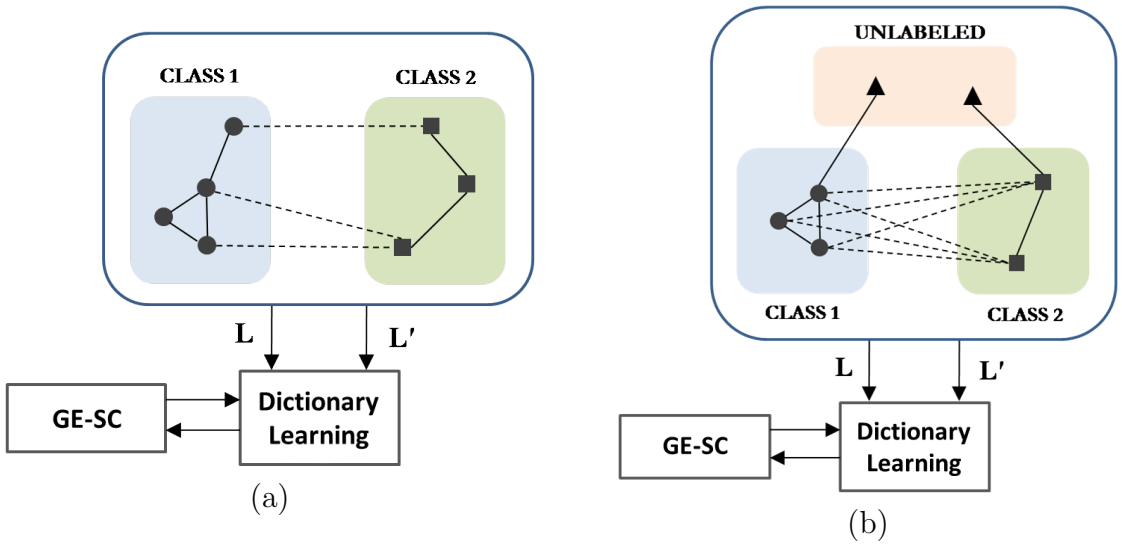


Figure 3.1: Proposed graph-embedded sparse coding approaches: (a) Supervised graph, (b) Semi-supervised graph. Note that solid and dotted lines denote the edges in the graphs G and G' respectively. In (b), G' is fully connected within a class also (overlaps with the solid lines).

algorithm to obtain GE-SC for supervised and semi-supervised graphs, as illustrated in Figure 3.1. Results obtained with the AR face database demonstrate that the sparse codes computed using the proposed dictionaries perform better in comparison to other baseline approaches in supervised and semi-supervised classification.

3.2 Supervised Graph-Embedded Sparse Coding

The proposed optimization problem for dictionary learning and sparse coding with graph embedding constraints can be expressed as

$$\begin{aligned} &\min_{\mathbf{\Psi},\mathbf{X}} \|\mathbf{Y}-\mathbf{\Psi}\mathbf{X}\|_F^2 + \gamma \sum_i \|\mathbf{x}_i\|_1 + \beta \text{trace}(\mathbf{X}\mathbf{L}\mathbf{X}^T) \\ &\text{subj. to } \text{trace}(\mathbf{X}\mathbf{L}'\mathbf{X}^T) = c, \|\boldsymbol{\psi}_j\|_2 \leq 1, \forall j \end{aligned} \tag{3.3}$$

where γ and β are positive constants. The objective function ensures that the codes obtained are sparse, reconstruct the data well, and minimize the intra-class neighbor distance. The equality constraint ensures that the inter-class neighbor distances are fixed, and this along with the objective results in an improved discrimination

between the sparse codes, when compared to the original data. The GE-SCs and the dictionaries can also be obtained by posing the optimization as

$$\min_{\mathbf{\Psi},\mathbf{X}} \|\mathbf{Y}-\mathbf{\Psi}\mathbf{X}\|_F^2+\gamma\sum_i \|\mathbf{x}_i\|_1+\beta\left(\text{trace}\left(\mathbf{X}\mathbf{L}\mathbf{X}^T\right)-\lambda\text{trace}\left(\mathbf{X}\mathbf{L}'\mathbf{X}^T\right)\right)$$
$$\text{subj. to } \|\boldsymbol{\psi}_j\|_2 \leq 1, \forall j. \tag{3.4}$$

In this case, the parameter λ controls the inter-class and intra-class penalties. Codes for the novel test samples can be computed using unconstrained sparse coding with the learned dictionary $\mathbf{\Psi}$. The proposed GE-SC formulations incorporate the unsupervised GE-SC given by (3.1) as a special case.

3.2.1 Supervised Graph

When labeled training data is available, the problem in (3.3) can be solved to obtain dictionaries that produce highly discriminative sparse codes, by using the graph Laplacians as defined in LDE. Let the label for the training data $\mathbf{y}_i$ be l_i and let $\mathcal{N}_k(i)$ denote the set of k nearest samples of $\mathbf{y}_i$. For supervised GE-SC, the entries in the affinity matrix $\mathbf{W}$ and $\mathbf{W}'$ are defined similar to the case of LDE as [27]

$$w_{ij} = \begin{cases} 1 & \text{if } l_i = l_j \text{ AND } [i \in \mathcal{N}_k(j) \text{ OR } j \in \mathcal{N}_k(i)], \\ 0 & \text{otherwise,} \end{cases} \tag{3.5}$$

$$w'_{ij} = \begin{cases} 1 & \text{if } l_i \neq l_j \text{ AND } [i \in \mathcal{N}_{k'}(j) \text{ OR } j \in \mathcal{N}_{k'}(i)], \\ 0 & \text{otherwise.} \end{cases} \tag{3.6}$$

The graph Laplacians $\mathbf{L}$ and $\mathbf{L}'$, used in (3.3), are then constructed using $\mathbf{W}$ and $\mathbf{W}'$.

3.2.2 Semi-Supervised Graph

When the training set consists of both unlabeled data $\mathbf{Y}_u \in \mathbb{R}^{M\times T_u}$, and labeled data $\mathbf{Y}_\ell \in \mathbb{R}^{M\times T_\ell}$, such that $\mathbf{Y} = [\mathbf{Y}_\ell\ \mathbf{Y}_u]$, the dictionary $\mathbf{\Psi}$ and sparse codes can be obtained using the graph Laplacians defined by SDA. For the semi-supervised GE-SC approach, the labeled training data is augmented using unlabeled training samples in

the neighborhood. Let us denote T_{l_i} as the number of samples in class specified by l_i and α as the parameter that adjusts the relative importance of labeled and unlabeled data. The entries in the affinity matrices are defined similar to the case of SDA [28]

$$w_{ij} = \begin{cases} 1/T_{l_i} + \alpha s_{ij} & \text{if } l_i = l_j, \\ \alpha s_{ij} & \text{otherwise,} \end{cases} \tag{3.7}$$

$$w'_{ij} = \begin{cases} 1/T_{\ell} & \text{if both } \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ are labeled,} \\ 0 & \text{otherwise,} \end{cases} \tag{3.8}$$

where

$$s_{ij} = \begin{cases} 1 & \text{if } i \in \mathcal{N}_k(j) \text{ OR } j \in \mathcal{N}_k(i), \\ 0 & \text{otherwise.} \end{cases} \tag{3.9}$$

3.3 Proposed Algorithms

We will first consider the problem of unsupervised GE-SC given in (3.1). It is solved as an alternating minimization procedure that optimizes the dictionary while fixing the sparse codes and vice-versa. When sparse codes are fixed, dictionary learning is a convex problem and hence can be solved efficiently using various approaches. We adopt the Lagrangian dual method proposed in [97] for learning the dictionary. Given the dictionary, unsupervised GE-SC can be obtained using the LARS algorithm [101]. Although computing sparse codes using (3.1) is a convex optimization procedure, it is computationally expensive to obtain the sparse code matrix $\mathbf{X}$ as a whole. Therefore, we obtain the sparse codes of each data sample sequentially. For the data sample i, this can be expressed as

$$\min_{\mathbf{x}_i} \|\mathbf{y}_i - \mathbf{\Psi}\mathbf{x}_i\|_2^2 + \gamma\|\mathbf{x}_i\|_1 + \beta(l_{ii}\|\mathbf{x}_i\|_2^2 + 2\mathbf{x}_i^T\mathbf{X}_{\setminus i}\mathbf{l}_{\setminus i}), \tag{3.10}$$

where l_{ii} is the i^{th} diagonal element of $\mathbf{L}$, $\mathbf{X}_{\setminus i}$ is the coefficient matrix with its i^{th} column removed and $\mathbf{l}_{\setminus i}$, is the i^{th} columns of $\mathbf{L}$ with its i^{th} elements removed. The

above optimization can be equivalently expressed as

$$\min_{\mathbf{x}_i} \mathbf{x}_i^T(\mathbf{\Psi}^T\mathbf{\Psi} + \beta l_{ii}\mathbf{I})\mathbf{x}_i - 2\mathbf{x}_i^T(\mathbf{\Psi}^T\mathbf{y}_i - \mathbf{X}_{\setminus i}\mathbf{l}_{\setminus i}) + \gamma\|\mathbf{x}_i\|_1, \tag{3.11}$$

where $\mathbf{I}$ is the identity matrix of appropriate dimensions. Clearly, (3.11) contains only quadratic and linear terms with a sparsity penalty, and hence it can be efficiently implemented using the LARS algorithm.

For supervised GE-SC given in (3.3), when the dictionary is fixed, sparse codes are computed sequentially for each data sample, and for any sample $\mathbf{y}_i$, we can simplify (3.3) to obtain

$$\begin{aligned} &\min_{\mathbf{x}_i} \|\mathbf{y}_i - \mathbf{\Psi}\mathbf{x}_i\|_2^2 + \gamma\|\mathbf{x}_i\|_1 + \beta(l_{ii}\|\mathbf{x}_i\|_2^2 + 2\mathbf{x}_i^T\mathbf{X}_{\setminus i}\mathbf{l}_{\setminus i}) \\ &\text{subj. to } l'_{ii}\|\mathbf{x}_i\|_2^2 + 2\mathbf{x}_i^T\mathbf{X}_{\setminus i}\mathbf{l}'_{\setminus i} - \delta_t = 0, \end{aligned} \tag{3.12}$$

where l_{ii} and l'_{ii} are the i^{th} diagonal elements of $\mathbf{L}$ and $\mathbf{L}'$, $\mathbf{X}_{\setminus i}$ is the coefficient matrix with its i^{th} column removed and $\mathbf{l}_{\setminus i}$, $\mathbf{l}'_{\setminus i}$ are the i^{th} columns of $\mathbf{L}$, $\mathbf{L}'$ with their i^{th} elements removed. The objective is convex but non-differentiable when the coefficients are zero, and furthermore, we also have a non-convex (quadratic equality) constraint. SQP is an approach that can be used to solve nonlinearly constrained optimization problems [96], by posing it as a sequence of quadratic subproblems. However it requires that the objective be differentiable everywhere, which is not the case for the problem defined in (3.12). Therefore, in order to solve for the sparse codes, we consider a modified version of the SQP method considering only the non-zero coefficients (active set) at every step. The active set can be modified using the feature sign search (FSS) algorithm proposed in [97]. We provide an optimization strategy that combines the modified SQP and the FSS methods.

We also consider obtaining supervised GE-SCs using (3.4). In this case, the sparse codes for individual samples can be obtained by solving

$$\min_{\mathbf{x}_i} \|\mathbf{y}_i - \mathbf{\Psi}\mathbf{x}_i\|_2^2 + \gamma\|\mathbf{x}_i\|_1 + \beta\Big(l_{ii}\|\mathbf{x}_i\|_2^2 + 2\mathbf{x}_i^T\mathbf{X}_{\setminus i}\mathbf{l}_{\setminus i} - \lambda\left(l'_{ii}\|\mathbf{x}_i\|_2^2 + 2\mathbf{x}_i^T\mathbf{X}_{\setminus i}\mathbf{l}'_{\setminus i}\right)\Big). \tag{3.13}$$

Note that the objective is non-convex and more specifically, it is a sum of convex and concave terms. This can be solved efficiently using the concave-convex procedure (CCCP) [102], which iteratively computes the sparse codes by linearizing the concave term.

3.3.1 Modified SQP Method

In order to compute the GE-SCs in (3.12) using the SQP method, we will denote $\mathbf{X}_{\setminus i}\mathbf{l}_{\setminus i}$ as $\mathbf{d}_i$, $\mathbf{X}_{\setminus i}\mathbf{l}'_{\setminus i}$ as $\mathbf{c}_i$ and the sign vector for $\mathbf{x}_i$ as $\boldsymbol{\theta}_i$. Each entry of $\boldsymbol{\theta}_i$ is obtained from the set $\{-1, 0, 1\}$ depending on the sign of the corresponding entry in $\mathbf{x}_i$. We will use $\hat{\mathbf{x}}_i$ to specify the current non-zero or active set of coefficients, and use $\hat{\mathbf{\Psi}}$ for the reduced dictionary with columns corresponding to the active set. Similar notation is used for other vectors and matrices also. Expressing

$$f(\hat{\mathbf{x}}_i) = \|\mathbf{y} - \hat{\mathbf{\Psi}}\hat{\mathbf{x}}_i\|_2^2 + \beta(\l_{ii}\|\hat{\mathbf{x}}_i\|_2^2 + 2\hat{\mathbf{x}}_i^T\hat{\mathbf{d}}_i), \tag{3.14}$$

$$h(\hat{\mathbf{x}}_i) = \l'_{ii}\|\hat{\mathbf{x}}_i\|_2^2 + 2\hat{\mathbf{x}}_i^T\hat{\mathbf{c}}_i - \delta_t, \tag{3.15}$$

$$s(\hat{\mathbf{x}}_i) = \gamma\hat{\boldsymbol{\theta}}_i^T\hat{\mathbf{x}}_i, \tag{3.16}$$

and considering only the active coefficients, the optimization problem in (3.12) can be written as,

$$\min_{\hat{\mathbf{x}}_i} f(\hat{\mathbf{x}}_i) + s(\hat{\mathbf{x}}_i) \text{ subj. to } h(\hat{\mathbf{x}}_i) = 0. \tag{3.17}$$

Since SQP is a non-convex optimization procedure, it is necessary to define a merit function to ensure that it converges from remote starting points. The merit function measures both the reduction in objective, as well as the degree to which the constraint

is met and hence it plays the role of objective function in unconstrained minimization. We will use the ℓ_1 merit function

$$m(\hat{\mathbf{x}}_i) = f(\hat{\mathbf{x}}_i) + \sigma|h(\hat{\mathbf{x}}_i)|, \tag{3.18}$$

where σ controls the relative importance of $f(\hat{\mathbf{x}}_i)$ and $h(\hat{\mathbf{x}}_i)$, and its gradient is given by

$$\nabla m(\hat{\mathbf{x}}_i) = \nabla f(\hat{\mathbf{x}}_i) + \sigma \; sign(h(\hat{\mathbf{x}}_i))\nabla h(\hat{\mathbf{x}}_i). \tag{3.19}$$

In the subsequent description, for simplicity, we will drop the subscript i that corresponds to the index of the data sample.

The SQP method sequentially solves the Karush-Kuhn-Tucker (KKT) system of the problem sequentially, using Newton-Lagrange method. Therefore, we first define the Lagrangian function for (3.17) as,

$$\mathcal{L}(\hat{\mathbf{x}}, \lambda) = g(\hat{\mathbf{x}}) - \lambda h(\hat{\mathbf{x}}), \tag{3.20}$$

where $g(\hat{\mathbf{x}}) = f(\hat{\mathbf{x}}) + s(\hat{\mathbf{x}})$ and λ is the dual variable. The KKT conditions can be expressed as

$$\mathbf{G}(\hat{\mathbf{x}}, \lambda) = \begin{bmatrix} \nabla g(\hat{\mathbf{x}}) - \lambda \nabla h(\hat{\mathbf{x}}) \\ h(\hat{\mathbf{x}}) \end{bmatrix}. \tag{3.21}$$

Assuming that we are at the k^{th} iteration, the solution at the next iteration is denoted as

$$\begin{bmatrix} \hat{\mathbf{x}}^{(k+1)} \\ \lambda^{(k+1)} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{x}}^{(k)} \\ \lambda^{(k)} \end{bmatrix} + \begin{bmatrix} \Delta\hat{\mathbf{x}}^{(k)} \\ \Delta\lambda^{(k)} \end{bmatrix}, \tag{3.22}$$

Table 3.1: The Feature Sign Search algorithm that incorporates the modified SQP method for computing GE-SC.

Step 1: Initialize $\mathbf{x}^{(0)}$, active set $\mathcal{A}$, $\theta(i) = sign(x(i))$, $k = 0$.

Step 2: Initialize $\lambda^{(0)} = \frac{\nabla h(\hat{\mathbf{x}}^{(0)})^T \nabla g(\hat{\mathbf{x}}^{(0)})}{\|\nabla h(\hat{\mathbf{x}}^{(0)})\|_2^2}$.

Step 3: Select $i = \underset{i}{\operatorname{argmax}} \left| \frac{\partial m(\mathbf{x})}{\partial x(i)} \right|$ from non-active coefficients. Add i to $\mathcal{A}$ if local improvement in merit function is obtained.
- if $\frac{\partial m(\mathbf{x})}{\partial x(i)} > \gamma$, set $\theta(i) = -1, \mathcal{A} = \mathcal{A} \cup \{i\}$,
- if $\frac{\partial m(\mathbf{x})}{\partial x(i)} < -\gamma$, set $\theta(i) = 1, \mathcal{A} = \mathcal{A} \cup \{i\}$.

Step 4: Feature Sign Step
- Using modified SQP, compute $\Delta\hat{\mathbf{x}}^{(k)}$ and $\Delta\lambda^{(k)}$.
- Perform line search on $\hat{\mathbf{x}}^{(k)} + \tau\Delta\hat{\mathbf{x}}^{(k)}$ and $\lambda^{(k)} + \tau\Delta\lambda^{(k)}$ where $0 \leq \tau \leq 1$.
- Check objective at all values of τ where at least one coefficient changes sign.
- Find $\|G(\hat{\mathbf{x}}, \lambda)\|_2$ at all such points and update $\hat{\mathbf{x}}^{(k+1)}$, $\lambda^{(k+1)}$ to the point where $\|G(\hat{\mathbf{x}}, \lambda)\|_2$ is the smallest.
- Update active set by removing zero coefficients and set $\boldsymbol{\theta} = sign\left(\mathbf{x}^{(k+1)}\right)$.

Step 5: Check Optimality
(a) For non-zero coefficients, check if $G(\hat{\mathbf{x}}, \lambda) = 0$. If true, check condition (b) else go to Step 4.
(b) In non-active set, check if $\left|\frac{\partial m(\mathbf{x})}{\partial x(i)}\right| \leq \gamma, \forall x(i) = 0$. If true, return $\mathbf{x}$ as the solution else go to Step 3.

the increment $(\Delta\hat{\mathbf{x}}^{(k)}, \Delta\lambda^{(k)})$ can be obtained by solving the KKT system (3.21) as

$$\begin{bmatrix} \nabla^2\mathcal{L}(\hat{\mathbf{x}}^{(k)}, \lambda^{(k)}) & -\nabla h(\hat{\mathbf{x}}^{(k)}) \\ \nabla h(\hat{\mathbf{x}}^{(k)})^T & 0 \end{bmatrix} \begin{bmatrix} \Delta\hat{\mathbf{x}}^{(k)} \\ \Delta\lambda^{(k)} \end{bmatrix}$$

$$= - \begin{bmatrix} \nabla g(\hat{\mathbf{x}}^{(k)}) - \lambda^{(k)}\nabla h(\hat{\mathbf{x}}^{(k)}) \\ h(\hat{\mathbf{x}}^{(k)}) \end{bmatrix}, \tag{3.23}$$

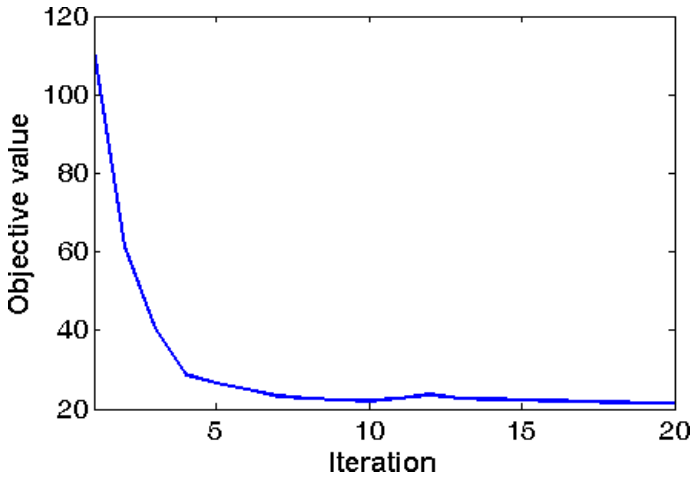


Figure 3.2: Convergence of the GE-SC objective function over 30 iterations using the modified SQP procedure.

where

$$\nabla g(\hat{\mathbf{x}}^{(k)}) = 2(\hat{\mathbf{\Psi}}^T\hat{\mathbf{\Psi}} + \beta l\mathbf{I})\hat{\mathbf{x}}^{(k)} + 2(\hat{\mathbf{d}} - \hat{\mathbf{\Psi}}^T\mathbf{y} + (\gamma/2)\hat{\boldsymbol{\theta}}), \tag{3.24}$$

$$\nabla h(\hat{\mathbf{x}}^{(k)}) = 2(l'\hat{\mathbf{x}}^{(k)} + \hat{\mathbf{c}}), \tag{3.25}$$

and

$$\nabla^2\mathcal{L}(\hat{\mathbf{x}}^{(k)}, \hat{\boldsymbol{\lambda}}^{(k)}) = 2(\hat{\mathbf{\Psi}}^T\hat{\mathbf{\Psi}} + (\beta l - \lambda l')\mathbf{I}). \tag{3.26}$$

The Feature Sign Search Algorithm: The active coefficient set is updated and optimality of (3.12) is ensured using the FSS algorithm described in Table 3.1. The coefficients are initialized using unconstrained sparse coding, and they are updated sequentially for the data samples. Here $x(i)$ and $\theta(i)$ denote the i^{th} elements of the vectors $\mathbf{x}$ and $\boldsymbol{\theta}$ respectively. The active set $\mathcal{A}$ contains the indices of the non-zero coefficients. The empirical convergence of the objective function in a supervised learning scenario is given in Figure 3.2.

3.3.2 The Concave-Convex Procedure

The CCCP is a procedure that is guaranteed to monotonically decrease the objective value of energy functions [102]. Any function with a Hessian that has eigen values

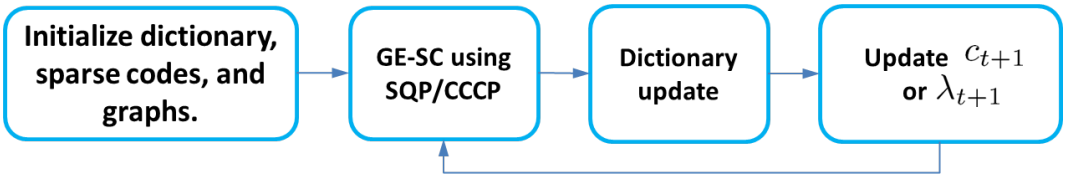


Figure 3.3: The general algorithm for optimizing the dictionary and obtaining supervised GE-SC.

bounded above zero can be expressed as a sum of a convex part and a concave part. Therefore, CCCP is a general technique that can be applied to several optimization problems. Given the energy function $E(\mathbf{x})$, which can be expressed as the sum of concave and convex parts, $E_{vex}(\mathbf{x}) + E_{cave}(\mathbf{x})$, the energy function to be minimized in step $k+1$ is

$$E_{k+1}(\mathbf{x}^{(k+1)}) = E_{vex}(\mathbf{x}^{(k+1)}) + \mathbf{x}^{(k+1)^T} \nabla E_{cave}(\mathbf{x}^{(k)}). \tag{3.27}$$

Essentially, this involves linearizing the concave part of the objective around its previous solution and finding the minimum of the modified objective. In the case of the GE-SC objective function given in (3.13), clearly the concave part is given by $(-\beta\lambda l'_{ii}\|\mathbf{x}_i\|_2^2)$. Hence, computing the GE-SC is performed by linearizing the concave part and the objective for $(k+1)^{\text{th}}$ iteration is given as

$$\begin{aligned}\min_{\mathbf{x}_i} \|\mathbf{y}_i - \mathbf{\Psi}\mathbf{x}_i\|_2^2 + \gamma\|\mathbf{x}_i\|_1 + \beta\Big(& l_{ii}\|\mathbf{x}_i\|_2^2 + 2\mathbf{x}_i^T\mathbf{X}_{\setminus i}\mathbf{l}_{\setminus i} \\ & - \lambda\left(l'_{ii}\mathbf{x}_i^{(k)^T}\mathbf{x}_i + 2\mathbf{x}_i^T\mathbf{X}_{\setminus i}\mathbf{l}'_{\setminus i}\right)\Big),\end{aligned} \tag{3.28}$$

where $\mathbf{x}_i^{(k)}$ results in a minimum value for the objective in the previous iteration. The solution can be obtained using LARS by identifying the linear, quadratic and ℓ_1 terms in 3.28 similar to the case of (3.11). Repeating this for a few iterations is reduces the objective sufficiently and results in GE-SCs. The dictionary can be updated using the Lagrangian dual method.

3.4 Dictionary Learning and GE-SC

The overall dictionary learning and GE-SC computation procedure is summarized in Figure 3.3 for discriminative graphs. For unsupervised GE-SC, the procedure is similar to the alternating sparse coding and dictionary training optimization. For the supervised/semi-supervised case, graphs are obtained using the appropriate procedure as described in Sections 3.2.1 and 3.2.2. The dictionaries are initialized as normalized K-means cluster centers of the data. The GE-SC are initialized as plain sparse codes.

3.4.1 Iterative Trace Ratio Procedure

Considering the conventional problem of supervised graph embedding, the discriminative directions for the embedding can be obtained using (3.2). This can also be expressed as trace ratio maximization

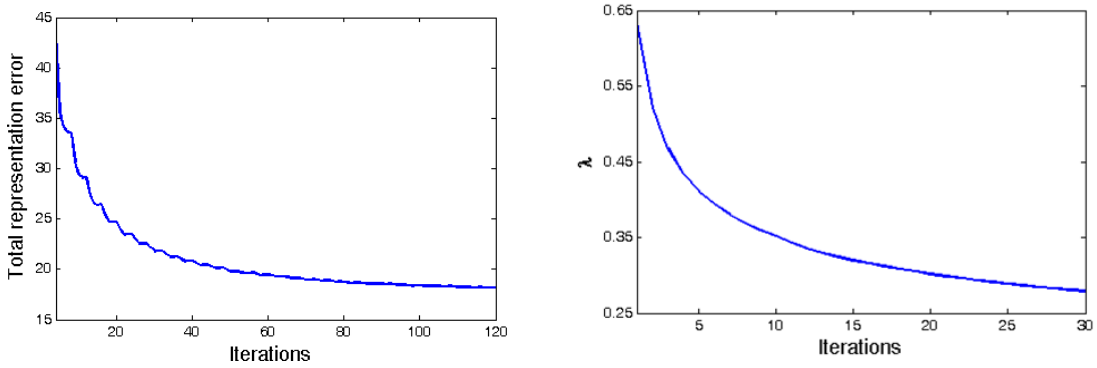
$$\max_{\mathbf{V}^T\mathbf{V}=\mathbf{I}} \frac{\text{trace}(\mathbf{V}^T\mathbf{Y}\mathbf{L}'\mathbf{Y}^T\mathbf{V})}{\text{trace}(\mathbf{V}^T\mathbf{Y}\mathbf{L}\mathbf{Y}^T\mathbf{V})}. \tag{3.29}$$

The greedy solution to this problem is obtained by computing the embedding directions $\mathbf{v}_i$ sequentially such that $\frac{\mathbf{v}_i^T\mathbf{Y}\mathbf{L}'\mathbf{Y}^T\mathbf{v}_i}{\mathbf{v}_i^T\mathbf{Y}\mathbf{L}\mathbf{Y}^T\mathbf{v}_i}$ is maximized, and the current direction is orthogonal to the previous directions. This greedy optimization scheme leads to the ratio-trace problem

$$\max_{\mathbf{V}^T\mathbf{V}=\mathbf{I}} \text{trace}[(\mathbf{V}^T\mathbf{Y}\mathbf{L}\mathbf{Y}^T\mathbf{V})^{-1}(\mathbf{V}^T\mathbf{Y}\mathbf{L}'\mathbf{Y}^T\mathbf{V})] \tag{3.30}$$

and clearly, this can be solved as a generalized eigenvalue problem. However, as it can be observed, the convenient greedy solution will result only in a suboptimal trace ratio (3.29). Various methods to obtain the global optima for (3.29) have been described in [82, 103, 104]. The iterative algorithm proposed in [104] computes the global optimum by posing the trace difference maximization

$$\mathbf{W}_t = \underset{\mathbf{V}^T\mathbf{V}=\mathbf{I}}{\text{argmax}}\, \text{trace}(\mathbf{V}^T\mathbf{Y}\mathbf{L}' - \lambda_t\mathbf{L}\mathbf{Y}^T\mathbf{V}), \tag{3.31}$$



(a) (b)

Figure 3.4: (a) Convergence of representation error when learning a dictionary for GE-SC using CCCP, (b) Convergence of the inverse trace ratio (λ) parameter, that clearly shows the increasing discrimination across classes as time proceeds.

where λ_{t+1} for next iteration is updated as the trace ratio

$$\lambda_{t+1} = \frac{\text{trace}(\mathbf{V}_t^T \mathbf{Y} \mathbf{L}' \mathbf{Y}^T \mathbf{V}_t)}{\text{trace}(\mathbf{V}_t^T \mathbf{Y} \mathbf{L} \mathbf{Y}^T \mathbf{V}_t)}. \tag{3.32}$$

When $\mathbf{V}$ is an orthonormal system, it has been proved in [104] that this iterative trace ratio (ITR) scheme converges to the global optimum of the trace ratio maximization (3.29).

3.4.2 Incorporating ITR in GE-SC

In our proposed GE-SC approaches, (3.3) and (3.4), the goal is to improve the discrimination across classes similar to the trace ratio approach. The difference, however, is that instead of computing low-dimensional projection directions, we compute sparse codes $\mathbf{X}$ that can directly provide an improved discrimination. For a given dictionary $\mathbf{\Psi}$, the iterative trace ratio problem can be adapted to compute the sparse codes that result in improved discrimination. Note that, in our case, the proof of global optimality is not straightforward, since the dictionary $\mathbf{\Psi}$ is not usually orthonormal. Nevertheless, we formulate the iterative trace ratio schemes for both (3.3) and (3.4). The parameters c and λ in (3.3) and (3.4) are replaced by c_t and λ_t where t denotes the iteration number. The GE-SCs are computed fixing the dictionary, and the

updated parameters are

$$c_{t+1} = \text{trace}(\mathbf{X}_t \mathbf{L}' \mathbf{X}_t^T), \tag{3.33}$$

and

$$\lambda_{t+1} = \frac{\text{trace}(\mathbf{X}_t \mathbf{L} \mathbf{X}_t^T)}{\text{trace}(\mathbf{X}_t \mathbf{L}' \mathbf{X}_t^T)} \tag{3.34}$$

respectively. In order to simplify the learning procedure, as shown in Figure 3.3, the algorithm updates the dictionary every time before updating the parameters. This is similar to an "annealing" scheme and was incorporated in the GE-SC learning with CCCP, since it is a simpler scheme compared to using the modified SQP method. The convergence of representation error $\|\mathbf{Y} - \mathbf{\Psi X}\|_F^2$ for this learning scheme is shown in Figure 3.4(a) and the convergence of the parameter λ is shown in Figure 3.4(b). Note that $1/\lambda$ indicates the trace ratio $\frac{\text{trace}(\mathbf{X}_t \mathbf{L}' \mathbf{X}_t^T)}{\text{trace}(\mathbf{X}_t \mathbf{L} \mathbf{X}_t^T)}$, and a high value for the trace ratio implies high discrimination across classes.

3.5 Simulation Results - Unsupervised Clustering

The first set of simulations study the behavior of GE-SCs obtained using (3.1) in unsupervised clustering. We compare it with clusterings obtained using LPP. The number of neighbors for LPP was fixed at 10, and the Laplacian $\mathbf{L}$ of LPP was used to compute GE-SCs also. The non-negative GE-SC $\mathbf{X}$ were obtained with parameters $\beta = 1$ and the parameter γ was tuned for best performance. The Gramm matrix $\mathbf{X}^T\mathbf{X}$ obtained from the sparse codes is shown in Figure 3.5 for the USPS dataset [3]. The dataset has 10 classes and clearly, the codes of different classes are uncorrelated, whereas, the codes belong to the same class are highly correlated. From (3.1), it can be seen that the GE-SCs consider the local neighborhood structure using the penalty $\text{trace}(\mathbf{X}\mathbf{L}\mathbf{X}^T)$, and also the sparse subspace structure with the sparse coding term, $\|\mathbf{Y} - \mathbf{\Psi X}\|_F^2 + \gamma \sum_i \|\mathbf{x}_i\|_1$. This can be contrasted to schemes such as LPP, which consider only the local neighborhood structure, and ℓ_1 graphs [19], that consider only the sparse subspace structure, when clustering the data. Spectral Clustering

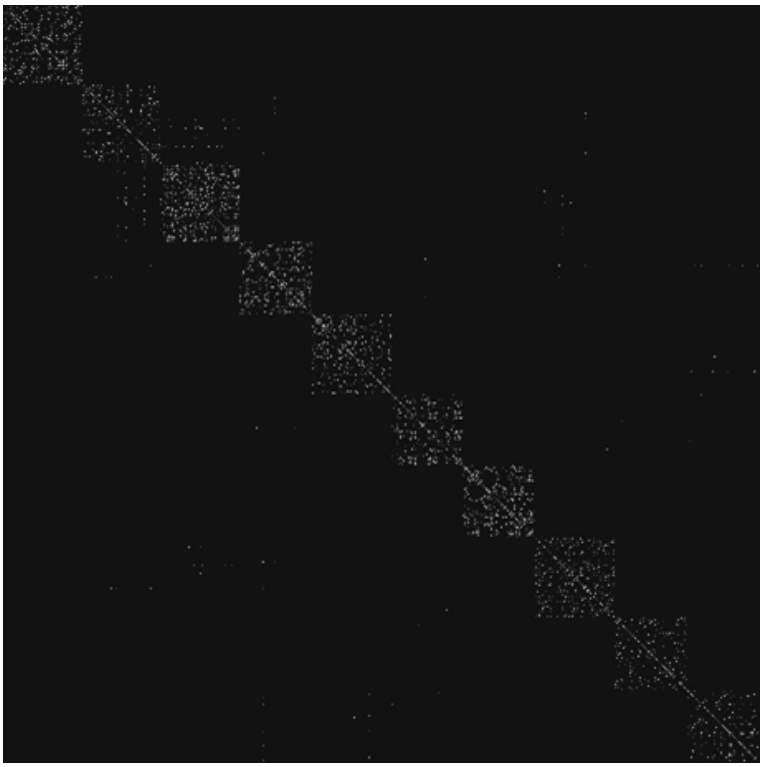


Figure 3.5: The structure of the Gramm matrix of the sparse codes when performing unsupervised GE-SC. The sparse codes here are obtained from 10 classes of the USPS dataset [3] and clearly, the codes of different clusters are unccorrelated.

was performed with GE-SCs using $\mathbf{X}^T\mathbf{X}$ as the weight matrix [105]. The clustering performance was evaluated for the standard datasets [3, 106] whose attributes are given in Table 3.2, with accuracy (ACC) and normalized mutual information (NMI) as the measures. For each dataset, the clustering was performed 50 times and the average and maximum performances were measured. Results in Table 3.3 show that the codes obtained using the proposed approach result in higher maximum and average performances compared to LPP. We also noted when performing the experiments that the performance of the GE-SC codes was immune to the choice of number of neighbors when building the graph, to a good extent.

Table 3.2: Datasets used in clustering.

Dataset	Dimensions	Instances	Clusters
Digits	16	10992	10
Soybean	35	562	15
Segment	19	2309	7
Satimage	36	6435	6
USPS	256	9298	10

3.6 Simulation Results - Face Recognition

The proposed algorithm was tested with the AR face database in supervised and semi-supervised classification. The database has 100 classes with 26 examples per class. 504 random projections were obtained from each image and used as the feature vector. The dictionary $\mathbf{\Psi}$ consisted of 500 columns. All experiments were repeated with 5 random train and test sets to increase the reliability of the results. For both supervised and semi-supervised learning, dictionaries were learned using the proposed approach with the parameters $\gamma = 0.01, \beta = 0.005$. The parameter δ_t was fixed at 30 in the first iteration and reduced up to 0.5 in the last iteration of training. Classification was performed using the unconstrained sparse codes generated from the learned dictionary with the parameter $\gamma = 0.003$ for both train and test datasets.

Table 3.3: Comparison of unsupervised clustering with graphs obtained from LPP and the proposed GE-SC approach.

Dataset	**ACC**				**NMI**			
	LPP		**GE-SC**		**LPP**		**GE-SC**	
	max	**avg**	**max**	**avg**	**max**	**avg**	**max**	**avg**
Digits	78.6	70.5	79.9	72	71	69	73	70
Soybean	70.46	64.2	72.8	65.3	76	72	78	74
Segment	57.6	53.3	71.1	68.2	54	49	65	64
Satimage	74.9	73.8	76.72	75.8	64	63	65	64
USPS	73.3	65.1	76.9	70.11	67	65	69	66

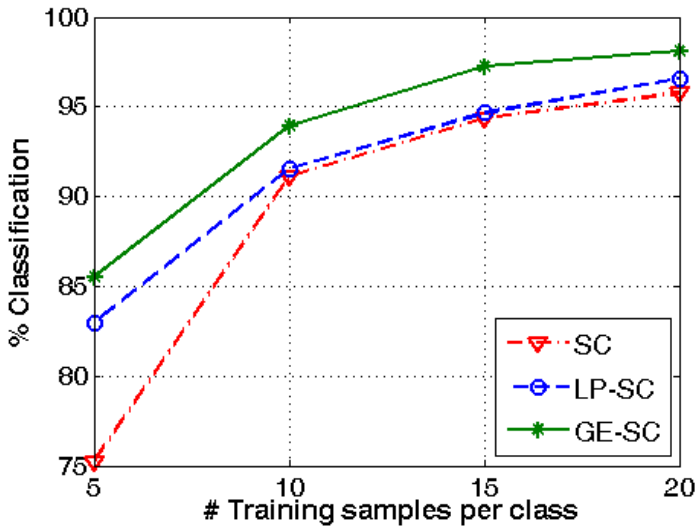


Figure 3.6: Performance of SC, unsupervised GE-SC, and supervised GE-SC based classification schemes for various number of training samples per class.

3.6.1 Supervised Learning

For supervised learning, we used 20 samples per class for training and the remaining 6 for testing. The recognition performance is reported in Table 3.4. Supervised GE-SC is compared with unconstrained sparse coding (SC), unsupervised GE-SC as well as with LDE. For all the approaches, classification was performed using linear support vector machines. Figure 3.6 shows that the supervised GE-SC scheme consistently works better compared to the SC and unsupervised GE-SC methods, for different number of training samples per class.

Table 3.4: Recognition results for AR face database. For LDE and supervised GE-SC, the number of neighbors $k = k' = 4$, for unsupervised GE-SC, k is fixed at 4, and for LDE, the number of reduced dimensions $d = 150$.

Algorithm	**% Classification**
Local disc. embedding	97.55
Sparse coding	95.77
Locality preserving SC	96.57
Graph embedded SC	**98.12**

Table 3.5: Comparison of face recognition performance using semi-supervised learning in a subset of AR faces. The parameters used in the algorithms are $d = 150$, $k = k' = 2$ and $\alpha = 0.1$.

Algorithm	**% Classification**
Semi-supervised disc. analysis	89.80
Sparse coding	89.62
Unsupervised GE-SC	90.42
Supervised GE-SC	**96.13**

3.6.2 Semi-supervised Learning

We used the unoccluded subset (14 images per class) of AR faces for semi-supervised learning. The training set consisted of 6 labeled examples, and 4 unlabeled examples. The unlabeled examples along with the remaining 4 examples were used for testing. The performance of various approaches for this dataset is reported in Table 3.5. The graph for SDA and supervised GE-SC was constructed using the procedure described in Section 3.2.2. Note that GE-SC results in the best performance followed by unsupervised GE-SC which used an unlabeled graph on the training set with $k = 2$.

3.7 Conclusions

We proposed algorithms to compute graph-embedded sparse codes and demonstrated their use in unsupervised, supervised and semi-supervised learning. Unconstrained sparse coding does not result in discriminative codes, although it may improve the performance of learning algorithms by rejecting "noise" in the data that hampers the learning performance. Including neighborhood relations using unsupervised graph constraints stabilizes the sparse codes and hence it usually results in an improved performance, compared to using unconstrained codes. Explicit discriminative graph constraints posed on sparse codes result in a *discriminative sparse embedding*, which may have utility in several learning applications.

Chapter 4

ENSEMBLE SPARSE MODELS FOR IMAGE ANALYSIS

4.1 Introduction

Sparse representations using learned dictionaries have been successful in several image analysis applications. In this chapter, we propose and analyze the framework of ensemble sparse models, and demonstrate their utility in image restoration and unsupervised clustering. The proposed ensemble model is an additive linear combination of multiple *weak* sparse models. Theoretical analysis of the ensemble model reveals that even in the worst-case, the ensemble model performs better than any of its constituent individual models. The dictionaries corresponding to the individual sparse models are obtained using either random selection or boosted approaches. In the random selection approach, each dictionary consists of a random subset of normalized training examples. Boosted approaches learn one dictionary per round such that the dictionary learned in a particular round is optimized for the training examples having high reconstruction error in the previous round. The final restored data is computed as a linear combination of the individual approximations. Results with compressed recovery of standard images show that the ensemble representations lead to a better performance compared to using a single dictionary obtained with the conventional alternating minimization approach. The proposed ensemble models are also used for single image superresolution and we show that they perform comparably to the recent approaches. Furthermore, learning ensemble models using the random selection approach incurs very little computational complexity. In unsupervised clustering, experiments show that the proposed model performs better than baseline approaches in several standard datasets. The work discussed in this chapter has been reported in [56] and [57].

Using the linear generative model, the sparse code of a data sample $\mathbf{x}$ can be obtained by optimizing,

$$h(\mathbf{x}, \mathbf{D}) = \min_{\mathbf{a}} \|\mathbf{x} - \mathbf{D}\mathbf{a}\|_2^2 + \lambda \|\mathbf{a}\|_1. \tag{4.1}$$

Here $\|\mathbf{a}\|_1$ is the ℓ_1 penalty that promotes sparsity of the coefficients, and the equivalence of (4.1) to ℓ_0 minimization has been discussed in [64] under some strong conditions on the dictionary $\mathbf{D}$. Some methods to obtain sparse representations used include the Matching Pursuit (MP) [107], Orthogonal Matching Pursuit (OMP) [108], Order-Recursive Matching Pursuit [109], Basis Pursuit (BP) [110], FOCUSS [111] and iterated shrinkage algorithms [112, 113].

In several image processing andsample machine learning applications, it is advantageous to learn a dictionary, such that the set of training examples obtained from a probability space have a small approximation error with sparse coding. This problem can be expressed as minimizing the objective [114]

$$g(\mathbf{D}) = \mathbf{E}_{\mathbf{x}}[h(\mathbf{x}, \mathbf{D})], \tag{4.2}$$

where the columns of $\mathbf{D}$, referred to as dictionary atoms, are constrained to have unit ℓ_2 norm, i.e., $\|\mathbf{d}_j\|_2 \leq 1, \forall j$. If the distribution in the probability space is unknown and we only have T training examples $\{\mathbf{x}_i\}_{i=1}^T$, each with probability mass $p(\mathbf{x}_i)$, (4.2) can be modified as the empirical cost function,

$$\hat{g}(\mathbf{D}) = \sum_{i=1}^{T} h(\mathbf{x}_i, \mathbf{D}) p(\mathbf{x}_i). \tag{4.3}$$

Typically dictionary learning algorithms solve for the sparse codes [101, 115] using (4.1), and obtain the dictionary by minimizing $\hat{g}(\mathbf{D})$, repeating the steps until convergence. We refer to this baseline algorithm as *Alt-Opt*. Since this is an alternating minimization process, it is important to provide a good initial dictionary and this is performed by setting the atoms to normalized cluster centers of the data [13].

Instead of learning dictionaries using sophisticated learning algorithms, it is possible to use the training examples themselves as the dictionary. Since the number of examples T is usually much larger than the number of dictionary atoms K, it is much more computationally intensive to obtain sparse representations with examples. Nevertheless, both learned and example-based dictionaries have found applications in inverse problems [2, 11, 14] and also in machine learning applications such as clustering and classification [15, 17–19, 116–122].

4.1.1 Ensemble Sparse Models

In this chapter, we propose and explore the framework of ensemble sparse models, where we assume that data can be represented using a linear combination of L different sparse approximations, instead of being represented using an approximation obtained from a single sparse model. The approximation to $\mathbf{x}$ can be obtained by optimizing

$$\min_{\{\beta_l\}_{l=1}^L} \|\mathbf{x} - \sum_{l=1}^{L} \beta_l \mathbf{D}_l \mathbf{a}_l\|_2^2. \tag{4.4}$$

Here each coefficient vector $\mathbf{a}_l$ is assumed to be sparse, and is obtained by solving for the optimization (4.1) with $\mathbf{D}_l$ as the dictionary. The weights $\{\beta_l\}_{l=1}^L$ control the contribution of each base model to the ensemble.

Since the ensemble combines the contributions of multiple models, it is sufficient that the dictionary for model is obtained using a "weak" training procedure. We propose to learn these *weak dictionaries* $\{\mathbf{D}_l\}_{l=1}^L$ sequentially, using a greedy forward selection procedure, such that training examples that incurred a high approximation error with the dictionary $\mathbf{D}_l$ are given more importance while learning $\mathbf{D}_{l+1}$. Furthermore, we also propose an ensemble model where each individual dictionary is designed as a random subset of training samples. The formulations described in this chapter belong to the category of boosting [31] and random selection algorithms [29] in machine learning. In supervised learning, boosting is used to improve the accu-

racy of learning algorithms, using multiple weak hypotheses instead of a single strong hypothesis. The proposed ensemble sparse models are geared towards two image analysis problems, the inverse problem of restoring degraded images, and the problem of unsupervised clustering. Note that, boosted ensemble models have been used with the bag-of-words approach for updating codebooks in classification [123] and medical image retrieval [124]. However, it has not been used so far in sparsity based image restoration problems or unsupervised clustering. Also when compared to [125], where the authors propose to obtain multiple randomized sparse representations from a single dictionary, in our approach, we propose to learn an ensemble of dictionaries and obtain a single representation from each of them. Typical ensemble methods for regression [33] modify the samples in each round of leveraging, whereas in our case the same training set is used for each round.

4.1.2 Contributions

In this work, we propose the framework of ensemble sparse models and perform a theoretical analysis that relates their performance when compared to its constituent base sparse models. We show that, even in the worst case, an ensemble will perform at least as well as its best constituent sparse model. Experimental demonstrations that support this theory are also provided. We propose two approaches for learning the ensemble: (a) using a random selection and averaging (*RandExAv*) approach, where each dictionary is chosen as a random subset of the training examples, and (b) using a boosted approach to learn dictionaries sequentially by modifying the probability masses of the training examples in each round of learning. In the boosted approach, two methods to learn the weak dictionaries for the individual sparse models, one that performs example selection using the probability distribution on the training set (*BoostEx*), and the other that uses a weighted K-means approach (*BoostKM*), are provided. For all cases of ensemble learning, we also provide methods to obtain the

ensemble weights, $\{\beta_l\}_{l=1}^{L}$, from the training examples. Demonstrations that show the convergence of ensemble learning, with the increase in the number of constituent sparse models are provided. Experiments also show that the proposed ensemble approaches perform better than their best constituent sparse models, as predicted by theory.

In order to demonstrate the effectiveness of the proposed ensemble models, we explore its application to image recovery and clustering. The image recovery problems that we consider here are compressive sensing using random projections and single image superresolution. When boosted ensemble models are learned for image recovery problems, the form of degradation operator specific to the application is also considered, thereby optimizing the ensemble for the application. For compressive recovery, we compare the performance of the proposed Random Example Averaging (*RandExAv*), Boosted Example (*BoostEx*), and Boosted K-Means (*BoostKM*) approaches to the single sparse model, whose dictionary is obtained using the *Alt-Opt* approach. It is shown that the ensemble methods perform consistently better than a single sparse model at different number of measurements. Note that, the base sparse model for example-based approaches is designed as a random subset of examples, and hence it requires minimal training. Furthermore, in image superresolution, the performance of the proposed ensemble learning approaches is comparable to the recent sparse representation methods [1], [2].

Furthermore, we explore the use of the proposed approaches in unsupervised clustering. When the data are clustered along unions of subspaces, an ℓ_1 graph [19] can be obtained by representing each data sample $\mathbf{x}_i$ as a sparse linear combination of the rest of the samples in the set. Another approach proposed in [117] computes the sparse coding-based graph using codes obtained with a learned dictionary. We propose to use ensemble methods to compute sparse codes for each data sample, and perform spectral clustering using graphs obtained from them. Results with sev-

eral standard datasets show that high clustering performance is obtained using the proposed approach when compared to ℓ_1 graph-based clustering.

4.2 Analysis of Ensemble Models

We will begin by motivating the need for an ensemble model in place of a single sparse model, and then proceed to derive some theoretical guarantees on the ensemble model. Some demonstrations on the performance of ensemble models will also be provided.

4.2.1 Need for the Ensemble Model

In several scenarios, a single sparse model may be insufficient for representing the data, and using an ensemble model instead may result in a good performance. The need for ensemble models in supervised learning have been well-studied [86]. We will argue that the same set of reasons apply to the case of ensemble sparse models also. The first reason is statistical, whereby several sparse models may have a similar training error when learned from a limited number of training samples. However, the performance of each of these models with test data can be poor. By averaging representations obtained from an ensemble, we may obtain an approximation closer to the true test data. The second reason is computational, which can occur with the case of large training sets also. The inherent issue in this case is that sparse modeling is a problem with locally optimal solution. Therefore, we may never be able to reach the global optimal solution with a single model and hence using an ensemble model may result in a lesser representation error. Note that this case is quite common in dictionary learning, since many dictionary learning algorithms only seek a local optimal solution. The third reason for using an ensemble model is representational, wherein the hypothesis space assumed cannot represent the test data sample. In the case of sparse models, this corresponds to the case where the dictionary cannot provide a high-fidelity sparse approximation for a novel test data sample. This also happens in the case where the test observation is a corrupted version of the underlying

test data, and there is ambiguity in obtaining a sparse representation. In this case also, it may be necessary to combine multiple sparse models to improve the estimate of the test data.

In order to simplify notation in the following analysis, let us denote the l^{th} approximation in the ensemble model as $\mathbf{c}_l = \mathbf{D}_l\mathbf{a}_l$. The individual approximations are stacked in the matrix $\mathbf{C} \in \mathbb{R}^{M\times L}$, where $\mathbf{C} = [\mathbf{c}_1 \dots \mathbf{c}_L]$ and the weight vector is denoted as $\boldsymbol{\beta} = [\beta_1 \dots \beta_L]^T$. The individual residuals are denoted as $\mathbf{r}_l = \mathbf{x} - \mathbf{c}_l$, for $i = 1, \dots, L$, and the total residual of the approximation is given as

$$\mathbf{r} = \mathbf{x} - \mathbf{C}\boldsymbol{\beta}. \tag{4.5}$$

We characterize the behavior of the ensemble sparse model by considering four different cases for the weights $\{\beta_l\}_{l=1}^L$.

Case 1: Weights are Unconstrained

In this case, the ensemble weights $\{\beta_l\}_{l=1}^L$ are assumed to be unconstrained and computed using the unconstrained least squares approximation

$$\min_{\boldsymbol{\beta}} \|\mathbf{x} - \mathbf{C}\boldsymbol{\beta}\|_2^2 \tag{4.6}$$

When the data $\mathbf{x}$ lies in the span of $\mathbf{C}$, the residual will be zero, i.e., $\mathbf{r} = 0$. The residual that has minimum energy in the L approximations is denoted as $\mathbf{r}_{min}$. This residual can be obtained by setting the corresponding weight in the vector $\boldsymbol{\beta}$ to be 1, whereas (4.6) computes $\boldsymbol{\beta}$ that achieves the best possible residual $\mathbf{r}$ for the total approximation. Clearly this implies

$$\|\mathbf{r}\|_2 \leq \|\mathbf{r}_{min}\|_2. \tag{4.7}$$

Therefore, at worst, the total approximation will be as good as the best individual approximation.

Case 2: Weights are Non-negative

The ensemble weights $\{\beta_l\}_{l=1}^{L}$ are assumed to be non-negative in this case. The least squares approximation (4.4), with the constraint $\boldsymbol{\beta} \geq 0$ will now result in a zero residual if the data $\mathbf{x}$ lies in the simplical cone generated by the columns of $\mathbf{C}$. The simplical cone is defined as the set $\{\mathbf{b} : \mathbf{b} = \sum_{l=1}^{L} \mathbf{c}_l \beta_l\}$. Otherwise, the bound on the total residual given by (4.7) holds in this case, since $\mathbf{r}_{min}$ can be obtained by setting the appropriate weight in $\boldsymbol{\beta}$ to 1 in (4.5), and the rest to 0 in this case also.

Case 3: Weights Sum to 1

When the ensemble weights are constrained to sum to 1, the total residual can be expressed as

$$\mathbf{r} = \sum_{l=1}^{L} \beta_l \mathbf{r}_l. \tag{4.8}$$

This can be easily obtained by replacing $\mathbf{x}$ as $\sum_{l=1}^{L} \beta_l \mathbf{x}$ in (4.5). Denoting the residual matrix $\mathbf{R} = [\mathbf{r}_1 \dots \mathbf{r}_L]$, the optimization (4.4) to compute the weights can also be posed as $\min_{\beta} \|\mathbf{R}\boldsymbol{\beta}\|_2$. Incorporating the constraint $\sum_{l=1}^{L} \beta_l = 1$, it can be seen that the final approximation $\mathbf{C}\boldsymbol{\beta}$ lies in the affine hull generated by the columns of $\mathbf{C}$, and the final residual, $\mathbf{R}\boldsymbol{\beta}$, lies in the affine hull generated by the columns of $\mathbf{R}$. Clearly the final residual will be zero, only if the data $\mathbf{x}$ lies in the affine hull of $\mathbf{C}$, or equivalently the zero vector lies in the affine hull of $\mathbf{R}$. When this does not hold, the worst case bound on $\mathbf{r}$ given by (4.7) holds in this case as well.

Case 4: Weights are Non-negative and Sum to 1

Similar to the previous case, the total residual can be expressed as (4.8). As a result, the final representation $\mathbf{C}\boldsymbol{\beta}$ lies in the convex hull generated by the columns of $\mathbf{C}$, and the final residual, $\mathbf{R}\boldsymbol{\beta}$, lies in the convex hull generated by the columns of $\mathbf{R}$. Furthermore, the final residual will be zero only if the zero vector lies in the convex hull of $\mathbf{R}$. Clearly, the worst case bound on $\mathbf{r}$ given by (4.7) holds in this case.

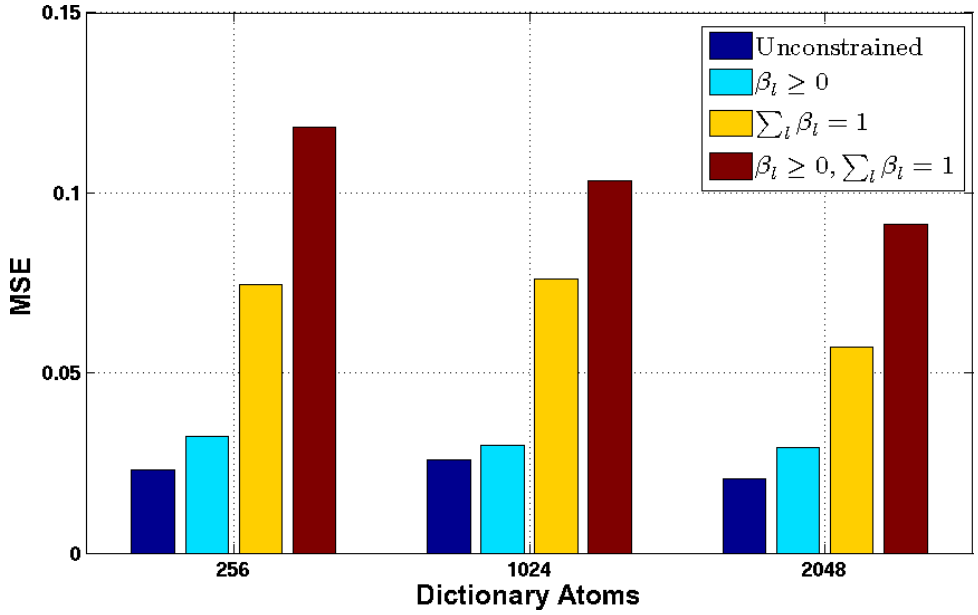


Figure 4.1: Performance of the "oracle" ensemble models for various constraints on weights and different dictionary sizes in the base models.

Although the worst case bounds for all the four cases are the same, the constraint spaces for the cases might provide us an idea about their relative performances with real data. The first case is unconstrained and it should result in the least error. The second case constrains that the solution should lie in the simplical cone spanned by the columns of $\mathbf{C}$, and this should lead to higher residual energy than Case 1. Case 3 constrains the solution to lie in an affine hull, which is of $L-1$ dimensions compared to simplical cone in L dimensions, so it could lead to a higher error compared to Case 2. Case 4 is the subset of constraint spaces for Cases 1 to 3 and hence it will lead to the highest residual error.

4.2.2 Demonstration of Ensemble Representations

In order to demonstrate the performance of ensemble representations with real data, we obtain a random set of $100,000$ patches, each of size 8×8, from a set of natural images. The training images were obtained from the superresolution toolbox published by Yang *et. al.* [126], and consist of a wide variety of patterns and textures.

Figure 4.2: Samples images from the training set. Note that the training images have a good mix of geometric patterns as well as textures.

A few images from the training set are shown in Figure 4.2. We will refer to this set of training images simply as the *training image set* throughout this chapter. The chosen patches are then processed to remove the mean, followed by the removal of low-variance patches. Since image recovery is the important application of the proposed models, considering high-variance patches alone is beneficial. Each dictionary in the ensemble $\mathbf{D}_i$ is obtained as a random set of K vectorized, and normalized patches. We fix the number of models in the ensemble as $L = 20$. The test data is a random set of 1000 grayscale patches obtained from the Berkeley segmentation dataset [127]. For each test sample, we compute the set of L approximations using the sparse model given in (4.1), with $\lambda = 0.2$. The individual approximations are combined into an ensemble, under the four conditions on the weights, $\{\beta_l\}$, described above. The optimal weights are computed and the mean squared norm of the residuals for all the test samples are compared in Figure 4.1, for the dictionary sizes $K = \{256, 1025, 2048\}$. We observe that the performance of the ensembles generally improve as the size of the dictionaries used in the base models increase. The variation in performance across all the four cases of weights follows our reasoning in the previous section. We refer to these as "oracle" ensemble models, since the weights are optimally computed with perfect knowledge of all the individual approximations and the actual data. In reality, the weights will be precomputed from the training data.

4.3 Proposed Ensemble Sparse Representation Algorithms

The ensemble model proposed in (4.4) results in a good approximation for any known data. However, in order to use ensemble models in analysis and recovery of images,

that are possibly corrupted or degraded, both the weights $\{\beta_l\}_{l=1}^{L}$ and the dictionaries $\{\mathbf{D}_l\}_{l=1}^{L}$ must be inferred from uncorrupted training data. The set of weights is fixed to be common for all test observations instead of computing a new set of weights for each observation. Let us denote the set of training samples as $\mathbf{X} = [\mathbf{x}_1\ \mathbf{x}_2\ \ldots\ \mathbf{x}_T]$, and the set of coefficients in base model l as $\mathbf{A}_l = [\mathbf{a}_{l,1}\ \mathbf{a}_{l,2}\ \ldots\ \mathbf{a}_{l,T}]$, where $\mathbf{a}_{l,i}$ is the coefficient vector of the i^{th} sample for base model l. In the proposed ensemble learning procedures, we consider both simple averaging and boosting approaches.

4.3.1 Random Example Averaging Approach

The first approach chooses L random subsets of K samples from the training data itself and normalizes them to form the dictionaries $\{\mathbf{D}_l\}_{l=1}^{L}$. The weights $\{\beta_l\}$ are chosen to be equal for all base models as $1/L$. Note that the selection of dictionaries follows the same procedure as given in the previous demonstration (Section 4.2.2). We refer to this ensemble approach as *Random Example Averaging* (*RandExAv*).

4.3.2 Boosting Approaches

The next two approaches use boosting and obtain the dictionaries and weights sequentially, such that the training examples that resulted in a poor performance with the $l-1^{\text{th}}$ base model are given more importance when learning the l^{th} base model. We use a greedy forward selection procedure for obtaining the dictionaries and the weights. In each round l, the model is augmented with one dictionary $\mathbf{D}_l$, and the weight α_l corresponding to the dictionary is obtained. The cumulative representation for round l is given by

$$\mathbf{X}_l = (1-\alpha_l)\mathbf{X}_{l-1} + \alpha_l \mathbf{D}_l \mathbf{A}_l. \tag{4.9}$$

Note that the weights of the greedy forward selection algorithm, α_l, and the weights of the ensemble model, β_l, are related as

$$\beta_l = \alpha_l \prod_{t=l+1}^{L} (1-\alpha_t). \tag{4.10}$$

From (4.9), it can be seen that $\mathbf{X}_l$ lies in the affine hull of $\mathbf{X}_{l-1}$ and $\mathbf{D}_l\mathbf{A}_l$. Furthermore, from the relationship between the weights $\{\alpha_l\}$ and $\{\beta_l\}$ given in (4.10), it is clear that $\sum_{l=1}^{L}\beta_l = 1$ and hence the ensemble model uses the constraints given in Case 3. Only the Cases 3 and 4 lead to an efficient greedy forward selection approach for the ensemble model in (4.4), and we use Case 3 since it leads to a better approximation performance (Figure 4.1).

In boosted ensemble learning, the importance of the training samples in a particular round is controlled by modifying their probability masses. Each round consists of (a) learning a dictionary $\mathbf{D}_l$ corresponding to the round, (b) computing the approximation for the current round l, (c) estimating the weight α_l, (d) computing the residual energy for the training samples, and (e) updating the probability masses of the training samples for the next round. Since the goal of ensemble approaches is to have only *weak* individual models, $\mathbf{D}_l$ is obtained using naive dictionary learning procedures as described later in this section. The dictionaries for the first round are obtained by fixing uniform probability masses for each training example in the first round, (i.e.), $p_1(\mathbf{x}_i) = 1/T$ for $i = \{1, 2, \ldots, T\}$. Assuming that $\mathbf{D}_l$ is known, the approximation for the current round is computed by coding the training samples $\mathbf{X}$ with the dictionary using (4.1). The weight α_l is computed such that the error between the training samples and the cumulative approximation $\mathbf{X}_l$ is minimized. Using (4.9), this optimization can be expressed as

$$\min_{\alpha_l} \|\mathbf{X}_l - [(1-\alpha_l)\mathbf{X}_{l-1} + \alpha_l\mathbf{D}_l\mathbf{A}_l]\|_F^2\,, \tag{4.11}$$

and can be solved in closed form with the optimal value given as,

$$\alpha_l = \frac{\mathrm{Tr}\left[(\mathbf{X}-\mathbf{X}_{l-1})^T(\mathbf{D}_l\mathbf{A}_l - \mathbf{X}_{l-1})\right]}{\|\mathbf{D}_l\mathbf{A}_l - \mathbf{X}_{l-1}\|_F^2}, \tag{4.12}$$

where Tr denotes the trace of the matrix. The residual matrix for all the training samples in round l is given by $\mathbf{R}_l = \mathbf{X} - \mathbf{D}_l\mathbf{A}_l$. The energy of the residual for the i^{th}

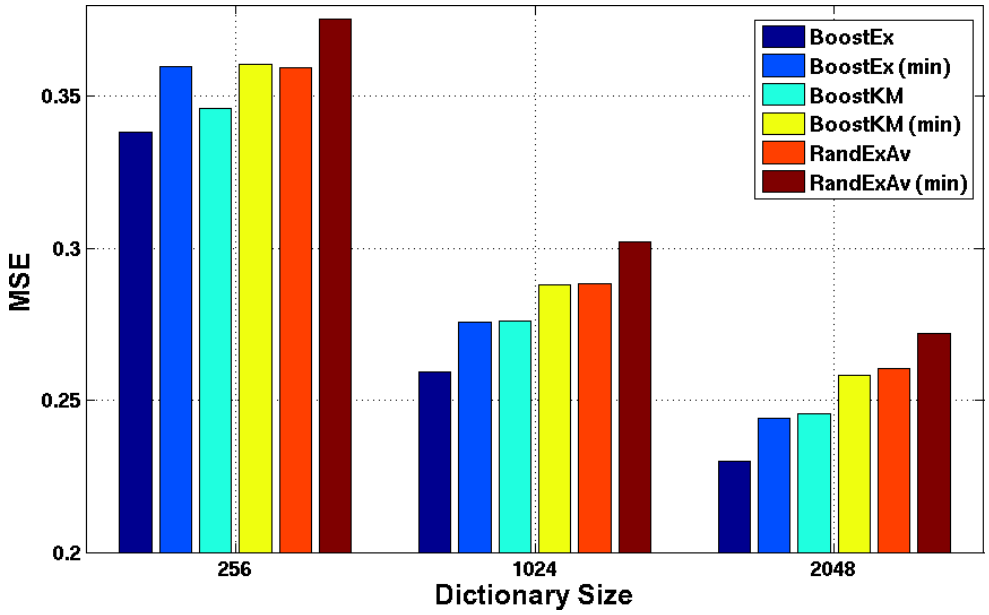


Figure 4.3: Comparison of the reconstruction performance of the proposed ensemble methods with test data, when various sizes of dictionaries are used in the base sparse models.

training sample is given as $e_l(i) = \|\mathbf{r}_{l,i}\|_2^2$. If the dictionary in round l provides a large approximation error for sample i, then that sample will be given more importance in round $l+1$. This will ensure that the residual error for sample i in round $l+1$ will be small. The simple scheme of updating the probability masses as $p_{l+1}(\mathbf{x}_i) = e_l(i)$, upweights the badly represented samples and downweights the well-represented ones for the next round.

Given a training set $\{\mathbf{x}_i\}_{i=1}^L$, and its probability masses $\{p_l(\mathbf{x_i})\}_{i=1}^L$, we will propose two simple approaches for learning the dictionaries corresponding to the individual sparse models.

BoostKM

When the sparse code for each training example is constrained to take one only one non-zero coefficient of value 1, and the norms of the dictionary atoms are unconstrained, the dictionary learning problem (4.3) can be shown to reduce to K-Means clustering. Hence, computing a set of K-Means cluster centers and normalizing them

to unit ℓ_2 norm constitutes a reasonable weak dictionary. However, since the distribution on the data could be non-uniform in our case, we need to alter the clustering scheme to incorporate this. Denoting the cluster centers to be $\{\boldsymbol{\mu}_k\}_{k=1}^K$, the cluster membership sets to be $\{\mathcal{M}_k\}_{k=1}^K$, the weighted K-Means objective is denoted as

$$\min_{\{\boldsymbol{\mu}_k\}_{k=1}^K, \{\mathcal{M}_k\}_{k=1}^K} \sum_{k=1}^{K} \sum_{i \in \mathcal{M}_k} p(\mathbf{x}_i) \|\mathbf{x}_i - \boldsymbol{\mu}_k\|_2^2. \tag{4.13}$$

The weighted K-Means procedure is implemented by modifying the scalable K-Means++ algorithm, also referred to as the K-Means‖ (K-Means Parallel) algorithm [128]. The K-Means‖ algorithm is an improvement over the K-Means++ algorithm [129] that provides a method for careful initialization leading to improved speed and accuracy in clustering. The advantage with the K-Means‖ algorithm is that the initialization procedure is scalable to a large number of samples. In fact, it has been shown in [128] that just the initialization procedure in K-Means‖ results in a significant reduction in the clustering cost. Since we are interested in learning only a weak dictionary, we will use the normalized cluster centers obtained after initialization as our dictionary. The K-Means++ algorithm selects initial cluster centers sequentially such that they are relatively spread out. For initializing K cluster centers, the algorithm creates a distribution on the data samples and picks a cluster center by sampling it and appends it to the current set of centers. The distribution is updated after each cluster center is selected. In contrast, the K-Means‖ algorithm updates the distribution much more infrequently, after choosing q cluster centers in each iteration. This process is repeated for s iterations, and finally the number of cluster centers obtained is sq. The chosen centers are re-clustered to obtain the initial set of K clusters. It is clear that s must be chosen such that $sq > K$. We provide only the initialization of the weighted K-Means‖ algorithm that takes the data distribution, $\{p_l(\mathbf{x}_i)\}_{i=1}^T$, also into consideration.

Let us denote δ_i as the shortest distance of the i^{th} training sample to the set of cluster centers already chosen. The initialization of the weighted K-Means‖ algorithm proceeds as follows:

(a) Initialize $\overline{\mathcal{M}} = \{\}$.

(b) Pick the first center $\boldsymbol{\mu}_1$ from the training set based on the distribution $\{p_l(\mathbf{x}_i)\}_{i=1}^{T}$, and append it to $\overline{\mathcal{M}}$.

(c) The set of intermediate cluster centers, $\mathcal{M}'$, is created using q samples from the data, $\{\mathbf{x}_i\}_{i=1}^{T}$, according to the probability $\frac{p_l(\mathbf{x}_i)\delta_i^2}{\sum_{j=1}^{T} p_l(\mathbf{x}_j)\delta_j^2}$.

(d) Augment the set $\overline{\mathcal{M}} \leftarrow \overline{\mathcal{M}} \cup \mathcal{M}'$.

(e) Repeat steps 2, 3 and 4 for s iterations.

(f) Set the weight of each element $\boldsymbol{\mu}$ in the set $\overline{\mathcal{M}}$, as the sum of weights of samples in $\mathbf{X}$ that are closer to $\boldsymbol{\mu}$ than any other sample in $\overline{\mathcal{M}}$.

(g) Perform weighted clustering on the elements of $\overline{\mathcal{M}}$ to obtain the set of K cluster centers, $\mathcal{M}$.

Note that the steps (b) and (c) are used to compute the initial cluster centers giving preference to samples with higher probability mass. Finally, each dictionary atom $\mathbf{d}_k$ is set as the normalized cluster center $\frac{\boldsymbol{\mu}_k}{\|\boldsymbol{\mu}_k\|_2}$.

BoostEx

From (4.3), it is clear that the learned dictionary atoms are close to training samples that have higher probabilities. Therefore, in the *BoostEx* method, the dictionary for round l is updated by choosing K data samples based on the non-uniform weight distribution, $\{p_l(\mathbf{x}_i)\}_{i=1}^{T}$, and normalizing them. This scheme will ensure that those samples with high approximation errors in the previous round, will be better represented in the current round.

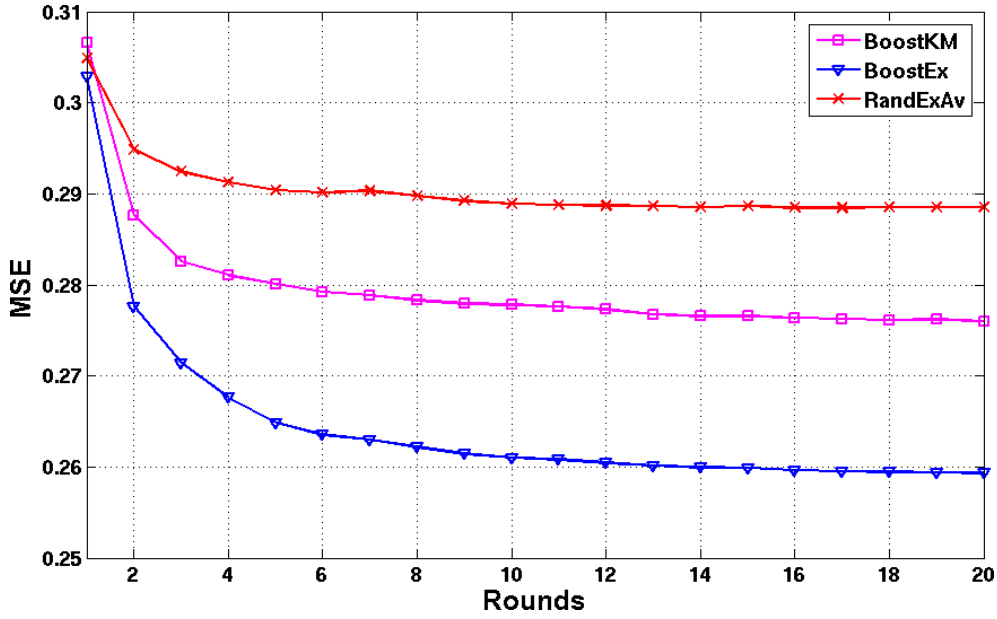


Figure 4.4: Convergence of approximation error with test data for proposed ensemble methods, when dictionaries of size 1024 are used with the base sparse models.

4.3.3 Demonstration of the Proposed Approaches

The performance of the proposed ensemble schemes for dictionaries of three different sizes $K = \{256, 1024, 2048\}$ are compared. The training set described in Section 4.2.2 is used with the *RandExAv*, *BoostKM*, and *BoostEx* schemes. The dictionaries $\{\mathbf{D}_l\}_{l=1}^{L}$ and the weights $\{\beta_l\}_{l=1}^{L}$ are obtained with the above schemes for $L = 20$. The individual approximations in the training set are obtained using (4.1) with the sparsity penalty set as $\lambda = 0.2$. For each sample in the test set described in Section 4.2.2, the individual representations are computed using (4.1) with $\lambda = 0.2$. The ensemble approximation the i^{th} test sample is obtained as $\sum_{l=1}^{L} \beta_l \mathbf{D}_l \mathbf{a}_{l,i}$. Figure 4.3 compares the performances of the proposed schemes for different dictionary sizes. The minimum error obtained across all individual approximations is also shown for comparison, with all the three methods and the different dictionary sizes. It can be seen that the proposed schemes satisfy the basic property of the ensemble discussed in Section 4.2, where it has been shown that the ensemble approximation performs

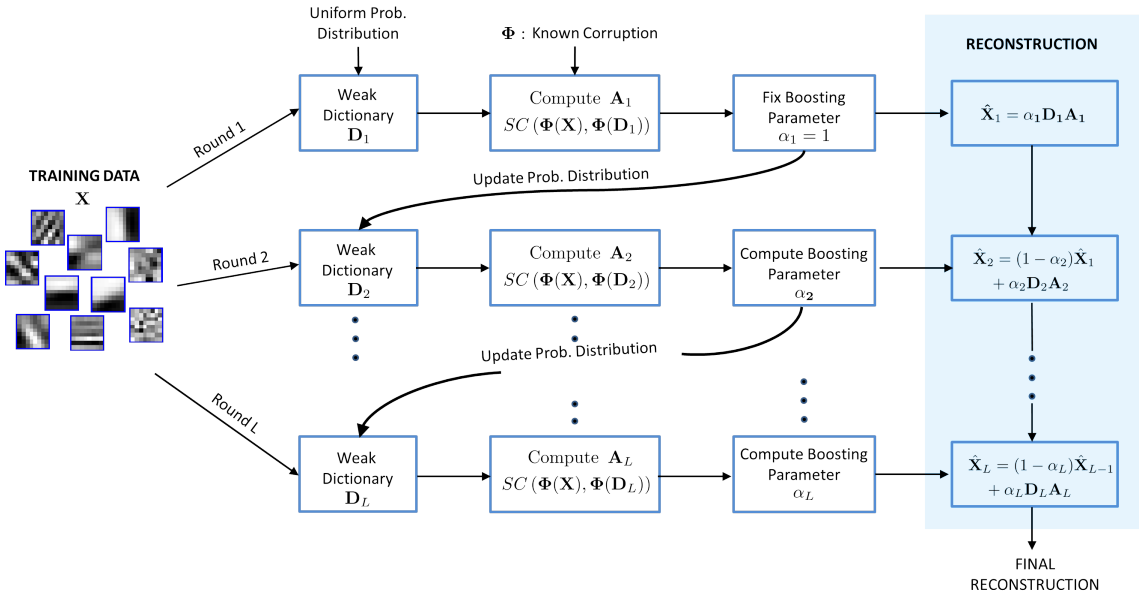


Figure 4.5: Illustration of the proposed boosted dictionary learning for image restoration. SC denotes sparse coding using (4.15).

better than the best constituent individual approximation. As the number of number of approximations in the ensemble increase, the average mean squared error (MSE) for the three proposed methods reduce, as shown in Figure 4.4 for a dictionary size of 1024. Clearly, increasing the number of models in the ensemble results in a better approximation, but the MSE flattens out as the number of rounds increase.

4.4 Application: Image Restoration

In restoration applications, it is necessary to solve an inverse problem, in order to estimate the test data $\mathbf{y}$ from

$$\mathbf{z} = \mathbf{\Phi}(\mathbf{y}) + \mathbf{n}, \tag{4.14}$$

where $\mathbf{\Phi}(.)$ is the corruption operator and $\mathbf{n}$ is the additive noise. If the operator $\mathbf{\Phi}(.)$ is linear, we can represent it using the matrix $\mathbf{\Phi}$. With the prior knowledge that $\mathbf{y}$ is sparsely representable in a dictionary $\mathbf{D}$ according to (4.1), (4.14) can be expressed as $\mathbf{z} = \mathbf{\Phi D a} + \mathbf{n}$. Restoring $\mathbf{x}$ now reduces to computing the sparse codes

$\mathbf{a}$ by solving

$$\min_{\mathbf{a}} \|\mathbf{z} - \mathbf{\Phi D a}\|_2^2 + \lambda \|\mathbf{a}\|_1. \tag{4.15}$$

and finally estimating $\mathbf{y} = \mathbf{Da}$ [11]. In the proposed ensemble methods, the final estimate of $\mathbf{x}$ is obtained as a weighted average of the individual approximations. Furthermore, in the boosting approaches, *BoostKM* and *BoostEx*, the degradation operation can be included when learning the ensemble. This is achieved by degrading the training data as $\mathbf{\Phi X}$, and obtaining the approximation with the coefficients computed using (4.15) instead of (4.1). The procedure to obtain boosted dictionaries using degraded data and computing the final approximation is illustrated in Figure 4.5. In this figure, the final approximation is estimated sequentially using the weights $\{\alpha_l\}_{l=1}^L$, but it is equivalent to computing $\{\beta_l\}_{l=1}^L$ using (4.10) and computing the ensemble estimate $\sum_{l=1}^L \beta_l \mathbf{D}_l \mathbf{A}_l$. The set of test images that we use to test our restoration framework are given in Figure 4.6.

4.4.1 Compressive Recovery

In compressed sensing (CS), the $N-$dimensional observation $\mathbf{z}$ is obtained by projecting the $M-$dimensional data $\mathbf{y}$ onto a random linear subspace, where $N \ll M$ [130]. In this case, the entries of the degradation matrix $\mathbf{\Phi} \in \mathbb{R}^{N \times M}$ are obtained as i.i.d. realizations of a Gaussian or Bernoulli random variable. Compressive recovery can be effectively performed using conventional dictionaries or ensemble dictionaries. In addition, the proposed idea of ensemble learning can be incorporated in existing learning schemes to achieve improved recovery performance. In particular, the multilevel dictionary learning algorithm [12] can be very easily adapted to compute ensemble representations. Before discussing the experimental setup, and the results of the proposed methods, we will describe the modification to multilevel dictionary learning for improving the compressed recovery performance with learned dictionaries.

Table 4.1: Compressed recovery of standard images: PSNR (dB) obtained using Alternating Dictionary Optimization (*Alt-Opt*), BoostEx (*BEx*), BoostKM (*BKM*), RandExAv (*RExAv*), and *Ex-MLD* methods, for different values of N. The results reported were obtained by averaging over 10 iterations with different random measurement matrices. In each, the higher PSNR is given in bold font.

Image	**Number of Measurements (N)**														
	N = 8					**N = 16**					**N = 32**				
	Alt-Opt	**BEx**	**BKM**	**RExAv**	**Ex-MLD**	**Alt-Opt**	**BEx**	**BKM**	**RExAv**	**Ex-MLD**	**Alt-Opt**	**BEx**	**BKM**	**RExAv**	**Ex-MLD**
Barbara	21.55	22.05	22.04	22.08	**22.95**	23.52	23.86	23.73	23.68	**24.39**	26.45	26.53	26.44	26.28	**26.66**
Boat	23.08	23.73	23.95	23.99	**25.08**	25.91	26.29	26.57	26.59	**26.96**	28.79	29.32	29.61	29.61	**29.9**
Couple	23.15	23.81	24.02	24.05	**25**	25.87	26.30	26.57	26.56	**27.19**	28.83	29.33	29.68	29.67	**29.8**
Fingerprint	18.10	18.76	19.15	19.16	**20.39**	21.74	22.18	22.84	22.86	**23.19**	25.36	25.85	26.59	26.63	**26.82**
House	24.52	25.12	25.51	25.52	**26.55**	28.01	28.14	28.63	28.66	**28.93**	31.28	31.53	32.01	32.03	**32.25**
Lena	25.14	25.84	26.18	26.25	**27.17**	28.31	28.73	29.08	29.13	**29.59**	31.12	31.77	32.15	32.17	**32.65**
Man	23.90	24.60	24.83	24.89	**25.84**	26.60	27.12	27.35	27.40	**27.68**	29.45	30.14	30.40	30.42	**30.67**
Peppers	21.31	21.83	22.17	22.23	**23.12**	24.30	24.54	24.82	24.91	**25.68**	27.28	27.69	28.03	28.11	**28.57**

Figure 4.6: The set of standard images used in our experiments on image restoration. In a raster-scan fashion, the images are: *Barbara*, *Boat*, *Cameraman*, *Couple*, *Fingerprint*, *Girl*, *House*, *Lena*, *Man*, *Peppers*, and *Straw*.

Improved Multilevel Dictionaries

The multilevel dictionary (MLD) learning algorithm is a hierarchical procedure where the dictionary atoms in each level are obtained using a 1-D subspace clustering procedure [12]. Multilevel dictionaries have been shown to generalize well to novel test data, and have resulted in high performance in compressive recovery. We propose to employ the *RandExAv* procedure in each level of multilevel learning to reduce overfitting and thereby improve the accuracy of the dictionaries in representing novel test samples. In each level, L different dictionaries are drawn as random subsets of normalized training samples. For each training sample, a $1-$sparse representation is computed with each individual dictionary, and the approximations are averaged to obtain the ensemble representation for that level. Using the residual vectors as the training data, this process is repeated for multiple levels. The sparse approximation for a test sample is computed in a similar fashion. Since the sparse code computation in each level is performed using simple correlation operations, the computation complexity is not increased significantly by employing ensemble learning. In our simulations, we will refer to this approach as Example-based Multilevel Dictionary learning (*Ex-MLD*).

Alt-Opt (24.81 dB)

BEx (25.7 dB)

BKM (26.06 dB)

RExAv (26.15 dB)

Ex-MLD (27.19 dB)

Figure 4.7: Compressed recovery of *Lena* image for $N = 8$ measurements using Alternating Dictionary Optimization (*Alt-Opt*), BoostEx (*BEx*), BoostKM (*BKM*), RandExAv (*RExAv*) and Example-based MLD (*Ex-MLD*) approaches. Reconstructed images and PSNRs are shown.

Results

The training set is the same as that described in Section 4.2.2. For the baseline *Alt-Opt* approach, we train a single dictionary with $K = 256$ using 100 iterations with the sparsity penalty λ_{tr} set to 0.1. The ensemble learning procedures *BoostEx*, *BoostKM* and *RandExAv* are trained with $L = 50$ and $K = 256$ for sparsity penalty $\lambda_{tr} = 0.1$. The boosted ensembles are trained by taking the random projection operator into consideration, as discussed in Section 4.4 for the reduced measurements, $N = \{8, 16, 32\}$. For the *Ex-MLD* method, both the number of levels and the number of atoms in each level were fixed at 16. In each level, we obtained $L = 50$ dictionaries to compute the ensemble representation.

The recovery performance of the proposed ensemble models is evaluated using the set of standard images shown in Table 4.1. Each image is divided into non-overlapping patches of size 8×8, and random projection is performed with the number of measurements set at $N = \{8, 16, 32\}$. For the *Alt-Opt* procedure, the individual patches are recovered using (4.15), and for the ensemble methods, the approximations computed using the L individual dictionaries are combined. The penalty λ_{te} is set to 0.1 for sparse coding in all cases. For each method, the PSNR values were obtained by averaging the results over 10 iterations with different random measurement matrices, and the results are reported in Table 4.1. It was observed that the proposed ensemble methods outperform the *Alt-Opt* methods in all cases. In particular, we note that the simple *RandExAv* performs better than the boosting approaches, although in Section 4.2.2 it was shown that boosting approaches show a superior performance. The reason for this discrepancy is that boosting aggressively reduces error with training data, and hence may lead to overfitting with degraded test data. Whereas, the *RandExAv* method provides the same importance to all individual approximations both during the training and the testing phases. As a result, it provides a better

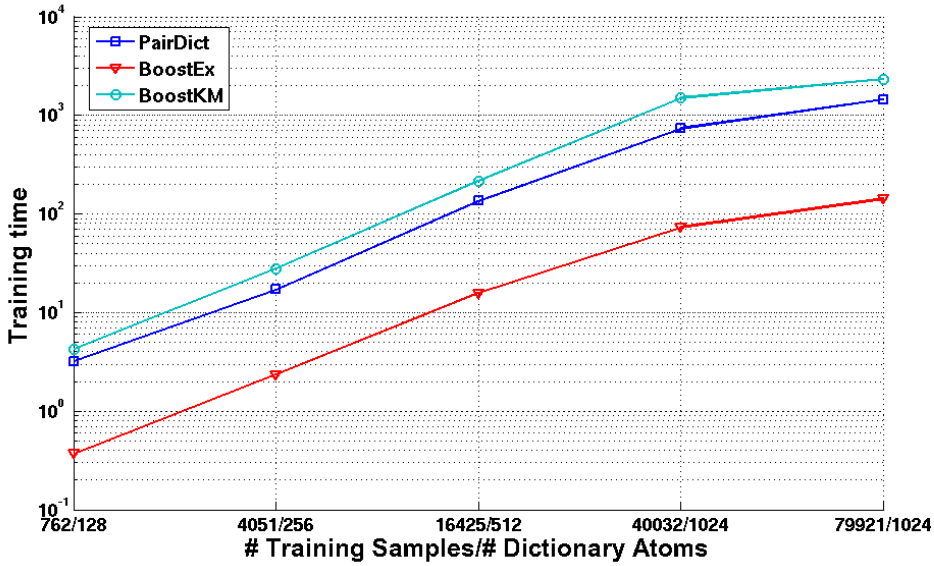


Figure 4.8: Effect of dictionary and training set sizes on the dictionary training time for different learning schemes. The training times given are in seconds and are compared only for *PairDict* (40 iterations), *BoostEx* ($L = 50$) and *BoostKM* ($L = 50$) since *ExDict* and *RandExAv* require no training.

generalization in the presence of degradation. We also note that similar behavior has been observed with ensemble classification methods [90]. Random sampling methods such as bagging perform better than boosting with noisy examples, since bagging exploits classification noise to produce more diverse classifiers. Furthermore, we observed that the proposed *Ex-MLD* method performed significantly better than all approaches, particularly for lower number of measurements. Figure 4.7 shows the images recovered using the different approaches, when N was fixed at 8. As it can be observed, the *Ex-MLD* and *RandExAv* methods provide PSNR gains of 2.38dB and 1.34dB respectively, when compared to the *Alt-Opt* approach.

4.4.2 Single Image Superresolution

Single image superresolution (SISR) attempts to reconstruct a high-resolution image using just a single low-resolution image. It is a severely ill-posed problem and in sparse representation based approaches, the prior knowledge that natural image

patches can be represented as a sparse linear combination of elementary patches, is used. The degraded test image is represented as $\mathbf{Z} = \mathbf{\Phi Y}$, where the operator $\mathbf{\Phi}$ the blurs the high-resolution image $\mathbf{Y}$ and then downsamples it. Note that $\mathbf{Y}$ and $\mathbf{Z}$ denote vectorized high- and low-resolution images respectively. Each overlapping patch obtained from the degraded image is denoted as $\mathbf{z}$. The paired dictionary learning procedure (*PairDict*) proposed in [1] has been very effective in recovering the high-resolution patches. This method initially creates degraded counterparts of the high-resolution training images, following which gradient-based features are extracted from the low-resolution patches and the features are appended to the corresponding vectorized high-resolution patches. These augmented features are used to train a paired dictionary $\begin{pmatrix} \mathbf{D}_{lo} \\ \mathbf{D}_{hi} \end{pmatrix}$ such that each low-resolution and its corresponding high-resolution patches share the same sparse code. For a low-resolution test patch $\mathbf{z}$, the sparse code $\mathbf{a}$ is obtained using $\mathbf{D}_{lo}$, and the corresponding high-resolution counterpart is recovered as $\mathbf{y} = \mathbf{D}_{hi}\mathbf{a}$. An initial estimate $\mathbf{Y}_0$ of the high-resolution image is obtained by appropriately averaging the overlapped high-resolution patches. Finally, a global reconstruction constraint is enforced by projecting $\mathbf{Y}_0$ on to the solution space of $\mathbf{\Phi Y} = \mathbf{Z}$,

$$\min_{\mathbf{Y}} \|\mathbf{Z} - \mathbf{\Phi Y}\|_2^2 + c\|\mathbf{Y} - \mathbf{Y}_0\|_2^2, \tag{4.16}$$

to obtain the final reconstruction. As an alternative, the example-based procedure (*ExDict*) proposed in [2], the dictionaries $\mathbf{D}_{lo}$ and $\mathbf{D}_{hi}$ are directly fixed as the features extracted from low-resolution patches and vectorized high resolution patches respectively. Similar to the *PairDict* method, the global reconstruction constraint in (4.16) is imposed for the final reconstruction.

In our simulations, standard grayscale images (Table 4.2) are magnified by a factor of 2, using the proposed approaches. In addition to the *PairDict* and *ExDict* methods, simple bicubic interpolation is also used as a baseline method. We also

Original

Degraded

PairDict (27.78 dB)

ExDict (27.78 dB)

BoostKM (27.78 dB)

Figure 4.9: SISR of the *Cameraman* image with scaling factor of 2. The *PairDict*, *ExDict*, and *BoostKM* methods result in very similar high resolution images. PSNRs of resulting images are also shown.

obtained paired dictionaries with 1024 atoms using $100,000$ randomly chosen patches of size 5×5 from the grayscale natural images in the training set. The sparsity penalty used in training was $\lambda_{tr} = 0.15$. The training set was reduced to the size

of $20,000$ samples and used as the dictionary for the *ExDict method*. For ensemble learning, L was fixed at 50 and the approximation for each data sample was obtained using just a $1-$sparse representation.

For different number of training samples, we compared the training times for *PairDict* (40 iterations), *BoostEx* ($L = 50$) and *BoostKM* ($L = 50$) algorithms in Figure 4.8. The computation times reported in this chapter were obtained using a single core of a 2.8 GHz Intel i7 Linux machine with 8GB RAM. The *BoostKM* approach has the maximum computational complexity for training, followed by *PairDict* and *BoostEx* approaches. The *ExDict* procedure requires no training and for *RandExAv*, training time is just the time for randomly selecting K samples from the training set of T samples, for L rounds. Clearly, the complexity incurred for this is extremely low.

For the test images, SISR is performed using the baseline *PairDict* and *ExDict* approaches using a sparsity penalty of $\lambda_{te} = 0.2$. For the *PairDict*, and *ExDict* approaches, the code provided by the authors [126] was used to generate the results. The recovery performance of the proposed algorithms are reported in Table 4.2. For *PairDict*, as well the proposed ensemble methods, the dictionary size is fixed at 1024, whereas all the examples are used for training with the *ExDict* approach. We observed from our results that an ensemble representation with a simplified sparse coding scheme (1-sparse) matched the performance of the baseline methods (Figure 4.9).

4.5 Application: Unsupervised Clustering

Conventional clustering algorithms such as K-Means provide good clusterings only when the natural clusters of the data are distributed around a mean vector in space. For data that lie in a union of low-dimensional subspaces, it is beneficial to develop clustering algorithms that try to model the actual data distribution better. The

Table 4.2: Superresolution of standard images upscaled by a factor of 2: PSNR in dB obtained with bicubic interpolation (*Bicubic*), paired dictionary (*PairDict*) [1], example dictionary (*ExDict*) [2], BoostEx (*BEx*), BoostKM (*BKM*), and RandExAv (*RExAv*) methods.

Image	**Bicubic**	**PairDict**	**ExDict**	**BEx**	**BKM**	**RExAv**
Lena	34.10	**35.99**	**35.99**	35.97	35.95	**35.99**
Boat	29.94	**31.34**	**31.34**	31.28	31.23	31.29
House	32.77	**34.49**	**34.49**	34.38	34.41	34.41
Cameraman	26.33	**27.78**	**27.78**	27.72	**27.78**	27.71
Straw	24.20	25.93	25.93	25.90	25.90	**25.94**
Girl	33.81	**35.39**	**35.39**	35.33	35.37	35.35

sparse subspace clustering method [131], a special case of which is referred to as the ℓ_1 graph clustering [19], results in clusters that correspond to subspaces of data. This is achieved by representing each example as a sparse linear combination of the others and finally performing spectral clustering using a similarity matrix obtained from the coefficient matrix. The clustering method has the advantage of incorporating the noise model directly when performing sparse coding, thereby achieving robustness. The coefficient vector for the i^{th} data sample is obtained as

$$\min_{\mathbf{b}_i} \|\mathbf{x}_i - \mathbf{X}\mathbf{a}_i\| + \lambda\|\mathbf{a}_i\|_1, \text{ subj. to. } a_{ii} = 0. \tag{4.17}$$

By imposing the constraint that the i^{th} element of $\mathbf{a}_i$ should be 0, we ensure that a data sample is not represented by itself, which would have resulted in a trivial approximation. The coefficient matrix is denoted as $\mathbf{A} = [\mathbf{a}_1\mathbf{a}_2 \dots \mathbf{a}_T]$, and spectral clustering [105] is performed by setting the similarity matrix to the symmetric non-negative version of the coefficient matrix, $\mathbf{S} = |\mathbf{A}| + |\mathbf{A}^T|$. Computing the graph in this case necessitates the computation of sparse codes of T data samples with a $M \times (T-1)$ dictionary. Sparse coding-based graphs can also constructed based on coefficients obtained with a dictionary $\mathbf{D}$, inferred using the *Alt-Opt* procedure. Denoting the sparse codes for the examples $\mathbf{X}$ by the coefficient matrix $\mathbf{A} = [\mathbf{a}_1 \dots \mathbf{a}_T]$, the similarity matrix can be constructed as $\mathbf{S} = |\mathbf{A}^T\mathbf{A}|$. Similar to the ℓ_1 graphs,

Table 4.3: Comparison of the clustering performances (accuracy and normalized mutual information) of the algorithms with standard datasets. The best maximum or average performance is given in bold font.

Dataset	ℓ_1 **graph**		**BoostEx**		**RandExAv**		**BoostKM**	
	max	avg	max	avg	max	avg	max	avg
Accuracy								
Digits	**88.72**	74.61	88.63	76.79	88.62	**77.56**	88.31	76.85
Soybean	67.44	58.22	67.26	63.21	**70.82**	**65.31**	69.22	63.50
Segment	65.32	57.84	63.42	58.46	63.33	56.49	**65.63**	**58.67**
Satimage	77.56	69.37	75.03	72.88	**83.59**	**75.25**	71.62	65.32
USPS	78.24	62.47	75.01	68.18	78.26	64.14	**90.96**	**75.18**
NMI								
Digits	**84.97**	76.50	84.69	77.67	84.73	**78.04**	84.31	78.02
Soybean	74.55	65.94	72.84	68.66	**77.50**	**73.71**	76.05	71.66
Segment	**59.14**	53.91	56.72	53.23	54.94	51.20	58.44	**55.32**
Satimage	65.09	61.20	62.98	59.01	**69.38**	**66.12**	60.27	55.57
USPS	81.04	74.14	70.52	66.89	82.67	76.95	**82.78**	**78.37**

this similarity matrix can be used with spectral clustering to estimate the cluster memberships [117]. In this case, the dominant complexity in computing the graph is in learning the dictionary, and obtaining the sparse codes for each example. When the number of training examples is large, or when the data is high-dimensional, approaches that use sparse coding-based graphs incur high computational complexity.

We propose to construct sparse representation-based graphs using our ensemble approaches and employ them in spectral clustering. In our ensemble approaches, we have two example-based procedures, (*RandExAv* and *BoostEx*) and one that uses K-Means dictionaries (*BoostKM*). For *BoostKM*, we obtain L dictionaries of size K using the boosting procedure, with $1-$sparse approximations. The final coefficient vector of length LK for the data sample $\mathbf{x}_i$ is obtained as, $\mathbf{a}_i = [\mathbf{a}_{1,i}^T \mathbf{a}_{2,i}^T \dots \mathbf{a}_{L,i}^T]^T$, where $\mathbf{a}_{l,i}$ is the coefficient vector for round l. The similarity matrix is then estimated as $\mathbf{S} = |\mathbf{A}^T\mathbf{A}|$. In the example-based procedures, again $1-$sparse representation is used to obtain the coefficient vectors $\{\mathbf{a}_{1,i}, \mathbf{a}_{2,i} \dots, \mathbf{a}_{L,i}\}$, for a data sample $\mathbf{x}_i$. A

cumulative coefficient vector of length T can be obtained by recognizing that each coefficient in $\mathbf{a}_{l,i} \in \mathbb{R}^K$, can be associated to a particular example, since $\mathbf{D}_l$ is an example-based dictionary. Therefore a new $1-$sparse coefficient vector $\bar{\mathbf{a}}_{l,i} \in \mathbb{R}^T$ is created such that $\mathbf{D}_l \mathbf{a}_{l,i} = \bar{\mathbf{X}} \bar{\mathbf{a}}_{l,i}$, where $\bar{\mathbf{X}}$ contains the normalized set of data samples $\mathbf{X}$. Finally the cumulative coefficient vector for $\mathbf{x}_i$ is obtained as $\sum_{l=1}^{L} \beta_l \bar{\mathbf{a}}_{l,i}$. They are then stacked to form the coefficient matrix $\bar{\mathbf{A}} = [\bar{\mathbf{a}}_1 \dots \bar{\mathbf{a}}_T]$. Spectral clustering can be now performed using the similarity matrix $\mathbf{S} = |\bar{\mathbf{A}}| + |\bar{\mathbf{A}}^T|$. The clustering performance was evaluated in terms of accuracy and normalized mutual information (NMI), and compared with ℓ_1 graphs. As seen from Table 4.3, the ensemble-based approaches result in high accuracy as well as NMI. In all our simulations, data was preprocessed by centering and normalizing to unit norm. It was observed that the proposed ensemble methods incur comparable computational complexity to ℓ_1 graphs for datasets with small data dimensions. However, we observed significant complexity reduction with the *USPS* dataset, which contains 9298 samples of 256 dimensions. To cluster the *USPS* samples, the ℓ_1 graph approach took 411.85 seconds to compute the sparse codes, whereas *BoostEx*, *RandExAv*, and *BoostKM* took 152.56, 147.93, and 83.58 seconds respectively. This indicates the suitability of the proposed methods for high-dimensional, large scale data.

4.6 Conclusions

We proposed and analyzed the framework of ensemble sparse models, where the data is represented using a linear combination of approximations from multiple sparse representations. Theoretical results and experimental demonstrations show that an ensemble representation leads to a better approximation when compared to its individual constituents. Three different methods for learning the ensemble were proposed. Results in compressive recovery showed that the proposed approaches performed better than the baseline sparse coding method. Furthermore, the ensemble approach

performed comparably to several recent techniques in single image superresolution. Results with unsupervised clustering also showed that the proposed method leads to better clustering performance in comparison to the ℓ_1 graph method.

Chapter 5

COMBINING SPARSE CODING AND MANIFOLD PROJECTION FOR IMAGE RECOVERY

5.1 Introduction

Sparse approximation techniques can be used to recover data from its low dimensional corrupted observations, based on the knowledge that the data is sparsely representable using a known dictionary. Global dictionaries learned using patches from a set of natural images have been found to generalize well in obtaining sparse representations of a wide range of images, not included in the training set [12]. The process of dictionary learning can effectively identify the elementary representative patterns of a dataset. However, the efficiency of sparse coding based recovery schemes that use predefined or learned dictionaries can be improved if additional regularization is performed using the higher dimensional training examples from the data manifold. We propose two models that combine sparse coding using a predefined dictionary with projection onto the data manifold, to improve data recovery. The first model performs regularization of the sparse codes using examples from the data manifold and the second model directly combines sparse coding and manifold projection. Using an example application of image inpainting, we demonstrate that the proposed models achieve a reduction in reconstruction error in comparison to using only sparse coding with predefined and learned dictionaries, when the percentage of missing pixels and/or noise level is high. The second model was used in image inpainting and compressive recovery of standard images under various noise/undersampling conditions. In all cases of compressive recovery and most cases of image inpainting, the proposed model performed better than sparse coding based recovery schemes using predefined and learned dictionaries. A part of the methods and results presented in this chapter were published in [39].

5.2 Combined Sparse Coding and Manifold Projection

Using a dictionary $\mathbf{D} \in \mathbb{R}^{M \times K_d}$, the sparse representation of the test data $\mathbf{y} \in \mathbb{R}^M$ can be computed using the optimization program,

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \|\mathbf{y} - \mathbf{D}\boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_1. \tag{5.1}$$

The dictionary can be a collection of predefined basis functions, such as the DCT, or can be learned from the training set itself using algorithms such as the K-SVD [13] or multilevel dictionary learning [12]. With the additional knowledge that the test data lies near the manifold $\mathcal{M}$, we present two approaches to combine the paradigms of manifold projection and sparse coding in order to achieve a better recovery performance. The assumption of nearness to a manifold is true in many cases such as when the data are patches from natural images. The T training samples obtained from the underlying manifold $\mathcal{M}$ are indicated by $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^T$, where $\mathbf{x}_i \in \mathbb{R}^M$ and $\mathbf{X} \in \mathbb{R}^{M \times T}$. The first approach uses manifold samples for regularizing the sparse code computed using a predefined dictionary, whereas the second approach directly combined the sparse coding and manifold projection models.

5.2.1 Model 1: Regularizing Sparse Coding using Manifold Samples

The knowledge of the underlying manifold can be exploited by learning a dictionary directly from the manifold samples and computing a sparse code for the test data using the learned dictionary. However, in cases with missing/incomplete data, we have the observations

$$\mathbf{z} = \boldsymbol{\Psi}^T \mathbf{y} + \mathbf{n}, \tag{5.2}$$

where $\boldsymbol{\Psi} \in \mathbb{R}^{M \times N}$ with $N < M$ and $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_N)$. When $N << M$ and σ^2 is high, apart from sparsity constraints, additional regularization in the form of samples from the manifold will aid in the recovery of $\mathbf{y}$. The proposed model uses a predefined dictionary for sparse coding, and the training examples (manifold samples)

in the neighborhood for regularization. It is important to note that once the sparse code is generated, the proposed approach performs the reconstruction using only the predefined dictionary and we do not require the manifold samples. To summarize, the goal of the proposed approach is to find a sparse code using the observation, such that the recovered test sample lies close to the manifold $\mathcal{M}$.

The proposed model involves two components: (a) computing the sparse code using the observation, $\mathbf{y}$, and $\mathbf{D}$ and (b) projection of the recovered test sample $\mathbf{D}\boldsymbol{\beta}$ close to $\mathcal{M}$. This in essence involves combining the sparse representation and manifold projection problems given in (6.3) and (3.1) into the joint optimization problem,

$$\begin{aligned} \{\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}\} &= \underset{\boldsymbol{\alpha},\boldsymbol{\beta}}{\operatorname{argmin}} \|\mathbf{D}\boldsymbol{\beta} - \mathbf{X}\boldsymbol{\alpha}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1 \\ &\text{subject to } \|\mathbf{z} - \boldsymbol{\Psi}^T\mathbf{D}\boldsymbol{\beta}\|_2^2 \leq \epsilon, \sum_{i=1}^{T} \alpha_i = 1, \alpha_i \geq 0 \ \forall i. \end{aligned} \tag{5.3}$$

Here, $\boldsymbol{\beta}$ denotes the sparse code for the observation $\mathbf{y}$, $\boldsymbol{\alpha}$ corresponds to the coefficient vector obtained from manifold projection, and λ controls the trade-off between sparsity of $\boldsymbol{\beta}$ and the manifold projection residual. The first term in the objective function along with the second and third constraints perform the manifold projection on the recovered data $\mathbf{D}\boldsymbol{\beta}$. When compared to (3.1), the manifold projection component in the cost function of (5.3) does not contain the weighted sparsity term, $\sum_{i=1}^{T} \|\mathbf{y} - \mathbf{x}_i\|_2^2|\alpha_i|$. This omission does not affect the performance of the proposed algorithm and leads to a speed-up of the optimization, as observed from our experiments. ϵ is the maximum residual error allowed for the sparse representation.

From the term $\|\mathbf{z} - \boldsymbol{\Psi}^T\mathbf{D}\boldsymbol{\beta}\|_2^2 \leq \epsilon$, we can observe that the error ellipsoid for the sparse recovery is constrained only in N dimensions as $\boldsymbol{\Psi}^T \in \mathbb{R}^{N\times M}$. When N is not large enough to permit a good sparse approximation, constraining $\mathbf{D}\boldsymbol{\beta}$ to be close to the manifold $\mathcal{M}$ will improve the approximation. A simplified version of this scenario is illustrated in Figure 5.1. We have $M = 2$ and $N = 1$ in this case. The test data $\mathbf{y}$ is lying close to the manifold $\mathcal{M}$. The noisy observation of $\mathbf{y}$ with one

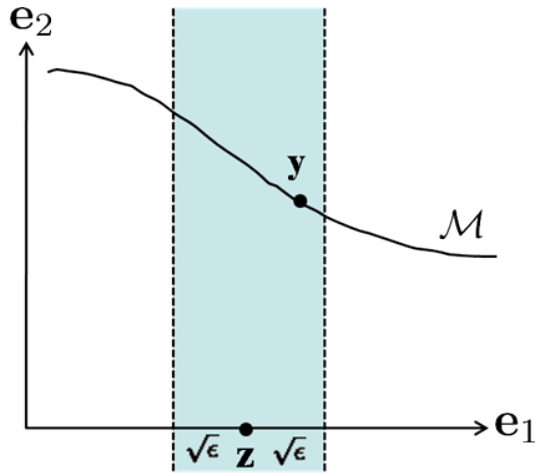


Figure 5.1: The data $\mathbf{y}$ lying near manifold $\mathcal{M}$ is corrupted with noise and one of the dimensions is removed to result in $\mathbf{y}$. The shaded region indicates the possible locations of recovered data, subject to error constraints.

of the dimensions removed is $\mathbf{z}$. $\mathbf{e}_1$ and $\mathbf{e}_2$ are the canonical basis vectors. The first constraint in (5.3) specifies that the projection of the recovered data on to $\mathbf{e}_1$ should lie within $\sqrt{\epsilon}$ of $\mathbf{z}$. The shaded region indicates the possible locations of recovered data. Incorporating the constraint that the recovered data $\mathbf{D}\boldsymbol{\beta}$ is close to $\mathcal{M}$ improves the chance of $\mathbf{D}\boldsymbol{\beta}$ being close to $\mathbf{y}$. This is particularly true when N is quite small because of which sparse coding alone cannot provide a good recovery for the test data from its noisy observation. Eqn. (5.3) is rewritten as

$$\begin{aligned}\hat{\boldsymbol{\eta}} = \underset{\boldsymbol{\eta}}{\operatorname{argmin}} \left\| [\mathbf{D} \quad -\mathbf{X}]\, \boldsymbol{\eta} \right\|_2^2 + \lambda \left\| [\mathbf{I} \quad \mathbf{0}]\, \boldsymbol{\eta} \right\|_1 \\ \text{subject to } \left\| [\mathbf{I} \quad \mathbf{0}]\, \bar{\mathbf{y}} - \left[\boldsymbol{\Psi}^T\mathbf{D} \quad \mathbf{0}\right] \boldsymbol{\eta} \right\|_2^2 \leq \epsilon, \\ [\mathbf{0} \quad -\mathbf{I}]\, \boldsymbol{\eta} \leq 0 \text{ and } [\mathbf{0} \quad \mathbf{I}]\, \boldsymbol{\eta} = 1. \end{aligned} \tag{5.4}$$

and solved using the MATLAB CVX [132] package. Here $\bar{\mathbf{y}} = [\mathbf{z}^T \ \mathbf{z}^T]^T$, $\boldsymbol{\eta} = [\boldsymbol{\beta}^T \boldsymbol{\alpha}^T]^T$, $\mathbf{0}$ indicates a zero vector or matrix and $\mathbf{I}$ is an identity matrix of appropriate dimensions.

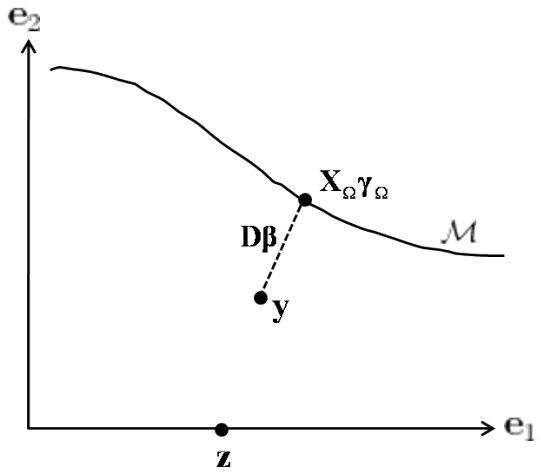


Figure 5.2: The test data $\mathbf{y}$ has a component that can be represented in the manifold $\mathcal{M}$ and another component that can be sparsely represented in a dictionary $\mathbf{D}$. The test observation $\mathbf{z}$ is a noisy, low dimensional version of the test data $\mathbf{y}$.

5.2.2 Model 2: Directly Combining Manifold Projection and Sparse Coding

Instead of regularizing the sparse code using manifold samples, we can also assume that the test data $\mathbf{y}$ has two components, one that can be represented by manifold projection on to the training examples $\mathbf{X}$ and the other that can be well-represented using the pre-defined dictionary $\mathbf{D}$. This model can be expressed as

$$\mathbf{y} = \mathbf{X}_\Omega \boldsymbol{\alpha}_\Omega + \mathbf{D}\boldsymbol{\beta}, \tag{5.5}$$

where Ω is the neighborhood of $\mathbf{y}$ assumed to be known. The only additional constraint posed here is $\boldsymbol{\alpha}_\Omega \geq 0$ since our experiments showed that the additional constraint that the elements of $\boldsymbol{\alpha}_\Omega$ sum to one did not yield a significant improvement in performance.

The test observation $\mathbf{z}$, which is a noisy and low-dimensional version of $\mathbf{y}$ is given by (5.2) and this model is illustrated in Figure 5.2 for a low-dimensional case when $M = 2$ and $N = 1$. Since the neighborhood Ω is usually not known, an estimated neighborhood $\hat{\Omega}$ is obtained by computing the distance between $\mathbf{z}$ and the low-dimensional examples $\boldsymbol{\Psi}^T\mathbf{X}$. The coefficients can be computed by solving the

optimization program

$$\{\hat{\boldsymbol{\alpha}}_{\hat{\Omega}}, \hat{\boldsymbol{\beta}}\} = \underset{\boldsymbol{\alpha}_{\hat{\Omega}}, \boldsymbol{\beta}}{\operatorname{argmin}} \|\boldsymbol{\alpha}_{\hat{\Omega}}\|_1 + \|\boldsymbol{\beta}\|_1 \text{ subject to } \|\mathbf{z} - \boldsymbol{\Psi}^T\mathbf{X}\boldsymbol{\alpha}_{\hat{\Omega}} - \boldsymbol{\Psi}^T\mathbf{D}\boldsymbol{\beta}\|_2^2 \leq \epsilon, \boldsymbol{\alpha}_{\hat{\Omega}} \geq \mathbf{0}. \tag{5.6}$$

The above program can be solved using the COMB-BP algorithm or its greedy variant, the COMB-OMP, proposed in Section 6.3.3. In both cases, we need to modify the algorithms to incorporate stopping criteria based on the error constraint. The COMB-BP can be efficiently implemented by modifying the LARS algorithm [101] to incorporate the non-negative constraint on a part of the coefficient vector.

5.3 Applications

Image inpainting and compressive recovery are the example applications that we consider for the frameworks we discussed above. Considering the observation model in (5.2), we present two applications. When the matrix $\boldsymbol{\Psi}$ is a subset of the columns of an identity matrix, the model describes a missing data scenario and $\mathbf{y}$ can be recovered from $\mathbf{z}$ by inpainting. When $\boldsymbol{\Psi}$ is a matrix with entries realized from certain probability distributions, then (5.2) denotes noisy measurements of the data computed through random projections and compressive recovery can be performed. Considering that the data is generated from the basic linear generative model $\mathbf{y} = \mathbf{D}\boldsymbol{\beta}$, where $\boldsymbol{\beta}$ is sparse, recovery of the test data can be performed by minimizing the ℓ_0 norm as

$$\min_{\boldsymbol{\beta}} \|\boldsymbol{\beta}\|_0 \text{ subject to } \|\mathbf{z} - \boldsymbol{\Psi}^T\mathbf{D}\boldsymbol{\beta}\|_2 < \epsilon \tag{5.7}$$

where ϵ is the error goal that depends on the noise level σ of $\mathbf{n}$. In order to solve (5.7) as a convex program, we replace ℓ_0 norm by its convex surrogate, the ℓ_1 norm.

In image inpainting, the missing pixels can be directly estimated by computing $\boldsymbol{\beta}$ using (5.7) and then reconstructing the test data from the sparse code. Since the dimensionality of the dictionary $\mathbf{D}$ is reduced when performing recovery, exact recovery is possible only if the number of non-zero coefficients in $\boldsymbol{\beta}$ satisfies the

coherence condition for the dictionary $\mathbf{\Psi}^T\mathbf{D}$ [11]. Some recent inpainting algorithms that use sparse representations are available in [9, 10, 133, 134].

The fundamental idea behind compressive sensing is that for most types of naturally occurring signals, only a few measurements are sufficient to accurately recover them [67,130,135]. When the signal $\mathbf{y}$ lies in a union of subspaces whose basis vectors are defined by the columns of an overcomplete dictionary $\mathbf{D}$, it can be represented as $\mathbf{y} = \mathbf{D}\boldsymbol{\beta}$, where $\boldsymbol{\beta}$ is a sparse vector. The measurement system $\mathbf{\Psi}$ can be random or optimized to the dictionary $\mathbf{D}$. When the measurement system is random, $\mathbf{\Psi}$ typically consists of i.i.d. samples drawn from distributions such as Gaussian, Bernoulli, Rademacher or random ternary [67]. They are universal in the sense that they can work well with a large class of dictionaries. When optimized to the dictionary $\mathbf{D}$, the measurement system leads to an improved recovery when compared to a randomly constructed $\mathbf{\Psi}$ [12, 136, 137]. The theoretical recovery performance of compressive sensing depends on the characteristics of $\mathbf{\Psi}^T\mathbf{D}$, which we refer to as the equivalent dictionary. More information on the theory of compressive sensing and recovery can be found in [138–143]. In the experiments below we recover the test data from the low-dimensional corrupted observations using the models proposed in this chapter.

5.4 Experiments

The experiments demonstrate the use of the proposed models in inpainting and compressive recovery of natural images. We resort to a patch-based approach and our training and test data are image patches of size 8×8. The pre-defined dictionary $\mathbf{D}$ used in our models is overcomplete DCT of size 64×256. The manifold samples consisted of 50000 randomly chosen patches of size 8×8 from 250 training images of the Berkeley Segmentation Dataset (BSDS) [127]. Some examples of training images are shown in Figure 5.3. For comparison, a KSVD dictionary of size 64×256 was also learned from the 50000 training patches.

(a) (b) (c)

Figure 5.3: Example training images from BSDS from which the training patches were extracted.

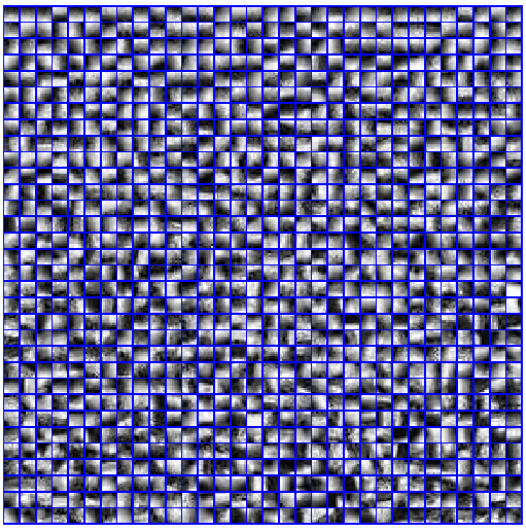


Figure 5.4: The first 1024 of the 10704 density filtered patches. Density filtration itself was performed on a set of 50000 examples chosen from the BSDS training set.

The training samples were preprocessed using density filtration in three stages. The procedure identifies training examples that constitute high density regions in space and these were stacked in the matrix $\mathbf{X}$ for use with the proposed models. In the first stage of preprocessing, for each sample (64−dimensional), the mean value of its coordinates was subtracted from each of its coordinate. The second step involved computing the energy, as the sum of squares of coordinates, for each sample. The mean energy of all samples was computed and only those samples with energy greater than 95% of the mean energy were retained. The third stage was density filtration [80] in which we specified the number of nearest neighbors (15 here) and picked a

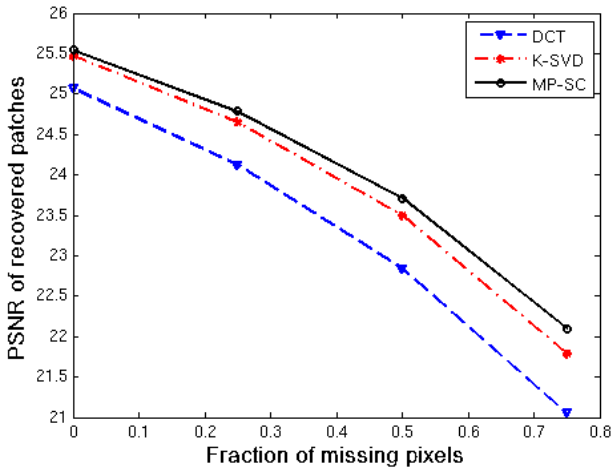


Figure 5.5: Average inpainting performance of the proposed Model 2 on 10,000 randomly chosen BSDS test patches at $\sigma = 25$.

Table 5.1: Inpainting performance for standard images when the fraction of missing pixels is 0.75 ($N = 16$) and 0.5 ($N = 32$). For each noise level (σ), the first row is the PSNR with the overcomplete DCT, second row is with the K-SVD dictionary, third row is with the proposed combined model.

Noise	Boat		House		Lena		Peppers		Man	
	N=16	**N=32**	**N=16**	**N=32**	**N=16**	**N=32**	**N=16**	**N=32**	**N = 16**	**N=32**
$\sigma = 0$	26.31	**31.66**	28.57	34.23	29.21	34.56	23.86	29.84	27.23	31.68
	27.09	31.52	29.4	**34.55**	30.17	**34.86**	25.29	30.43	28.21	**32.29**
	27.45	31.49	**30.19**	34.54	**30.6**	34.69	**26.2**	**30.96**	**28.52**	32.23
$\sigma = 15$	25.24	28.22	26.93	30.51	27.48	30.59	23.29	27.3	25.89	28.36
	25.86	28.55	27.42	30.76	28.03	30.87	24.47	28.06	26.57	28.88
	26.18	**28.6**	**28.09**	**30.98**	**28.36**	**30.91**	**25.3**	**28.55**	**26.83**	**28.92**
$\sigma = 25$	24.27	26.57	25.92	28.62	26.18	28.66	22.71	25.72	24.78	26.7
	24.78	26.89	26.31	28.87	26.58	28.95	23.74	26.42	25.28	**27.15**
	24.99	**26.95**	**26.8**	**29.02**	**26.76**	**28.95**	**24.58**	**26.88**	**25.44**	27.13

percentage of the samples (30% here) that had their neighbors closest to them in space. $\mathbf{X}$ contained 10704 samples after the preprocessing steps. The parameters for preprocessing were chosen in order to ensure that sufficient number of representative training samples were available, but the number was not so high that it would slow down the optimization. The first 1024 patches out of 10704 density filtered examples are shown in Figure 5.4. It can be seen that most of the patches that constitute high density regions in space have a "blurred-wedge" like structure.

Before performing extensive tests with standard images, we randomly selected $10,000$ patches of size 8×8 from the BSDS test set and performed inpainting in order to analyze the behavior of the proposed models. The test data were $10,000$ patches chosen at random from the BSDS test dataset. From each patch either 25%, 50% or 75% of the pixels were masked and AWGN with standard deviation $\sigma = \{15, 20, 25\}$ was added. Only the noisy, unmasked pixels were retained in $\mathbf{z}$. The number of unmasked pixels was N and $\mathbf{\Psi}^T$ would be an identity matrix with N of its randomly chosen rows retained. Inpainting based on sparse coding consisted of solving (6.3) using $\mathbf{z}$ and $\mathbf{\Psi}^T\mathbf{D}$, as the observation and the dictionary respectively. Inpainting using Model 1 was performed by solving (5.3) for each observation $\mathbf{z}$ using $\mathbf{X}$ and $\mathbf{\Psi}^T\mathbf{D}$ as manifold examples and the dictionary respectively. Inpainting using Model 2 amounts to finding $K = 500$ nearest neighbors indexed by the set $\hat{\Omega}$ using $\mathbf{z}$ and $\mathbf{\Psi}^T\mathbf{X}$, and using (5.6) using a modified LARS algorithm to solve for the coefficients. The residual error goal was computed as $(1.1\sigma)^2 N$ in all the three cases. For sparse coding and Model 1 proposed approach, the image patches were recovered as $\mathbf{D}\hat{\boldsymbol{\beta}}$ using their respective sparse codes. For Model 2, the recovered image patch was given by $\mathbf{X}_{\hat{\Omega}}\hat{\boldsymbol{\alpha}}_{\hat{\Omega}} + \mathbf{D}\hat{\boldsymbol{\beta}}$. The root mean-squared error (RMSE) of the inpainting procedure for all the P test patches is,

$$RMSE = \left[\frac{1}{MP}\sum_{i=1}^{P} \|\mathbf{y}_i - \mathbf{D}\boldsymbol{\beta}_i\|_2^2\right]^{\frac{1}{2}}. \tag{5.8}$$

The PSNR is computed as $20\log_{10}(255/RMSE)$. The performance of the Model 2 is shown only for the noise level $\sigma = 25$, where it shows significant performance improvement over predefined/learned dictionaries. Model 1 performs similar to Model 2 in this cases, but its computational complexity is very high.

5.4.1 Inpainting Standard Images

We performed inpainting on a set of standard images (*lena*, *boat*, *peppers*, *house*, *couple* and *man*) with 50% and 75% missing pixels and additive Gaussian noise of $\sigma =$

(a) (b) (c)

Figure 5.6: Inpainting *Peppers* image when the fraction of missing pixels is 0.75 ($N = 16$) and noise level $\sigma = 15$ using: (a) overcomplete DCT dictionary, PSNR = 23.39 dB, (b) KSVD dictionary, PSNR = 24.47 dB, (c) Proposed combined model, PSNR = 25.3 dB.

$\{0, 15, 25\}$. We only present the result of Model 2 here as its performance supersedes that of Model 1. The COMB-OMP algorithm was used to implement a greedy version of (5.6), with $K = 500$. Inpainting was performed on patches with $1-$pixel overlap and the image was recovered by averaging the patches. Table 5.1 presents the PSNR of the recovered images when 75% and 50% of the pixels were missing. It can be seen that in all cases where $N = 16$ and $\sigma = 25$, the proposed model performs better than sparse coding based on DCT/KSVD dictionaries. In general, it can be seen that as the noise level and number of missing pixels increase the performance of the proposed model improves over the other methods. Inpainted *Peppers* images with 75% missing pixels and $\sigma = 15$ are shown in Figure 5.6. The inpainted image using the proposed model has the details clearly visible when compared to the other two images.

5.4.2 Compressive Recovery of Standard Images

l Compressive sensing was performed on non-overlapping 8×8 image patches of the standard images (*Barbara*, *Boat*, *House*, *Lena* and *Man*) using Gaussian random measurement matrices with the number of measurements fixed at 16, 32 for various measurement SNRs of 0, 15 and 25 dB. The COMB-OMP algorithm was again used to

(a) (b) (c)

Figure 5.7: Recovery of *Boat* image from compressed measurements with $N = 16$ at 15 dB SNR: (a) overcomplete DCT dictionary, PSNR = 21.28 dB, (b) KSVD dictionary, PSNR = 23.97 dB, (c) Proposed combined model, PSNR = 26.21 dB

Table 5.2: Compressive recovery performance for standard images at various measurement SNRs when $N = 16$ and $N = 32$. For each SNR, the first row is the PSNR with the overcomplete DCT, second row is for K-SVD, third row is for MLD and the fourth row is for the proposed combined model.

SNR (dB)	Barbara		Boat		House		Lena		Man	
	N=16	**N=32**	**N=16**	**N=32**	**N=16**	**N=32**	**N=16**	**N=32**	**N=16**	**N=32**
0 dB	19.87	21.29	20.76	22.33	21.70	23.76	22.40	24.26	21.44	22.95
	20.43	21.44	22.02	23.38	23.25	24.86	23.83	25.40	22.86	24.22
	20.63	21.38	22.58	23.76	24.03	25.50	24.74	25.76	23.43	24.53
	20.87	**21.77**	**22.81**	**24.20**	**24.43**	**25.98**	**24.91**	**26.45**	**23.73**	**25.04**
15 dB	21.37	24.94	22.46	26.00	24.05	28.21	24.20	28.34	23.24	26.18
	21.92	24.33	24.39	27.66	26.43	30.03	26.67	30.35	25.40	28.35
	22.37	24.26	25.43	27.29	27.63	30.49	28.20	30.24	26.23	28.05
	22.85	**24.88**	**25.94**	**28.56**	**28.60**	**31.54**	**28.69**	**31.45**	**26.87**	**29.33**
25 dB	21.72	25.23	22.60	26.33	24.03	28.93	24.40	28.75	22.87	26.76
	22.10	24.76	24.63	28.53	26.83	31.74	26.90	31.59	25.42	29.31
	22.54	24.78	25.74	27.86	28.07	32.06	28.88	31.10	26.57	28.79
	23.11	**25.30**	**26.29**	**29.24**	**29.19**	**32.95**	**29.21**	**32.49**	**27.22**	**30.05**

implement (5.6) with $K = 500$, similar to the case of inpainting. The average recovery performance over 25 iterations using DCT, KSVD, MLD [12] and the proposed Model 2 are presented in Table 5.2. It can be seen that the proposed Model outperforms all the others in all the cases presented. Figure 5.7 also shows the recovered *Boat* image for the case when $N = 16$ and SNR= 15dB when DCT/KSVD dictionaries and Model 2 was used. Difference in performance is visible particularly when we look at

the edges (such as the mast) in the image. Since edges and corners form a significant component of human visual perception, subjective improvement is also significant.

5.5 Conclusions

We presented an two models that combine the standard sparse coding paradigm with manifold projection using training examples, for improving the recovery performance for missing/incomplete data cases. The first model combines a manifold regularization with sparse coding, whereas the second model assumes that the data has a sparse coding component and a manifold projection component. We demonstrated that, these two models performed better recovery when compared to using plain sparse coding based methods. As demonstrated in the experiments, the proposed models are particularly useful in conditions of severe corruption of images.

Chapter 6

COMBINED NON-NEGATIVE AND GENERAL REPRESENTATIONS

6.1 Introduction

The non-negative solution to an underdetermined linear system can be uniquely recovered sometimes, even without imposing any additional sparsity constraints. In this chapter, we derive conditions under which a unique non-negative solution for such a system can exist, based on the theory of polytopes. Furthermore, we develop the paradigm of combined sparse representations, where only a part of the coefficient vector is constrained to be non-negative, and the rest is unconstrained (general). We analyze the recovery of the unique, sparsest solution, for combined representations, under three different cases of coefficient support knowledge. Experiments that demonstrate the recovery of combined representations using randomly generated dictionaries and coefficients are also presented.

The system of linear equations with the constraint that the solution is non-negative can be expressed as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\alpha}, \text{ where } \boldsymbol{\alpha} \geq 0, \tag{6.1}$$

where $\mathbf{y} \in \mathbb{R}^M$ is the data vector, $\boldsymbol{\alpha} \in \mathbb{R}^{K_x}$ is the non-negative solution (coefficient vector), and $\mathbf{X} \in \mathbb{R}^{M \times K_x}$ is the dictionary with $K_x > M$. When only a part of the solution is constrained to be non-negative and the rest is unconstrained (general), we obtain the combined representation model,

$$\mathbf{y} = \mathbf{X}\boldsymbol{\alpha} + \mathbf{D}\boldsymbol{\beta}, \text{ where } \boldsymbol{\alpha} \geq 0. \tag{6.2}$$

Here, the coefficient vector $\boldsymbol{\beta} \in \mathbb{R}^{K_x}$ is unconstrained, and $\mathbf{X} \in \mathbb{R}^{M \times K_x}$ and $\mathbf{D} \in \mathbb{R}^{M \times K_d}$ are the sub-dictionaries for the non-negative and general representations respectively. We denote the combined coefficient vector as $\boldsymbol{\delta} = [\boldsymbol{\alpha}^T \ \boldsymbol{\beta}^T]^T$, and the combined dictionary as $\mathbf{G} = [\mathbf{X} \quad \mathbf{D}]$. We assume that $\mathbf{G}$ is overcomplete with

$K_x + K_d > M$, and the columns of the dictionaries are normalized to have unit ℓ_2 norm. The sparsest solutions to (6.1) and (6.2) are obtained by minimizing the ℓ_0 norm, the number of non-zero elements, of the corresponding non-negative coefficient vector, $\boldsymbol{\alpha}$, or the combined coefficient vector, $\boldsymbol{\delta}$. In both the cases, the unique minimum ℓ_0 norm solution, when it exists, will be referred to as ML_0 solution. In this chapter, we focus on obtaining deterministic guarantees for recovery of the ML_0 solutions to the linear systems (6.1) and (6.2), using both convex and greedy algorithms, based on the properties of the dictionaries.

Some of the applications of the non-negative representation model in (6.1), and the combined model in (6.2) are in image inpainting [39], automatic speech recognition using exemplars [40], protein mass spectrometry [41], astronomical imaging [144], spectroscopy [145], source separation [42], and clustering/semi-supervised learning of data [19, 43], to name a few.

6.1.1 Prior Work

For the non-negative representation model in (6.1), a sufficiently sparse ML_0 solution can be recovered by minimizing the ℓ_1 norm of $\boldsymbol{\alpha}$, using the non-negative version of the basis pursuit (BP) algorithm [110], which we refer to as NN-BP. The optimization program can be expressed as

$$\min_{\boldsymbol{\alpha}} \mathbf{1}^T \boldsymbol{\alpha} \text{ subject to } \mathbf{y} = \mathbf{X}\boldsymbol{\alpha}, \boldsymbol{\alpha} \geq \mathbf{0}. \tag{6.3}$$

The conditions on $\mathbf{X}$ under which the recovery of ML_0 solution using (6.3) is possible have been derived based on the neighborliness of polytopes [35–37], and the non-negative null-space property [38]. A non-negative version of the greedy orthogonal matching pursuit (OMP) algorithm [146], which we will refer to as NN-OMP, for recovering the coefficients has also been proposed [34]. If the set

$$\{\boldsymbol{\alpha} | \mathbf{y} = \mathbf{X}\boldsymbol{\alpha}, \boldsymbol{\alpha} \geq 0\} \tag{6.4}$$

contains only one solution, we can use any variational function instead of the ℓ_1 norm in order to obtain the unique non-negative solution [34–36]. In particular, the solution can be obtained by using the non-negative least squares (NNLS) algorithm [41, 147].

A major part of our work investigates the combined sparse representation model introduced in (6.2), where only a part of the sparse coefficient vector is constrained to be non-negative. We consider the *deterministic sparsity thresholds* i.e., the maximum number of non-zero coefficients possible in the ML_0 solution, such that the ML_0 solution can be uniquely recovered. To the best of our knowledge, such an investigation has not been reported so far in the literature. However, when both $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are unconstrained general sparse vectors, the sparsity thresholds for recovery of the ML_0 solution have been presented in [62, 63]. By considering the coherence parameters of $\mathbf{X}$ and $\mathbf{D}$ separately, the authors in [62] show that an improvement up to a factor of two can be achieved in the deterministic sparsity threshold when compared to considering $\mathbf{X}$ and $\mathbf{D}$ together as a single dictionary. Note that deterministic sparsity thresholds provide guarantees that hold for all sparsity patterns and non-zero values in the coefficient vectors. *Probabilistic* or *robust* sparsity thresholds, that hold for most sparsity patterns and non-zero values in the coefficient vectors have also been derived in [62], again for the case where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are general sparse vectors. When this representation is approximately sparse and corrupted by additive noise, theory and algorithms for coefficient recovery are presented in [148].

6.1.2 Contributions

We present deterministic recovery guarantees for both the non-negative and the combined sparse representation models given by (6.1) and (6.2) respectively. Furthermore, we propose a greedy algorithm for performing coefficient recovery in combined representations and derive deterministic sparsity thresholds for unique recovery using ℓ_1 minimization and the proposed greedy algorithm.

For the non-negative model in (6.1), we derive the sufficient conditions for (6.4) to be singleton based on the neighborliness properties of the quotient polytope corresponding to the dictionary $\mathbf{X}$. Similar analyses reported in [35, 36] assume that the dictionary $\mathbf{X}$ is obtained from a random ensemble and append a row of ones to it, such that the row span of $\mathbf{X}$ contains the vector $\mathbf{1}^T$. In contrast, we do not assume any randomness on $\mathbf{X}$ and only require that its row span intersects the positive orthant. We show that the sparsity threshold on $\boldsymbol{\alpha}$, for the set (6.4) to be singleton, is the same as the deterministic sparsity threshold for recovering the ML_0 solution of a general sparse representation. Whenever this threshold is satisfied, ℓ_1-norm regularization in (6.3) can be replaced with any variational function. Section 6.2 presents the analysis of the non-negative representation model.

For the combined model in (6.2), we propose a variant of the greedy OMP algorithm, the combined OMP (COMB-OMP) algorithm, for performing coefficient recovery. We also consider a ℓ_1 regularized convex algorithm, which we refer to as combined BP (COMB-BP). We derive the deterministic sparsity thresholds for recovering the ML_0 solution using both the COMB-BP and COMB-OMP algorithms. We show that a factor-of-two improvement in the sparsity threshold, observed when $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are general sparse vectors [62], holds for recovery using the COMB-BP also. We also show that such an improvement in the sparsity threshold cannot be observed using the COMB-OMP algorithm, because of the partial non-negativity constraint on the coefficient vector. However, COMB-OMP incurs very low computational complexity when compared to COMB-BP. Furthermore, we obtain the sparsity thresholds in the following cases of coefficient support knowledge: (a) the non-zero support of both $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ are known, and (b) non-zero support of $\boldsymbol{\beta}$ alone is known. When analyzing case (b), we factor out the contribution of the general representation component and arrive at conditions under which ℓ_1-norm regularization in the resulting optimization can be replaced with any variational function for the recovery of $\boldsymbol{\alpha}$ (Section 6.3).

The performance of the COMB-BP and the COMB-OMP algorithms are also analyzed using simulations. The dictionary $\mathbf{G}$ is obtained from a Gaussian ensemble and the non-zero coefficients are obtained either from uniform distribution or fixed as random signs (± 1). It is shown that both COMB-BP and COMB-OMP respectively perform better than their unconstrained counterparts, the BP and the OMP, particularly as the K_x becomes larger. We also show that the COMB-OMP incurs substantially less computational complexity when compared to the COMB-BP.

6.2 Non-negative Sparse Representations

For the non-negative representation given in (6.1), we denote the number of non-zero coefficients in $\boldsymbol{\alpha}$ as S_x.

Definition ([64]) The two-sided coherence (or simply coherence) of the dictionary $\mathbf{X}$ is

$$\mu_x = \max_{i \neq j} \frac{|\mathbf{x}_i^T \mathbf{x}_j|}{\|\mathbf{x}_i\|_2 \|\mathbf{x}_j\|_2}, \tag{6.5}$$

Definition ([34]) The one-sided coherence of the dictionary $\mathbf{X}$ is

$$\sigma_x = \max_{i \neq j} \frac{|\mathbf{x}_i^T \mathbf{x}_j|}{\|\mathbf{x}_i\|_2^2}. \tag{6.6}$$

If the columns of $\mathbf{X}$ are normalized, we have $\mu_x = \sigma_x$, and if they had different ℓ_2 norms, we would have $\mu_x \leq \sigma_x$ [34, Lemma 1].

Definition ([34]) The dictionary $\mathbf{X}$ belongs to the class of matrices denoted as $\mathcal{M}^+$, if its row span intersects the positive orthant.

If $\mathbf{X} \in \mathcal{M}^+$, $\exists \mathbf{h}$ such that $\mathbf{h}^T \mathbf{X} = \mathbf{w}^T, \mathbf{w} > \mathbf{0}$. Let us define $\mathbf{W} = diag(\mathbf{w})$, $\mathbf{U} = \mathbf{X}\mathbf{W}^{-1}$, and denote σ_u and μ_u as the one-sided and two-sided coherences of $\mathbf{U}$ respectively. In [36, Theorem 1] it is shown that $\mathbf{X} \in \mathcal{M}^+$ is a necessary condition for (6.4) to be singleton. The main result in [34, Theorem 2] states that the set (6.4) will be singleton if $\mathbf{X} \in \mathcal{M}^+$ and $S_x < 0.5(1 + 1/\sigma_u)$.

We will now state the main result of this section, whose proof will be relegated to the end of this section.

Theorem 6.2.1 *When* $\mathbf{X} \in \mathcal{M}^+$*, the set defined in (6.4) is singleton if the number of non-zero entries in* $\boldsymbol{\alpha}$*,* $S_x < 0.5(1 + 1/\mu_x)$*.*

The threshold given in the above theorem is better than that of [34, Theorem 2], because $\mu_x = \mu_u$, $\mu_u \leq \sigma_u$ and hence $\mu_x \leq \sigma_u$. We are able to improve the threshold by resorting to geometric arguments based on the theory of polytopes. The rest of this section will state and prove lemmas that will be used in the proof of our main result.

We will define three geometric entities, the cross-polytope $\mathcal{C}^{K_x}$, the simplex $\mathcal{T}^{K_x-1}$ and the positive orthant $\mathbb{R}_+^{K_x}$, that will be used in the proof. The cross-polytope is defined as the ℓ_1 ball, $\|\boldsymbol{\alpha}\|_1 \leq 1$, in $\mathbb{R}^{K_x}$, and $\mathcal{T}^{K_x-1}$ is the standard simplex, the convex hull of unit basis vectors. Any general sparse representation with S_x non-zero coefficients can be successfully recovered using ℓ_1 minimization (BP), if the quotient polytope $\mathbf{X}\mathcal{C}^{K_x}$ is *centrally* S_x-*neighborly* [68, Theorem 1]. This form of neighborliness implies that any set of S_x vertices of $\mathbf{X}\mathcal{C}^{K_x}$, not including an antipodal pair (pair of $\pm\mathbf{x}_i$), span a face. For unique recovery of non-negative S_x-sparse vectors using the linear program given in (6.3), the condition on the quotient polytope $\mathbf{X}\mathcal{T}$ is that it must be *outwardly* S_x*-neighborly* [37, Theorem 1]. Here, we fix $\mathcal{T} = \mathcal{T}^{K_x-1}$ if $\mathbf{0}$ can be expressed as a convex combination of the columns of $\mathbf{X}$, else we fix $\mathcal{T} = \mathcal{T}_{\mathbf{0}}^{K_x}$ where $\mathcal{T}_{\mathbf{0}}^{K_x}$ is the solid simplex, the convex hull of $\mathcal{T}^{K_x-1}$ and the origin. When every set of S_x vertices, not including the origin, span a face, the quotient polytope is said to be outwardly S_x-neighborly.

Lemma 6.2.2 *When* $\mathbf{X} \in \mathcal{M}^+$ *and the number of non-zero coefficients in* $\boldsymbol{\alpha}$ *is* S_x*, the set defined in (6.4) is singleton if the quotient polytope* $\mathbf{X}\mathcal{T}_{\mathbf{0}}^{K_x}$ *is outwardly* S_x-*neighborly.*

Proof By assumption, $\exists \mathbf{h}$ such that $\mathbf{h}^T\mathbf{X} = \mathbf{w}^T$, $\mathbf{w} > \mathbf{0}$. Consider the quotient polytope $\mathbf{U}\mathcal{T}_{\mathbf{0}}^{K_x}$, where $\mathbf{U} = \mathbf{X}\mathbf{W}^{-1}$ and $\mathbf{W} = diag(\mathbf{w})$. Since $\mathbf{X}\mathcal{T}_{\mathbf{0}}^{K_x}$ is outwardly S_x-neighborly and the positive scaling of vertices does not affect the neighborliness of a polytope, $\mathbf{U}\mathcal{T}_{\mathbf{0}}^{K_x}$ is also outwardly S_x-neighborly. Now denote $\hat{\mathbf{y}} = \mathbf{y}/(\mathbf{h}^T\mathbf{y})$ and $\boldsymbol{\gamma} = \mathbf{W}\boldsymbol{\alpha}/(\mathbf{h}^T\hat{\mathbf{y}})$. The set defined in (6.4) has a one-to-one correspondence with

$$\{\boldsymbol{\gamma} | \hat{\mathbf{y}} = \mathbf{U}\boldsymbol{\gamma}, \boldsymbol{\gamma} \geq 0\}, \tag{6.7}$$

If we show that (6.7) is singleton, then (6.4) is singleton as well.

Since we know that $\|\boldsymbol{\alpha}\|_0 = S_x$, this implies that $\|\boldsymbol{\gamma}\|_0 = S_x$. Because of the neighborliness of the quotient polytope $\mathbf{U}\mathcal{T}_{\mathbf{0}}^{K_x}$, $\hat{\mathbf{y}}$ lies in its simplicial face F of affine dimension S_x. Denote $\mathcal{V}$ to be the set of vertices of F. The remaining vertices in the quotient polytope are denoted by the set $\mathcal{V}^c$. Consider an arbitrary vector $\hat{\mathbf{y}}^c$ expressed as a convex combination of the vertices $\mathcal{V}^c$. Since F is a face, there exists a linear functional λ_F and a constant c such that $\lambda_F^T\hat{\mathbf{y}} = c$ and $\lambda_F^T\hat{\mathbf{y}}^c < c$ [68]. This means that for an arbitrarily chosen $\hat{\mathbf{y}}$ and $\hat{\mathbf{y}}^c$ which are convex combinations of vertices $\mathcal{V}$ and $\mathcal{V}^c$ respectively, $\|\hat{\mathbf{y}} - \hat{\mathbf{y}^c}\|_2 > 0$. By extension, the rays in the directions of $\hat{\mathbf{y}}$ and $\hat{\mathbf{y}^c}$ intersect only at the origin. The convex cones formed by the vertices $\mathcal{V}$ and $\mathcal{V}^c$ are denoted as $\mathbf{U}_{\mathcal{V}}\mathbb{R}_+^{|\mathcal{V}|}$ and $\mathbf{U}_{\mathcal{V}^c}\mathbb{R}_+^{|\mathcal{V}^c|}$ respectively. Since $\|\hat{\mathbf{y}} - \hat{\mathbf{y}^c}\|_2 > 0$ is true for arbitrary pairs of $\hat{\mathbf{y}}$ and $\hat{\mathbf{y}}^c$, the relative interiors of $\mathbf{U}_{\mathcal{V}}\mathbb{R}_+^{|\mathcal{V}|}$ and $\mathbf{U}_{\mathcal{V}^c}\mathbb{R}_+^{|\mathcal{V}^c|}$ are disjoint. Therefore, from [149, Theorem 1.32], there exists a hyperplane passing through the origin that separates the cones properly. From [41, Prop. 1], the existence of such a hyperplane is sufficient for (6.7), and by extension (6.4), to be singleton.

6.2.1 Proof of Theorem 6.2.1

If $S_x < 0.5(1 + 1/\mu_x)$, the quotient polytope $\mathbf{X}\mathcal{C}^{K_x}$ is centrally S_x-neighborly [68, Corollary 1.1]. Since vertices$(\mathcal{T}_{\mathbf{0}}^{K_x}) - \{\mathbf{0}\} \subset$ vertices$(\mathcal{C}^{K_x})$, central S_x-neighborliness of $\mathbf{X}\mathcal{C}^{K_x}$ implies outward S_x-neighborliness of $\mathbf{X}\mathcal{T}_{\mathbf{0}}^{K_x}$. Note that the vertex $\mathbf{0}$ will be

neglected when considering the outward neighborliness. Combining the assumption that $\mathbf{X} \in \mathcal{M}^+$, from Lemma 6.2.2, the set defined in (6.4) is singleton.

6.3 Combined Sparse Representations

We now turn to investigate the problem of combined sparse representations, where a part of the coefficient support is constrained to be non-negative. For the combined representation model given in (6.2), the number of non-zero coefficients and the coefficient support for $\boldsymbol{\alpha}$ are given by S_x and $\mathcal{X}$ respectively. For $\boldsymbol{\beta}$, they are respectively denoted as S_d and $\mathcal{D}$. Let us define the combined representation vector $\boldsymbol{\delta} = [\boldsymbol{\alpha}^T \ \boldsymbol{\beta}^T]^T$ and the combined dictionary $\mathbf{G} = [\mathbf{X} \quad \mathbf{D}]$. The set $\mathcal{G}$ indexes the non-zero coefficients in $\boldsymbol{\delta}$. The length of $\boldsymbol{\delta}$ is denoted by K_g and its number of non-zero coefficients is referred to as S_g. We will refer to the coefficient vector $\boldsymbol{\delta}$ as the combined representation, since it contains both non-negative and general entries from the coefficient vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ respectively. We will define the cross-coherence between the matrices $\mathbf{X}$ and $\mathbf{D}$ as

$$\mu_g = \max_{i,j} \frac{|\mathbf{x}_i^T \mathbf{d}_j|}{\|\mathbf{x}_i\|_2 \|\mathbf{d}_j\|_2}. \tag{6.8}$$

We will present deterministic sparsity thresholds for recovery of the ML_0 solution of (6.2) when the coefficient supports are unknown as well as partially known.

6.3.1 Non-Zero Supports of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ Known

The vectors $\boldsymbol{\alpha}_1 \in \mathbb{R}^{S_x}$ and $\boldsymbol{\beta}_1 \in \mathbb{R}^{S_d}$ contain the non-zero coefficients of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ indexed by $\mathcal{X}$ and $\mathcal{D}$ respectively. The matrices $\mathbf{X}_1$ and $\mathbf{D}_1$ contain the columns of $\mathbf{X}$ and $\mathbf{D}$ indexed by the sets $\mathcal{X}$ and $\mathcal{D}$ respectively. Since the coefficient supports are known, we can express (6.2) as

$$\mathbf{y} = \mathbf{X}_1 \boldsymbol{\alpha}_1 + \mathbf{D}_1 \boldsymbol{\beta}_1, \tag{6.9}$$

where $\boldsymbol{\alpha}_1 \geq 0$. We define $\boldsymbol{\delta}_1 = [\boldsymbol{\alpha}_1^T \quad \boldsymbol{\beta}_1^T]^T$ and the matrix $\mathbf{G}_1 = [\mathbf{X}_1 \quad \mathbf{D}_1]$. Recovery can be performed using least squares with inequality constraints (LSI) [4, Chap. 23]

as

$$\min_{\boldsymbol{\delta}_1} \|\mathbf{y} - \mathbf{G}_1\boldsymbol{\delta}_1\|_2 \text{ subject to } \mathbf{I}_{\mathcal{X}}\boldsymbol{\delta}_1 \geq \mathbf{0}, \tag{6.10}$$

where $\mathbf{I}_{\mathcal{X}} = [\mathbf{I}_{S_x} \quad \mathbf{0}_{S_x,S_g}]$ is the indicator matrix such the constraints $\mathbf{I}_{\mathcal{X}}\boldsymbol{\delta}_1 \geq \mathbf{0}$ and $\boldsymbol{\alpha}_1 \geq \mathbf{0}$ are equivalent.

If the matrix $\mathbf{G}_1$ has full column rank, $\boldsymbol{\delta}_1$ can be estimated by just using least squares (LS) instead of LSI, as the additional constraint in (6.10) will not impact the solution. The following theorem presents a sufficient condition for $\mathbf{G}_1$ to be of full column rank.

Theorem 6.3.1 *([62,63]) For the system defined in (6.9), the matrix* $\mathbf{G}_1 = [\mathbf{X}_1 \ \mathbf{D}_1]$ *has full column rank if*

$$S_x S_d < \frac{[1 - \mu_x(S_x - 1)]^+[1 - \mu_d(S_d - 1)]^+}{\mu_g^2}. \tag{6.11}$$

6.3.2 Non-Zero Support of $\boldsymbol{\beta}$ Alone Known

We will now consider the case where the non-zero support of $\boldsymbol{\beta}$ given by the set $\mathcal{D}$ is known for the system in (6.2). We will derive conditions for unique recovery of $\boldsymbol{\alpha}$ using NN-BP and NNLS. With the knowledge of non-zero support of $\boldsymbol{\beta}$, we can rewrite (6.2) as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\alpha} + \mathbf{D}_1\boldsymbol{\beta}_1. \tag{6.12}$$

Define $\mathbf{P}_{\mathcal{D}}$ to be the projection matrix for the subspace orthogonal to the column space of $\mathbf{D}_1$, i.e.,

$$\mathbf{P}_{\mathcal{D}} = \mathbf{I}_M - \mathbf{D}_1\mathbf{D}_1^{\dagger}. \tag{6.13}$$

Premultiplying (6.12) with $\mathbf{P}_{\mathcal{D}}$, we get

$$\mathbf{P}_{\mathcal{D}}\mathbf{y} = \mathbf{P}_{\mathcal{D}}\mathbf{X}\boldsymbol{\alpha} \text{ where } \boldsymbol{\alpha} \geq 0. \tag{6.14}$$

Let us define $\tilde{\mathbf{y}} = \mathbf{P}_{\mathcal{D}}\mathbf{y}$ and $\tilde{\mathbf{X}} = \mathbf{P}_{\mathcal{D}}\mathbf{X}$, such that (6.14) becomes

$$\tilde{\mathbf{y}} = \tilde{\mathbf{X}}\boldsymbol{\alpha} \text{ where } \boldsymbol{\alpha} \geq 0. \tag{6.15}$$

The condition for recovery of the unique solution $\boldsymbol{\alpha}$ from (6.15) using NN-BP is

$$S_x < 0.5\left(1 + \frac{1}{\mu_{\tilde{x}}}\right), \tag{6.16}$$

where $\mu_{\tilde{x}}$ is the coherence of $\tilde{\mathbf{X}}$.

Lemma 6.3.2 *([63]) The coherence of $\tilde{\mathbf{X}}$, given by $\mu_{\tilde{x}}$ can be upper bounded as*

$$\mu_{\tilde{x}} \leq 0.5\left(\frac{[1-\mu_d(S_d-1)]^+(1+\mu_x)}{\mu_x[1-\mu_d(S_d-1)]^+ + S_b\mu_g^2}\right). \tag{6.17}$$

The above lemma follows directly from [63, Theorem 5]. This also implies that for the existence of $\mathbf{D}_1^\dagger$, we need to have $S_d < 1 + 1/\mu_d$.

Lemma 6.3.3 *Let*

$$\{\hat{\boldsymbol{\alpha}} | \mathbf{y} = \mathbf{X}\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\alpha}} \geq 0\} = \{\boldsymbol{\alpha}\}. \tag{6.18}$$

For a given non-zero support set $\mathcal{D}$ of $\boldsymbol{\beta}$

$$\{\hat{\boldsymbol{\alpha}} | \tilde{\mathbf{y}} = \tilde{\mathbf{X}}\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\alpha}} \geq 0\} = \{\boldsymbol{\alpha}\} \tag{6.19}$$

holds if (6.16) is satisfied, and

$$\exists \mathbf{h} \text{ such that } \mathbf{h}^T\mathbf{X} > 0 \text{ and } \mathbf{h}^T\mathbf{D}_1 = \mathbf{0}. \tag{6.20}$$

Proof From Theorem 6.2.1, we know that the singleton condition (6.19) holds true if (a) the condition in (6.16) is satisfied, and (b) $\exists \mathbf{r}$ such that $\mathbf{r}^T\tilde{\mathbf{X}} > 0$. Since (6.18) is true by assumption, $\exists \mathbf{h}$ such that $\mathbf{h}^T\mathbf{X} > 0$. For $\mathbf{h}^T\mathbf{X} > 0$ and $\mathbf{r}^T\tilde{\mathbf{X}} > 0$ to hold together, we should have $\mathbf{h} = \mathbf{P}_{\mathcal{D}}^T\mathbf{r}$. Therefore, we have $\mathbf{h}^T\mathbf{D}_1 = \mathbf{0}$, following the definition of $\mathbf{P}_{\mathcal{D}}$ in (6.13).

If the sufficient conditions in Lemma 6.3.3 are satisfied, NNLS can be used to recover the unique solution of (6.14), for a given non-zero support $\mathcal{D}$ of $\boldsymbol{\beta}$.

6.3.3 Non-zero Supports of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are Unknown

When the supports of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ in (6.2) are unknown, we will first consider the problem of recovering the coefficients using the convex program,

$$\min_{\boldsymbol{\delta}} \|\boldsymbol{\delta}\|_1 \text{ subject to } \mathbf{y} = \mathbf{G}\boldsymbol{\delta},\ \mathbf{I}_{\bar{\mathcal{X}}}\boldsymbol{\delta} \geq \mathbf{0}, \tag{6.21}$$

which we refer to as COMB-BP. Here, $\mathbf{I}_{\bar{\mathcal{X}}} = [\mathbf{I}_{K_x} \quad \mathbf{0}_{K_x,K_g}]$ is an indicator matrix that picks out $\boldsymbol{\alpha}$ from the vector $\boldsymbol{\delta}$ such that the constraint in (6.21) is equivalent to $\boldsymbol{\alpha} \geq 0$. When deriving the threshold on S_g for the recovery of the ML_0 solution, without loss of generality, we assume that $S_x \leq S_d$ and $\mu_x \leq \mu_d$. Similar thresholds can be derived for the other cases also.

Condition for Recovering the ML_0 Solution using the Convex Program

The sufficient condition for the COMB-BP to recover the ML_0 solution is

$$\max_{i \in \mathcal{G}^c} \|\mathbf{G}_1^{\dagger}\mathbf{g}_i\|_1 < 1. \tag{6.22}$$

This condition is same as the one given in [150, Theorem 3.3] for recovery of a general sparse vector using BP, since the ℓ_1 norm does not depend on the sign of the coefficients. From [62, Theorem 3], the condition (6.22) can be expressed as

$$(1+\mu_d)(2S_x\mu_d + S_d(\mu_g+\mu_d)) + 2S_xS_d(\mu_g^2-\mu_d^2) < (1+\mu_d)^2. \tag{6.23}$$

The threshold on the total number of non-zero coefficients, S_g, is derived using (6.23), and can be found in [62, Corollary 3].

Condition for Recovering the ML_0 Solution using a Greedy Algorithm

We propose a greedy pursuit algorithm that can be used to recover the ML_0 solution from (6.2). The proposed COMB-OMP algorithm follows a procedure similar to the OMP algorithm [150] and is presented in Appendix A. The stopping criterion for this algorithm is either the maximum number of iterations/non-zero coefficients, T,

or the ℓ_2 norm of the residual, ϵ. In the algorithm, $\pi(i)$ denotes the correlations computed for the current residual with the normalized atom $\mathbf{g}_i$. When updating the index set of chosen dictionary atoms, $\mathcal{G}_t$, we consider only the positive maximum correlation for atoms corresponding to $\mathbf{X}$ and absolute maximum correlation for atoms corresponding to $\mathbf{D}$. This is consistent with our combined representation scheme. The solution update can be performed using a constrained least squares procedure. The final debiasing step computes the solution using the LSI algorithm described in Section 6.3.1. This step will be ignored when deriving the sparsity threshold, since it improves the solution only when the sparsity threshold is not satisfied and when there is additive noise in the combined model (6.2).

The sufficient sparsity threshold on the coefficient vector under which the COMB-OMP will recover the ML_0 solution will be investigated. Some of the strategies used in the proofs are inspired by similar techniques used in [62, 63, 150]. In order to derive the threshold, we will divide the dictionary $\mathbf{G} = [\mathbf{X} \quad \mathbf{D}]$ into four sub-dictionaries $\mathbf{X}_1 \in \mathbb{R}^{M \times S_x}$, $\mathbf{X}_2 \in \mathbb{R}^{M \times (K_x - S_x)}$, $\mathbf{D}_1 \in \mathbb{R}^{M \times S_d}$ and $\mathbf{D}_2 \in \mathbb{R}^{M \times (K_d - S_d)}$. We assume that the matrix $\mathbf{G}_1 = [\mathbf{X}_1 \ \mathbf{D}_1]$ contains the atoms that participate in the representation and $\mathbf{G}_2 = [\mathbf{X}_2 \ \mathbf{D}_2]$ contains those that do not participate. This implies that the signal $\mathbf{y}$ can be represented as

$$\mathbf{y} = \mathbf{X}_1 \boldsymbol{\alpha}_1 + \mathbf{D}_1 \boldsymbol{\beta}_1, \tag{6.24}$$

where the elements of $\boldsymbol{\alpha}_1 \in \mathbb{R}^{S_x}$ are strictly positive and those of $\boldsymbol{\beta}_1 \in \mathbb{R}^{S_d}$ are non-zero.

Lemma 6.3.4 *When the matrix $\mathbf{G}_1 = [\mathbf{X}_1 \ \mathbf{D}_1]$ has full column rank, $\mathbf{y}$ is given by (6.24), and the residual $\mathbf{r}_t$ of COMB-OMP satisfies*

$$\max(\max(\mathbf{X}_1^T \mathbf{r}_t, \mathbf{0}), \|\mathbf{D}_1^T \mathbf{r}_t\|_\infty) = \|\mathbf{G}_1^T \mathbf{r}_t\|_\infty, \tag{6.25}$$

the sufficient condition for COMB-OMP to uniquely recover the ML_0 solution from (6.2) is

$$\max_{i \in \mathcal{G}^c} \|\mathbf{G}_1^\dagger \mathbf{g}_i\|_1 < 1. \tag{6.26}$$

Proof For COMB-OMP to recover the unique sparsest representation, no atom from $\mathbf{G}_2$ must enter the support set $\mathcal{G}_t$ at any iteration. Therefore, the residual $\mathbf{r}_t$ at each iteration t must satisfy the condition

$$\rho(\mathbf{r}_t) \equiv \frac{\max(\max(\mathbf{X}_2^T \mathbf{r}_t, \mathbf{0}), \|\mathbf{D}_2^T \mathbf{r}_t\|_\infty)}{\max(\max(\mathbf{X}_1^T \mathbf{r}_t, \mathbf{0}), \|\mathbf{D}_1^T \mathbf{r}_t\|_\infty)} < 1. \tag{6.27}$$

Since

$$\max(\max(\mathbf{X}_2^T \mathbf{r}_t, \mathbf{0}), \|\mathbf{D}_2^T \mathbf{r}_t\|_\infty) \leq \|\mathbf{G}_2^T \mathbf{r}_t\|_\infty,$$

$\rho(\mathbf{r}_t)$ can be bounded as

$$\rho(\mathbf{r}_t) \leq \frac{\|\mathbf{G}_2^T \mathbf{r}_t\|_\infty}{\max(\max(\mathbf{X}_1^T \mathbf{r}_t, \mathbf{0}), \|\mathbf{D}_1^T \mathbf{r}_t\|_\infty)} \tag{6.28}$$

$$= \frac{\|\mathbf{G}_2^T (\mathbf{G}_1^\dagger)^T \mathbf{G}_1^T \mathbf{r}_t\|_\infty}{\max(\max(\mathbf{X}_1^T \mathbf{r}_t, \mathbf{0}), \|\mathbf{D}_1^T \mathbf{r}_t\|_\infty)} \tag{6.29}$$

$$\leq \frac{\|\mathbf{G}_2^T (\mathbf{G}_1^\dagger)^T\|_{\infty,\infty} \|\mathbf{G}_1^T \mathbf{r}_t\|_\infty}{\max(\max(\mathbf{X}_1^T \mathbf{r}_t, \mathbf{0}), \|\mathbf{D}_1^T \mathbf{r}_t\|_\infty)} \tag{6.30}$$

$$= \|\mathbf{G}_2^T (\mathbf{G}_1^\dagger)^T\|_{\infty,\infty} \tag{6.31}$$

$$= \|\mathbf{G}_1^\dagger \mathbf{G}_2\|_{1,1} \tag{6.32}$$

$$= \max_{i \in \mathcal{G}^c} \|\mathbf{G}_1^\dagger \mathbf{g}_i\|_1 \tag{6.33}$$

Eqn. (6.29) holds since $(\mathbf{G}_1^\dagger)^T \mathbf{G}_1^T$ is an orthoprojector onto the column space of $\mathbf{G}_1$. Both $\mathbf{y}$ and $\mathbf{G}\boldsymbol{\delta_t}$ lie in the column space of $\mathbf{G}_1$ and hence $\mathbf{r}_t$ lies in the same space. The properties of $\|.\|_{\infty,\infty}$ ensures that (6.30) is true. By assumption, the denominator of (6.30) equals $\|\mathbf{G}_1^T \mathbf{r}_t\|_\infty$. Therefore, (6.31) holds true and (6.32) follows from relation $\|\mathbf{A}^T\|_{\infty,\infty} = \|\mathbf{A}\|_{1,1}$ for any matrix $\mathbf{A}$. From (6.27), (6.28) and (6.33), the sufficient condition provided in (6.26) is obtained.

Note that the sufficient condition (6.26) given in Lemma 6.3.4 is the same for a general representation also [62, 150]. However, the important difference in the case of recovery using COMB-OMP is that (6.26) becomes sufficient only when (6.25) holds. As we will see in the following lemmas, this will lead to a significant difference in terms of sparsity threshold when compared to COMB-BP. We will first derive conditions under which the first step of the COMB-OMP, when $\mathbf{y} = \mathbf{r}_0$, will satisfy (6.25). This will then be extended to the residuals at all steps, $\mathbf{r}_t$, where $t \geq 1$.

Lemma 6.3.5 *When the matrix* $\mathbf{G}_1 = [\mathbf{X}_1 \ \mathbf{D}_1]$ *has full column rank, and* $\mathbf{y}$ *is given as (6.24), (6.25) will be satisfied for* $\mathbf{y} = \mathbf{r}_0$ *if*

$$(S_x - 1)\mu_d + S_d\mu_g < \frac{1}{2}. \tag{6.34}$$

Proof For (6.25) to be satisfied, the sufficient condition is that

$$\max(\mathbf{X}_1^T\mathbf{r}_t, \mathbf{0}) = \|\mathbf{X}_1^T\mathbf{r}_t\|_\infty. \tag{6.35}$$

Therefore, we only have to consider the case where an atom from $\mathbf{X}_1$ will be picked. Let us denote $\boldsymbol{\delta}_1 = [\boldsymbol{\alpha}_1^T \quad \boldsymbol{\beta}_1^T]^T$ and $\mathbf{z} = \mathbf{X}_1^T\mathbf{y} = \mathbf{X}_1^T\mathbf{G}_1\boldsymbol{\delta}_1$. We will derive the bounds on the maximum positive value, z_m, and the minimum negative value, z_n, of $\mathbf{z}$. We denote the smallest possible lower bound on z_m as $\hat{z}_m$, and the largest possible lower bound on $|z_n|$ as $\hat{z}_n$. The worst-case guarantee for (6.35) to be true is

$$\hat{z}_m > \hat{z}_n. \tag{6.36}$$

Using the fact that

$$\mathbf{X}_1^T\mathbf{G}_1 = [\mathbf{I}_{S_x} \quad \mathbf{0}] + [\mathbf{X}_1^T\mathbf{X}_1 - \mathbf{I}_{S_x} \quad \mathbf{X}_1^T\mathbf{D}_1],$$

the correlation vector $\mathbf{z}$ can be expressed as

$$\mathbf{z} = \boldsymbol{\alpha}_1 + [\mathbf{X}_1^T\mathbf{X}_1 - \mathbf{I}_{S_x} \quad \mathbf{X}_1^T\mathbf{D}_1]\boldsymbol{\delta}_1. \tag{6.37}$$

Let us define $\mathbf{C}_1 = [\mathbf{X}_1^T\mathbf{X}_1 - \mathbf{I}_{S_x} \quad \mathbf{X}_1^T\mathbf{D}_1]$ and the elementwise bounds on the submatrices are,

$$\begin{aligned}|\mathbf{X}_1^T\mathbf{X}_1 - \mathbf{I}_{S_x}| &\leq \mu_x(\mathbf{1}_{S_x,S_x} - \mathbf{I}_{S_x}) \\ &\leq \mu_d(\mathbf{1}_{S_x,S_x} - \mathbf{I}_{S_x}),\end{aligned}$$

and $\mathbf{X}_1^T\mathbf{D}_1 \leq |\mu_g\mathbf{1}_{S_x,S_d}|$. It is clear that the maximum row sum of $\mathbf{C}_1$ is

$$\|\mathbf{C}_1\|_{\infty,\infty} \leq (S_x - 1)\mu_d + S_d\mu_g. \tag{6.38}$$

In order to derive the smallest lower bound on z_m, we will assume that all the coefficients in $\boldsymbol{\delta}_1$ have the same absolute value given by α, and hence, from (6.37), we have

$$\begin{aligned}z_m &\geq \alpha - \alpha\|\mathbf{C}_1\|_{\infty,\infty} \\ &\geq \alpha(1 - [(S_x - 1)\mu_d + S_d\mu_g]) \equiv \hat{z}_m.\end{aligned} \tag{6.39}$$

The required bound on z_n can be obtained by setting one element of $\boldsymbol{\alpha}_1$ as $\hat{\alpha}$, where $0 < \hat{\alpha} < \alpha$, and the absolute value of all the other elements of $\boldsymbol{\delta}_1$ as α. We now have

$$z_n \geq \hat{\alpha} - \alpha\|\mathbf{C}_1\|_{\infty,\infty},$$

and as $\hat{\alpha} \to 0$,

$$|z_n| < \alpha[(S_x - 1)\mu_d + S_d\mu_g] \equiv \hat{z}_n, \tag{6.40}$$

which is the largest possible lower bound on z_n. Substituting (6.39) and (6.40) in (6.36), results in (6.34).

The condition given by (6.34) needs to be satisfied even if there is one non-negative component in the combined representation. For now, let us assume that (6.25) holds for all $\mathbf{r}_t$, where $t \geq 1$, and derive the threshold on S_g such that the ML_0 solution can be recovered from (6.2). It will be shown later in the section that the threshold on S_g obtained indeed implies that (6.25) holds for all $\mathbf{r}_t, t \geq 1$.

Lemma 6.3.6 *When the matrix* $\mathbf{G}_1$ *has full column rank, and* $\mathbf{y}$ *is given as (6.24), the sufficient condition for (6.26) to be satisfied is*

$$\frac{S_x\mu_d + S_d\mu_g}{1 - (S_x\mu_d + S_d\mu_g - \mu_d)} < 1 \tag{6.41}$$

Proof The condition for success of COMB-OMP can be written as

$$\max_{i\in\mathcal{G}^c} \|\mathbf{G}_1^\dagger \mathbf{g}_i\|_1 \leq \|(\mathbf{G}_1^T\mathbf{G}_1)^{-1}\|_{1,1} \max_{i\in\mathcal{G}^c} \|\mathbf{G}_1^T\mathbf{g}_i\|_1, \tag{6.42}$$

using the property of $\|.\|_{1,1}$ and the fact that $\mathbf{G}_1^\dagger = (\mathbf{G}_1^T\mathbf{G}_1)^{-1}\mathbf{G}_1^T$.

In order to compute the lower bound for $\|(\mathbf{G}_1^T\mathbf{G}_1)^{-1}\|_{1,1}$, we first expand the Gramm matrix

$$\mathbf{G}_1^T\mathbf{G}_1 = \mathbf{I}_{S_g} + \mathbf{C}, \tag{6.43}$$

where

$$\mathbf{C} \equiv \begin{bmatrix} \mathbf{X}_1^T\mathbf{X}_1 - \mathbf{I}_{S_x} & \mathbf{X}_1^T\mathbf{D}_1 \\ \mathbf{D}_1^T\mathbf{X}_1 & \mathbf{D}_1^T\mathbf{D}_1 - \mathbf{I}_{S_d} \end{bmatrix}.$$

$\mathbf{C}$ can be bounded elementwise as

$$\begin{aligned} |\mathbf{C}| &\leq \begin{bmatrix} \mu_x(\mathbf{1}_{S_x,S_x} - \mathbf{I}_{S_x}) & \mu_g\mathbf{1}_{S_x,S_d} \\ \mu_g\mathbf{1}_{S_d,S_x} & \mu_d(\mathbf{1}_{S_d,S_d} - \mathbf{I}_{S_d}) \end{bmatrix} \\ &\leq \begin{bmatrix} \mu_d(\mathbf{1}_{S_x,S_x} - \mathbf{I}_{S_x}) & \mu_g\mathbf{1}_{S_x,S_d} \\ \mu_g\mathbf{1}_{S_d,S_x} & \mu_d(\mathbf{1}_{S_d,S_d} - \mathbf{I}_{S_d}) \end{bmatrix}, \end{aligned}$$

since $\mu_x \leq \mu_d$ by assumption. The maximum column sum of $\mathbf{C}$ is bounded as

$$\|\mathbf{C}\|_{1,1} \leq (S_x - 1)\mu_d + S_d\mu_g, \tag{6.44}$$

since $S_x \leq S_d$. From (6.43), we also observe that $\|\mathbf{C}\|_{1,1} < 1$, since $\mathbf{G}_1^T\mathbf{G}_1$ is strictly diagonally dominant, because of the linear independence of the columns of $\mathbf{G}_1$. Using (6.43), we can write

$$\|(\mathbf{G}_1^T\mathbf{G}_1)^{-1}\|_{1,1} = \|(\mathbf{I}_{S_g} + \mathbf{C})^{-1}\|_{1,1}. \tag{6.45}$$

The Neumann series $\sum_k (-\mathbf{C})^k$ converges to $(\mathbf{I}_{S_g} + \mathbf{C})^{-1}$, whenever $\|\mathbf{C}\|_{1,1} < 1$ [151] and hence (6.45) can be expressed as

$$\begin{aligned}
\|(\mathbf{G}_1^T\mathbf{G}_1)^{-1}\|_{1,1} &= \left\|\sum_{k=1}^{\infty}(-\mathbf{C})^k\right\|_{1,1} \\
&\leq \sum_{k=1}^{\infty}\|\mathbf{C}\|_{1,1}^k \\
&= \frac{1}{1-\|\mathbf{C}\|_{1,1}} \\
&\leq \frac{1}{1-(S_x\mu_d + S_d\mu_g - \mu_d)},
\end{aligned} \tag{6.46}$$

using (6.44). If $\mathbf{g}_i$ is a vector chosen from $\mathbf{X}_2$, $|\mathbf{G}_1^T\mathbf{g}_i| \leq [\mu_d\mathbf{1}_{S_x}^T \quad \mu_g\mathbf{1}_{S_d}^T]^T$ and hence we have

$$\max_{i\in\mathcal{G}^c}\|\mathbf{G}_1^T\mathbf{g}_i\|_1 \leq S_x\mu_d + S_d\mu_g = (S_x+S_d)\mu_d + S_d(\mu_g-\mu_d). \tag{6.47}$$

If $\mathbf{g}_i$ is chosen from $\mathbf{D}_2$, $|\mathbf{G}_1^T\mathbf{g}_i| \leq [\mu_g\mathbf{1}_{S_x}^T \quad \mu_d\mathbf{1}_{S_d}^T]^T$. Therefore

$$\max_{i\in\mathcal{G}^c}\|\mathbf{G}_1^T\mathbf{g}_i\|_1 \leq S_x\mu_g + S_d\mu_d = (S_x+S_d)\mu_d + S_x(\mu_g-\mu_d). \tag{6.48}$$

Since $S_d \geq S_x$, among (6.47) and (6.48), we will choose (6.47) as our bound. Substituting (6.47) and (6.46) in (6.42), we can obtain (6.41) as the condition for COMB-OMP to succeed.

When (6.41) is satisfied, (6.34) holds as well. Since the number of non-zero coefficients, S_x and S_d, of $\boldsymbol{\alpha}_1$ and $\boldsymbol{\beta}_1$ are unknown, we need to derive the condition on recovery that depends only on the number of non-zero coefficients of the combined representation S_g.

Lemma 6.3.7 *When the matrix $\mathbf{G}_1$ has full column rank, and $\mathbf{y}$ is given as (6.24), the sparsity threshold on the combined coefficient vector $\boldsymbol{\delta}$ for (6.41) to hold is*

$$S_g < 0.5\left(1+\frac{1}{\mu_g}\right). \tag{6.49}$$

Proof Rewriting (6.41), we obtain

$$S_x < \frac{1 + \mu_d - 2S_d\mu_g}{2\mu_d} \equiv \psi(S_d).$$

The threshold on the total number of non-zero coefficients, S_g, can be obtained by minimizing $\psi(S_d) + S_d$ over S_d. The constraint is that $S_d \geq 1$ since $S_x \leq S_d$ and the overall representation will have at least one non-zero coefficient. Denoting S to be the sparsity threshold, it can be obtained as

$$S = \min_{S_d \geq 1} [S_d + \psi(S_d)].$$

Relaxing the constraint that S_d is an integer, the minimum will be obtained when $\mu_g = \mu_d$ and the minimum value is

$$S = 0.5 \left(1 + \frac{1}{\mu_g}\right),$$

which is the strict upper bound on S_g.

A similar procedure to obtain the threshold on S_g using (6.34), results in $S_g < 0.5(2 + 1/\mu_g)$, which is only slightly better than (6.49). Moreover, as stated already (6.41) implies (6.34) and not vice-versa. Therefore the sparsity threshold in (6.49) cannot be made better. We will assume that $\mu_g = \mu_d$, as obtained in the proof of the above lemma, and show that the above bound is sufficient for recovering the subsequent atoms using the residuals $\mathbf{r}_t$, when $t \geq 1$.

Lemma 6.3.8 *When the matrix $\mathbf{G}_1$ has full column rank, and $\mathbf{y}$ is given as (6.24), (6.25) will hold true for residual at any step, $\mathbf{r}_t$ for $t \geq 1$, when $\mu_g = \mu_d$ and (6.49) is satisfied.*

Proof Since we assumed that $\mu_g = \mu_d$, we will assume that the overall coherence of $\mathbf{G}_1$ is μ_g and the total number of columns in $\mathbf{G}_1$ is S_g. We will denote $\mathbf{G}_1 = [\mathbf{G}_a \quad \mathbf{G}_b]$, where $\mathbf{G}_a$ with S_a columns contains the atoms already chosen for the representation

and $\mathbf{G}_b$ contains $S_b = S_g - S_a$ atoms, one of which will be chosen by the residual. The residual $\mathbf{r}_t$ will be simply denoted as $\mathbf{r}$ for notational convenience and is obtained using a least squares procedure as

$$\mathbf{r} = \mathbf{y} - \mathbf{G}_a \boldsymbol{\delta}_a, \tag{6.50}$$

where

$$\boldsymbol{\delta}_a = \mathbf{G}_a^{\dagger} \mathbf{y}. \tag{6.51}$$

Let us denote the correlation vector as

$$\mathbf{z} = \mathbf{G}_b^T \mathbf{r}. \tag{6.52}$$

Similar to the proof of Lemma 6.3.5, we are only concerned about recovering the non-negative coefficients as it will give us sufficient conditions under which (6.25) will be satisfied. The bounds on the maximum positive value, z_m, and the minimum negative value, z_n, of $\mathbf{z}$ will be derived assuming that all the elements in $\boldsymbol{\delta}_b$ are constrained to be non-negative. In this case, the smallest possible lower bound on z_m, given by $\hat{z}_m$ and the largest possible lower bound on $|z_n|$ given by $\hat{z}_n$. For (6.25) to hold for any $\mathbf{r}_t$, where $t \geq 1$, similar to the proof of Lemma 6.3.5, we need to show that

$$\hat{z}_m > \hat{z}_n. \tag{6.53}$$

We will first expand (6.52) using (6.50) and (6.51)

$$\begin{aligned} \mathbf{z} &= \mathbf{G}_b^T (\mathbf{y} - \mathbf{G}_a \mathbf{G}_a^{\dagger} \mathbf{y}) \\ &= \mathbf{G}_b^T (\mathbf{I} - \mathbf{G}_a \mathbf{G}_a^{\dagger}) \mathbf{G}_1 \boldsymbol{\delta}_1 \qquad (6.54) \\ &= \mathbf{G}_b^T (\mathbf{I} - \mathbf{G}_a \mathbf{G}_a^{\dagger}) \mathbf{G}_b \boldsymbol{\delta}_b, \qquad (6.55) \end{aligned}$$

which is obtained by substituting $\mathbf{G}_1 = [\mathbf{G}_a \quad \mathbf{G}_b]$ and $\boldsymbol{\delta}_1 = [\boldsymbol{\delta}_a^T \quad \boldsymbol{\delta}_b^T]^T$ in (6.54). Let us denote any two distinct columns from $\mathbf{G}_b$ by $\mathbf{g}_i$ and $\mathbf{g}_j$. Let us also denote the

matrix $\mathbf{Q} = \mathbf{G}_b^T(\mathbf{I} - \mathbf{G}_a\mathbf{G}_a^\dagger)\mathbf{G}_b$, which is of size $S_b \times S_b$, and designate its $(i,j)^{\text{th}}$ element to be q_{ij}. The correlation vector in (6.55) can be now expressed as

$$\mathbf{z} = \mathbf{Q}\boldsymbol{\delta}_b$$

We will compute bounds on the diagonal and off-diagonal elements of $\mathbf{Q}$. Expanding $\mathbf{G}^\dagger$, we can write

$$|q_{ij}| = \left|\mathbf{g}_i^T[\mathbf{I} - \mathbf{G}_a(\mathbf{G}_a^T\mathbf{G}_a)^{-1}\mathbf{G}_a^T]\mathbf{g}_j\right| \tag{6.56}$$

$$\leq \left|\mathbf{g}_i^T\mathbf{g}_j\right| + \left|\mathbf{g}_i^T\mathbf{G}_a(\mathbf{G}_a^T\mathbf{G}_a)^{-1}\mathbf{G}_a^T\mathbf{g}_j\right| \tag{6.57}$$

$$\leq \left|\mathbf{g}_i^T\mathbf{g}_j\right| + \left\|\mathbf{G}_a^T\mathbf{g}_i\right\|_2^2 \left\|(\mathbf{G}_a^T\mathbf{G}_a)^{-1}\right\|_2. \tag{6.58}$$

Eqn. (6.57) follows from applying triangle inequality on (6.56) and (6.58) is obtained by upper bounding the second term in the right hand side of (6.57). We can express

$$\left\|\mathbf{G}_a^T\mathbf{g}_i\right\|_2^2 \leq S_a\mu_g^2$$

since the maximum absolute coherence between any two elements in $\mathbf{G}_1$ is μ_g. Since $\left\|(\mathbf{G}_a^T\mathbf{G}_a)^{-1}\right\|_2 \leq 1/\lambda_{min}(\mathbf{G}_a^T\mathbf{G}_a)$, and by Gershgorin's disc theorem [152, Theorem 6.1.1], $\lambda_{min}(\mathbf{G}_a^T\mathbf{G}_a) \leq [1 - \mu_g(S_a - 1)]^+$, we can rewrite (6.58) as

$$|q_{ij}| \leq \mu_g + \frac{S_a\mu_g^2}{[1 - \mu_g(S_a - 1)]^+}. \tag{6.59}$$

When $i = j$, we have

$$|q_{ii}| = \left|\mathbf{g}_i^T[\mathbf{I} - \mathbf{G}_a(\mathbf{G}_a^T\mathbf{G}_a)^{-1}\mathbf{G}_a^T]\mathbf{g}_i\right| \tag{6.60}$$

$$\geq \left|\mathbf{g}_i^T\mathbf{g}_j\right| - \left|\mathbf{g}_i^T\mathbf{G}_a(\mathbf{G}_a^T\mathbf{G}_a)^{-1}\mathbf{G}_a^T\mathbf{g}_j\right| \tag{6.61}$$

$$\geq 1 - \frac{S_a\mu_g^2}{[1 - \mu_g(S_a - 1)]^+}. \tag{6.62}$$

Eqn. (6.61) follows from applying reverse triangle inequality on (6.60) and (6.62) is obtained by following steps similar to the derivation of upper bound on $|q_{ij}|$. The bounds given by (6.59) and (6.62) are valid only if $1 - \mu_g(S_a - 1) > 0$, which can be

verified by substituting $\mu_g < 1/(2S_g - 1)$, from (6.49), and $S_a < S_g$. Therefore, (6.60) and (6.62) can be rewritten as $|q_{ij}| \leq q_1$, $|q_{ii}| \geq q_2$, where

$$q_1 = C_1 \mu_g$$
$$q_2 = C_1(1 - S_a \mu_g),$$

and $C_1 = (1 + \mu_g)/[1 - \mu_g(S_a - 1)]$.

We know that the diagonal elements of $\mathbf{Q}$ are lower bounded by q_1 and the off-diagonal elements are upper bounded by q_2. Since there are S_b rows in $\mathbf{Q}$, the smallest lower bound on z_m is

$$z_m \geq \alpha q_2 - \alpha(S_b - 1)q_1 \equiv \hat{z}_m \tag{6.63}$$

which is obtained when all the elements in $\boldsymbol{\delta}_b$ are set to α. The required bound on z_n is obtained by setting one element of $\boldsymbol{\delta}_b$ corresponding to a positive coefficient as $\hat{\alpha}$, where $0 < \hat{\alpha} < \alpha$, $\hat{\alpha}$ approaches zero and all the other values in $\boldsymbol{\delta}_b$ are set to α. z_n can be now bounded as $z_n \geq \hat{\alpha} q_2 - \alpha q_1(S_b - 1)$. As $\hat{\alpha} \to 0$, we have

$$|z_n| < \alpha(S_b - 1)q_2 \equiv \hat{z}_n. \tag{6.64}$$

Using (6.53), (6.63) and (6.64), we have

$$\mu_g < \frac{1}{2S_g - S_a - 2},$$

which is always satisfied since we know from (6.49) that $\mu_g < 1/(2S_g - 1)$ and $S_a \geq 1$.

Now, we are ready to state our main theorem without proof, since it follows directly from the lemmas stated in this section.

Theorem 6.3.9 *For any* $\mathbf{y}$ *that follows the combined model in (6.2), COMB-OMP will recover the* ML_0 *solution if the number of non-zero coefficients,* S_g*, is less than* $0.5(1 + 1/\mu_g)$.

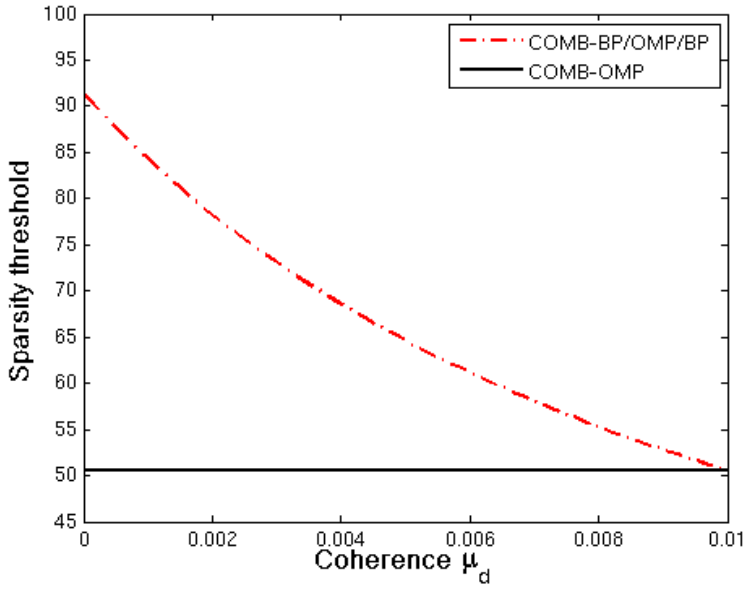


Figure 6.1: Deterministic sparsity thresholds for COMB-BP, BP, COMB-OMP and OMP with $\mu_g = 0.01$ in order to recover the ML_0 solution from (6.2).

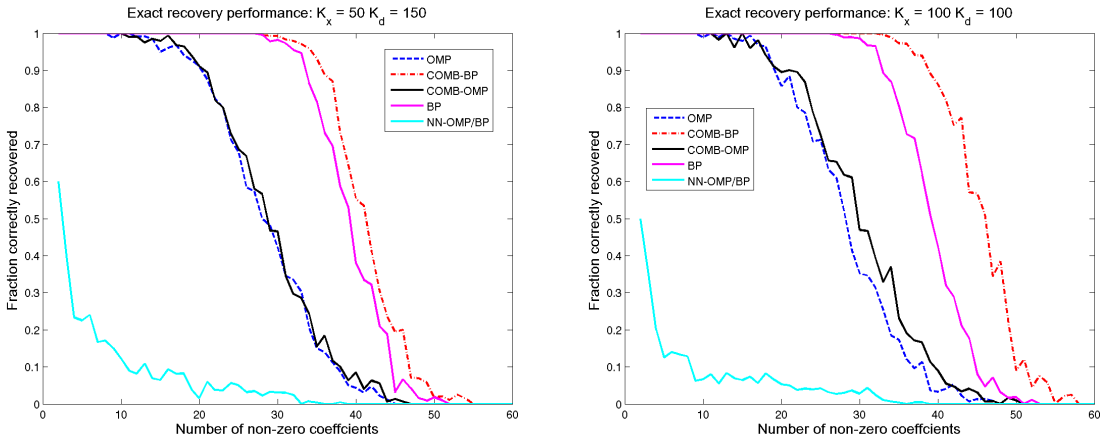
From the lemmas proved in this section, it is clear that this threshold cannot be made better. Contrast this with the case of recovery using a convex program discussed in Section 6.3.3, as well as sparsity thresholds for recovery using BP and OMP when $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are general sparse vectors [62, Eqn. (13)]. Figure 6.1 compares the thresholds on S_g when $\mu_g = 0.01$ and μ_d varies from 0 to 0.01. We can see that an improvement up to a factor of two can be observed with COMB-BP, BP and OMP algorithms when compared to COMB-OMP. From the proof of Lemma 6.3.5, it can be observed that introducing non-negative constraint on even one coefficient in the representation drastically alters the deterministic sparsity threshold of greedy OMP-like algorithms. Note that, however, the sparsity thresholds are pessimistic and in the experiments provided in the next section, COMB-OMP performs better than OMP and also has much reduced computational complexity when compared to COMB-BP and BP.

6.4 Experiments

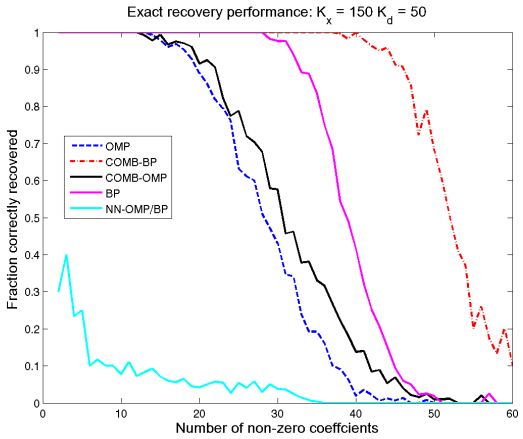
The COMB-BP and the COMB-OMP algorithms incorporate the prior knowledge that a set of coefficients to be recovered are non-negative. If we use BP and OMP algorithms for recovery, this prior knowledge cannot be exploited. In order to establish that this additional information leads to improvement in recovery performance, we performed numerical experiments by realizing the elements of dictionary $\mathbf{G}$ from an i.i.d. zero-mean, unit-variance, Gaussian distributions. The non-zero coefficients in the coefficient vector $\boldsymbol{\delta}$ are either signs (± 1) or realized from a uniform distribution. We varied the proportion of the non-negative and the unconstrained coefficients in the combined representation and tested the performance of COMB-BP, BP, COMB-OMP and OMP algorithms in exact and approximate recovery. For the case of exact recovery, we also compared NN-BP and NN-OMP algorithms which impose the constraint that all coefficients are non-negative.

The total number of atoms in $\mathbf{G}$ was fixed at $K_g = 200$. The three cases tested were (a) $K_x = 50$, $K_d = 150$, (b) $K_x = 100$, $K_d = 100$, and (c) $K_x = 150$, $K_d = 50$. The location of the non-zero coefficients in $\boldsymbol{\delta}$ were fixed uniformly at random. When the non-zero coefficients were realized from a uniform distribution, the non-negative coefficients were obtained from the uniform distribution $U(0, 1)$ and the general coefficients were obtained from $U(-1, 1)$. When the non-zero coefficients were signs, the general coefficients were obtained with equiprobable positive and negative signs. The number of non-zero coefficients S_x and S_d were varied from 1 to 30 each, and hence the total number of non-zero coefficients, S_g, varied from 2 to 60. $\mathbf{y}$ was obtained using the combined model (6.2), and coefficient recovery was performed using the six algorithms. The relative recovery error between the recovered coefficient vector $\hat{\boldsymbol{\delta}}$ and the actual coefficient vector $\boldsymbol{\delta}$ is

$$RRE = \frac{\|\hat{\boldsymbol{\delta}} - \boldsymbol{\delta}\|_2^2}{\|\boldsymbol{\delta}\|_2^2}. \tag{6.65}$$



(a) (b)



(c)

Figure 6.2: Exact recovery performance of COMB-BP, BP, COMB-OMP, OMP, NN-OMP and NN-BP when $\mathbf{G}$ is obtained from a Gaussian ensemble and the non-zero coefficients are realized from a uniform distribution. (a) $K_x = 50$ and $K_d = 150$, (b) $K_x = 100$ and $K_d = 100$, (a) $K_x = 150$ and $K_d = 50$.

If the RRE is less than 10^{-6}, the coefficient is said to be recovered exactly. Each experiment was repeated 10 times in order to measure the average performance in cases of both exact and approximate recovery. The average relative recovery error is the mean of RRE over all iterations.

Let us first consider the case when the coefficients are realized from the uniform distribution. From Figure 6.2 it can be seen that as K_x increases, the performance of

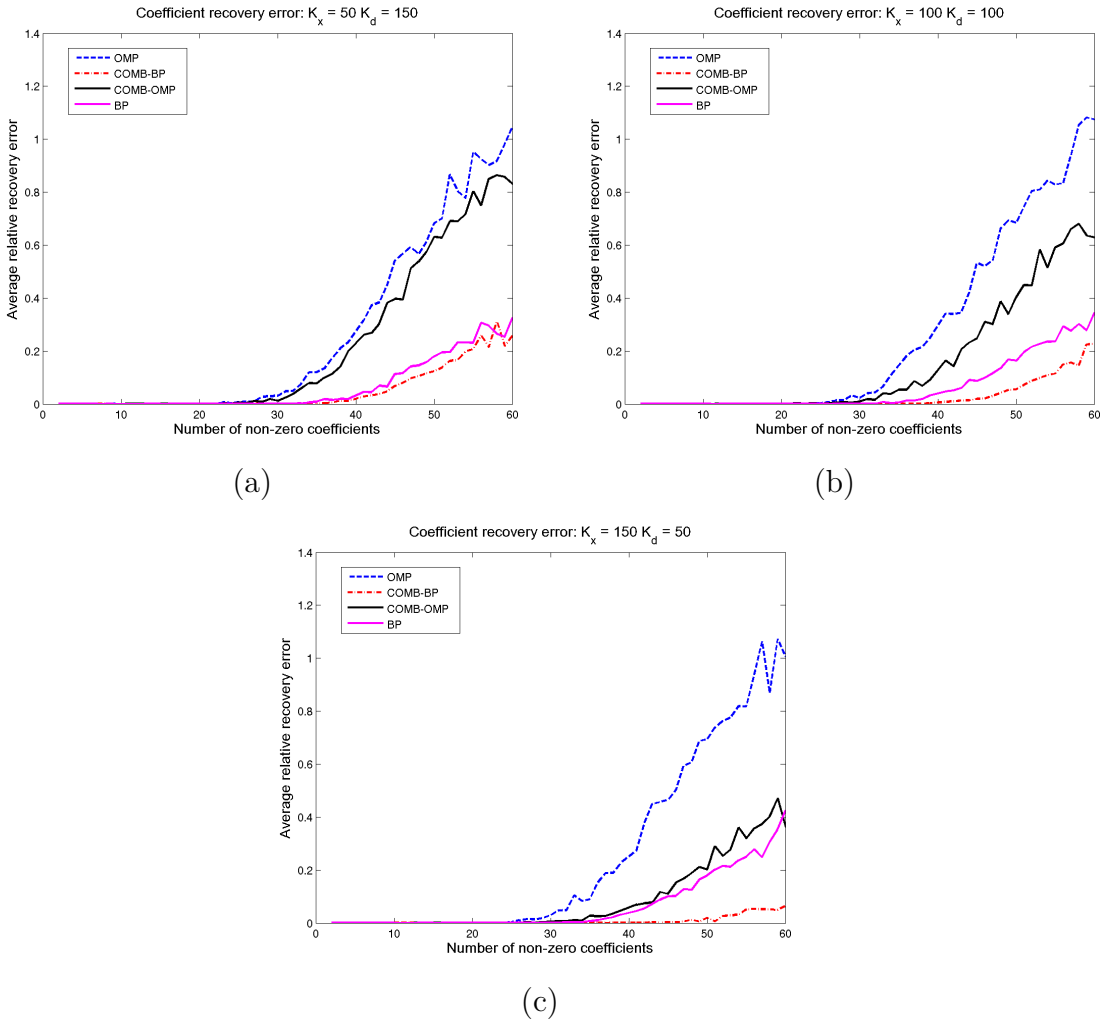


Figure 6.3: Average relative recovery error of COMB-BP, BP, COMB-OMP and OMP when $\mathbf{G}$ is obtained from a Gaussian ensemble and the non-zero coefficients are realized from a uniform distribution. (a) $K_x = 50$ and $K_d = 150$, (b) $K_x = 100$ and $K_d = 100$, (a) $K_x = 150$ and $K_d = 50$.

COMB-BP and COMB-OMP become increasingly better when compared to that of BP and OMP respectively. In particular, the performance of COMB-BP substantially improves as the non-negative component in the representation becomes bigger. Note that the recovery performance of NN-OMP and NN-BP algorithms do not compare well with the rest of the algorithms, since they impose the constraint that all coefficients are non-negative, whereas actually only a part of them are. Furthermore, in

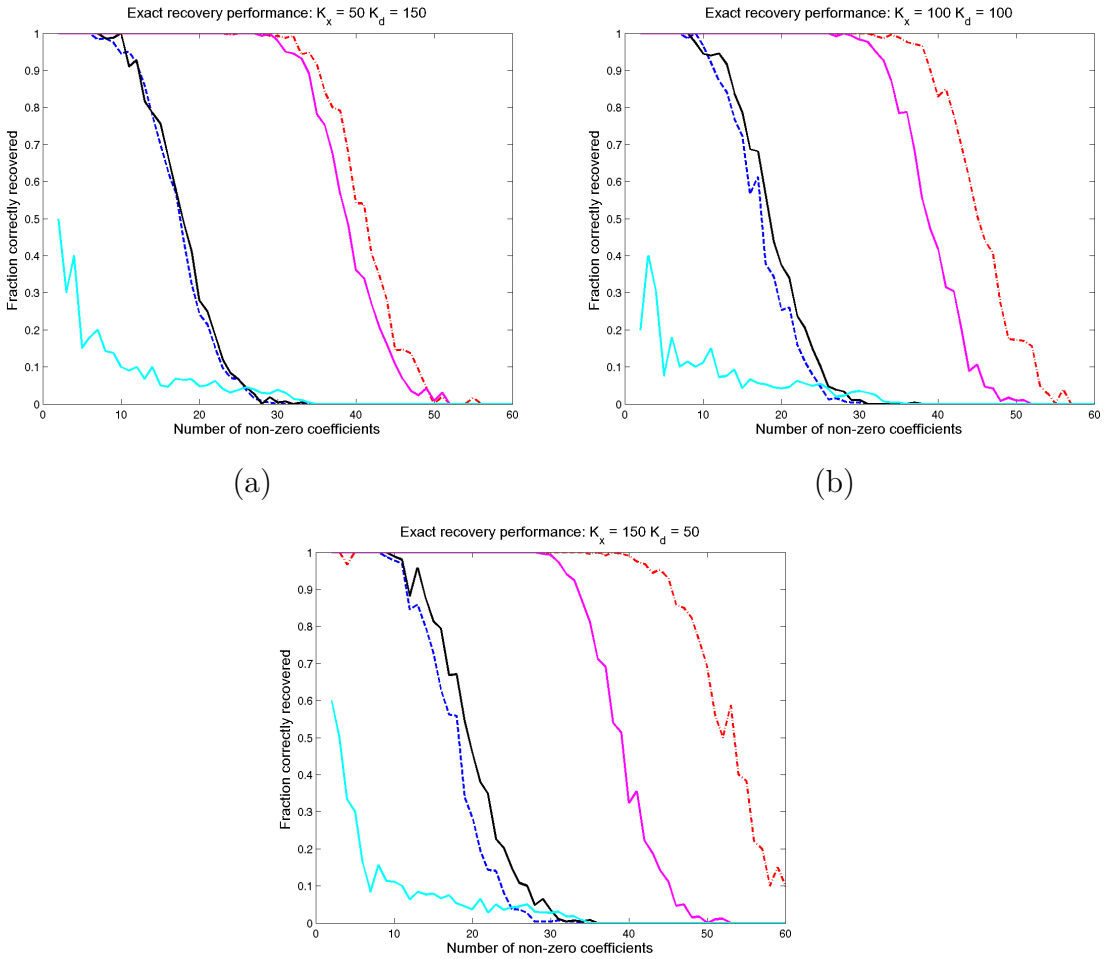


Figure 6.4: Exact recovery performance of COMB-BP, BP, COMB-OMP and OMP when $\mathbf{G}$ is obtained from a Gaussian ensemble and the non-zero coefficients are signs. (a) $K_x = 50$ and $K_d = 150$, (b) $K_x = 100$ and $K_d = 100$, (a) $K_x = 150$ and $K_d = 50$. The legend here is same as that of Figure 6.2.

Figure 6.3, it can be observed that COMB-BP and COMB-OMP algorithms exhibit lesser average RRE when compared to their unconstrained counterparts. In this case, the gap in performance between OMP and COMB-OMP is very pronounced when K_x is large. Similar behavior can be observed when non-zero coefficients are signs (Figures 6.4 and 6.5) but in general the differences in performance of the algorithms are less prominent. The experiments clearly show that COMB-BP and COMB-OMP per-

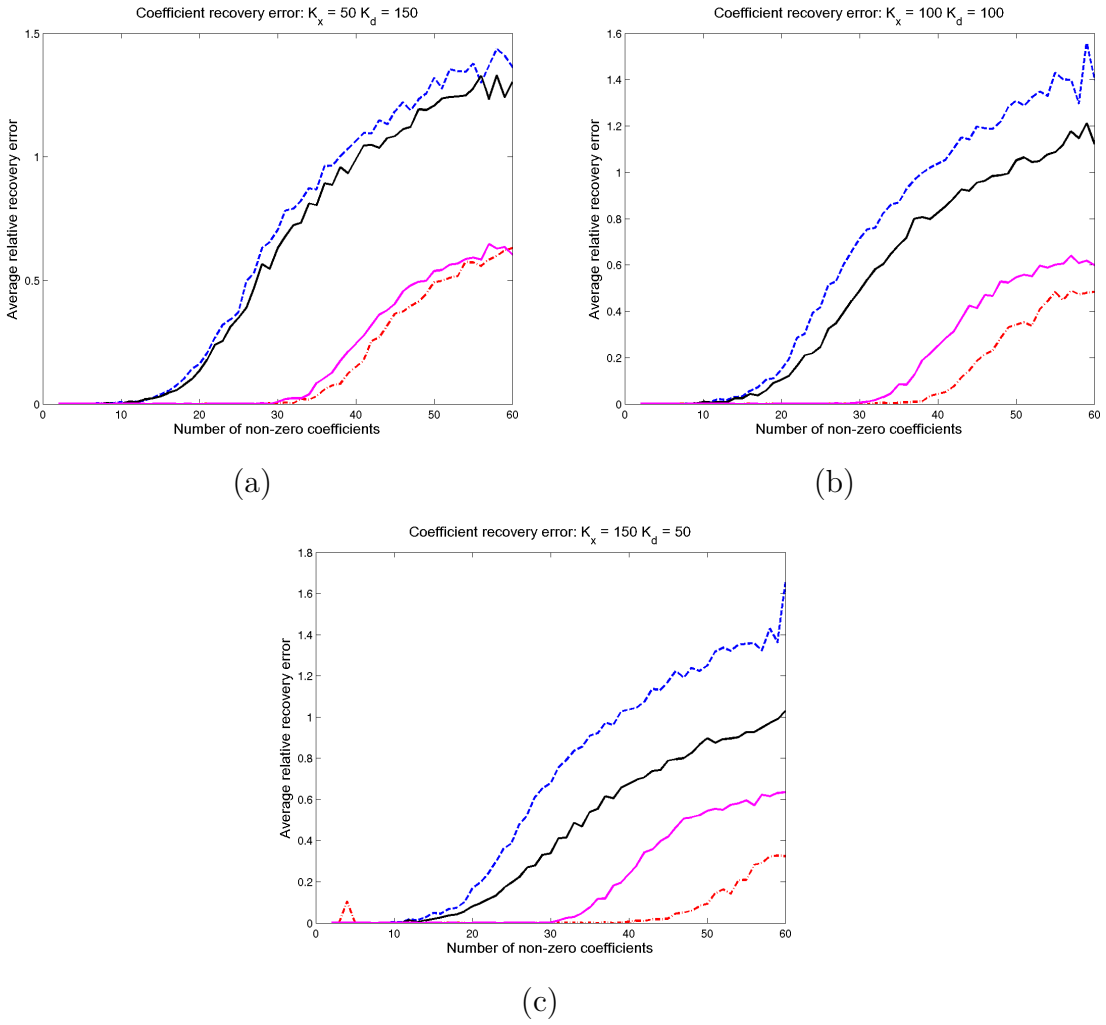


Figure 6.5: Average relative recovery error of COMB-BP, BP, COMB-OMP and OMP when $\mathbf{G}$ is obtained from a Gaussian ensemble and the non-zero coefficients are signs. (a) $K_x = 50$ and $K_d = 150$, (b) $K_x = 100$ and $K_d = 100$, (a) $K_x = 150$ and $K_d = 50$. The legend here is same as that of Figure 6.3.

form better than BP and OMP respectively, although the sparsity thresholds derived in the previous section did not point to such an improvement. This is because the deterministic sparsity bounds are generally pessimistic. Furthermore, the presence of a large non-negative component substantially improves the recovery performances of COMB-BP and COMB-OMP. The average RRE using NN-OMP and NN-BP algorithms are not shown since they are much higher than those of the other algorithms.

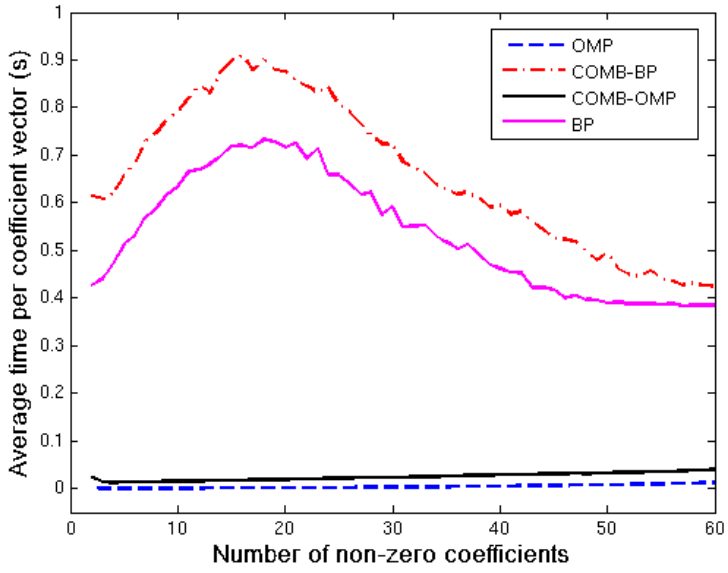


Figure 6.6: Average CPU time in seconds taken by COMB-BP, BP, COMB-OMP and OMP algorithms to recover one coefficient, when $\mathbf{G}$ is obtained from a Gaussian ensemble and the non-zero coefficients are realized from a uniform distribution with $K_x = 100$ and $K_d = 100$.

Finally, the average CPU time taken by each of the algorithms to recover a coefficient vector is computed and shown in Figure 6.6, for the case when $\mathbf{G}$ is obtained from a Gaussian ensemble, the non-zero coefficients are realized from a uniform distribution, $K_x = 100$ and $K_d = 100$. The experiments were performed using MATLAB R2010b on a 2.8GHz, Intel i7 desktop. Clearly, COMB-OMP and OMP have much lesser computational complexity when compared to COMB-BP and BP.

6.5 Conclusions

We considered the problem of recovering sparse solutions from a overcomplete linear model when the solution vector was constrained to be either completely or partially non-negative. When the solution was completely non-negative, based on the theory of polytopes, we derived conditions on the dictionary for the existence of a unique solution. In the case of combined sparse representations, we considered cases when the

coefficient support was completely known, partially known or completely unknown. When the coefficient support was completely unknown, we proposed the COMB-OMP algorithm and derived the deterministic sparsity threshold that guarantees recovery of the unique, minimum ℓ_0 norm solution. Experimental results, using dictionaries drawn from a Gaussian ensemble and non-zero coefficients realized from a uniform distribution or a equiprobable distribution of signs, showed that the COMB-BP and COMB-OMP algorithms perform better in terms of exact and approximate recovery compared to their unconstrained counterparts. A possible direction for future work is to derive probabilistic sparsity thresholds for NN-BP and COMB-BP algorithms, under appropriate assumptions on the dictionary and the coefficient vectors, that will explain the improved experimental performance of sparse recovery algorithms with non-negativity constraints when compared to their unconstrained versions.

Chapter 7

NON-LINEAR COMPRESSIVE SENSING USING KERNELS

7.1 Introduction

Certain classes of signals can be represented using a few principal components in a feature space that are obtained by a nonlinear transformations of the input signal space. Compressive sensing of such signals can be performed using the kernel trick, by transforming the linear measurements from the input space to the feature space. The overview of such a system is shown in Figure 7.1. For the case of random measurements, this framework has been proposed in [54] and results in a good recovery performance with fewer measurements compared to linear compressive sensing. When class-specific training signals are available, the measurement system can be optimized to the signals in that class. We propose such an optimized measurement system for non-linear compressive sensing that achieves a significantly higher recovery performance when compared to using random measurements. We show that the optimized measurement vectors are the training signals that have maximum projection energy on the first few kernel principal components (Figure 7.2). Simulation results with handwritten digits [153] and sculpted faces data [22] show that the recovery performance of the proposed non-linear compressive sensing framework is better than that obtained using optimized measurements with linear compressive sensing. The methods and results presented in this chapter were published in [59].

7.2 Nonlinear Compressive Sensing using Kernels

Consider the T training signals given by the matrix $\mathbf{X} = [\mathbf{x}_i]_{i=1}^{T}$ obtained from the $M-$dimensional Euclidean inner product space $\mathcal{X}$. Let us assume that the non-linear transformation to the feature space $\phi(.)$ results in signals that are centered in the feature space. We will denote a kernel matrix or vector with similarity values as $\mathbf{K}(.,.)$ with the $(i,j)^{\text{th}}$ element given as $K(\mathbf{x}_i, \mathbf{x}_j)$. We will use projective kernels

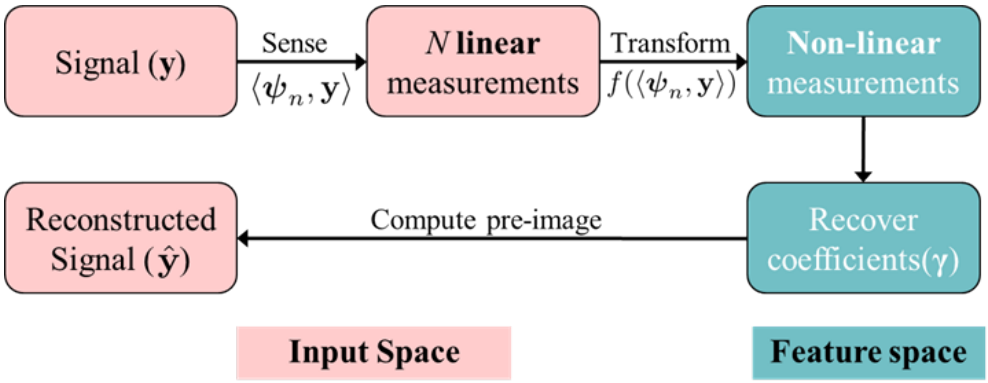


Figure 7.1: The nonlinear compressive sensing and recovery system, where N linear measurements are obtained and transformed to feature space to obtain non-linear measurements. The recovered coefficients are used to reconstruct the original data in the input space.

of the form $K(\mathbf{x}_i, \mathbf{x}_j) = f(\langle \mathbf{x}_i, \mathbf{x}_j \rangle_{\mathcal{X}})$, since they can directly transform the inner products from the input space to the feature space. Let us define an orthonormal basis set for the centered feature space computed using kernel PCA [49] as $[\mathbf{V}\ \mathbf{V}^{\perp}]$. The basis set for the L dimensions that account for most of the energy of the feature space data $\phi(\mathbf{X})$ are given by

$$\mathbf{V} = \phi(\mathbf{X})\boldsymbol{\alpha}, \tag{7.1}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^{T \times L}$ is obtained as the first L eigen vectors from the eigen decomposition of $\mathbf{K}(\mathbf{X}, \mathbf{X})$. Note that $\phi(\mathbf{X})$ denotes the matrix $[\phi(\mathbf{x_i})]_{i=1}^{T}$.

In order to perform non-linear compressive sensing in the feature space, we transform the linear measurements $\langle \boldsymbol{\psi}_n, \mathbf{y} \rangle_{\mathcal{X}}$, where $n = \{1, \cdots, N\}$ and $N \ll M$, to the feature space as $K(\boldsymbol{\psi}_n, \mathbf{y}) = f(\langle \mathbf{y}, \boldsymbol{\psi}_n \rangle_{\mathcal{X}})$. The function $f(.)$ depends only on $\langle \mathbf{y}, \boldsymbol{\psi}_n \rangle_{\mathcal{X}}$ and not on $\mathbf{y}$ or $\boldsymbol{\psi}_n$, and hence the $K(.,.)$ is referred to as a projective kernel. Let us assume that the data samples can be represented in the first L principal components in feature space. Therefore, we can represent

$$\phi(\mathbf{y}) = \mathbf{V}\boldsymbol{\gamma} + \boldsymbol{\epsilon}, \tag{7.2}$$

where $\boldsymbol{\gamma} \in \mathbb{R}^{L}$ and $\langle \boldsymbol{\epsilon}, \boldsymbol{\epsilon} \rangle_{\mathcal{F}}$ is much smaller compared to $\langle \mathbf{V}\boldsymbol{\gamma}, \mathbf{V}\boldsymbol{\gamma} \rangle_{\mathcal{F}}$. The kernel

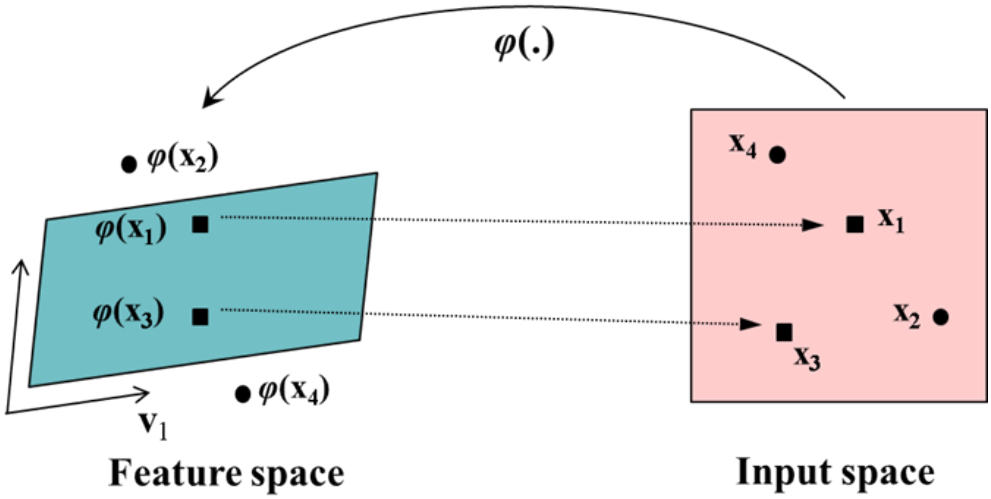


Figure 7.2: Computing optimized measurements for kernel compressive sensing. The feature samples $\phi(\mathbf{x}_1)$ and $\phi(\mathbf{x}_3)$ have the maximum projection energy in the principal feature subspace spanned by $\mathbf{V} = [\mathbf{v}_1\ \mathbf{v}_2]$. Data samples $\mathbf{x}_1$ and $\mathbf{x}_3$ are chosen as optimized measurement vectors in the input space.

measurements can be written as

$$K(\boldsymbol{\psi}_n, \mathbf{y}) = \langle \phi(\boldsymbol{\psi}_n), \mathbf{V}\boldsymbol{\gamma} \rangle_{\mathcal{F}} + \langle \phi(\boldsymbol{\psi}_n), \boldsymbol{\epsilon} \rangle_{\mathcal{F}}. \tag{7.3}$$

Including all N measurements, from (7.1), we obtain

$$\mathbf{K}(\boldsymbol{\Psi}, \mathbf{y}) = \mathbf{K}(\boldsymbol{\Psi}, \mathbf{X})\boldsymbol{\alpha}\boldsymbol{\gamma} + \mathbf{K}(\boldsymbol{\Psi}, \boldsymbol{\epsilon}), \tag{7.4}$$

where $\boldsymbol{\Psi} = [\boldsymbol{\psi}_n]_{n=1}^N$. $\boldsymbol{\gamma}$ can be estimated using a least squares procedure from (7.4). Using (7.2) and (7.1), the recovered signal in $\mathcal{F}$ can be represented as

$$\phi(\hat{\mathbf{y}}) = \phi(\mathbf{X})\boldsymbol{\alpha}\boldsymbol{\gamma}. \tag{7.5}$$

In order to actually compute $\hat{\mathbf{y}}$ from $\phi(\hat{\mathbf{y}})$, we can use either an exact or approximate preimage method since the inverse map of $\phi(.)$ is usually not defined.

In the approximate MDS-based preimage method, we first compute $\mathbf{K}(\mathbf{X}, \hat{\mathbf{y}}) = \mathbf{K}(\mathbf{X}, \mathbf{X})\boldsymbol{\alpha}\boldsymbol{\gamma}$ and $K(\hat{\mathbf{y}}, \hat{\mathbf{y}}) = \boldsymbol{\gamma}^T\boldsymbol{\alpha}^T\mathbf{K}(\mathbf{X}, \mathbf{X})\boldsymbol{\alpha}\boldsymbol{\gamma}$ using (7.5). For an arbitrary training signal $\mathbf{x}_i$, we have $\langle \mathbf{x}_i, \hat{\mathbf{y}} \rangle_{\mathcal{X}} = f^{-1}\left(K(\mathbf{x}_i, \hat{\mathbf{y}})\right)$ and $\langle \hat{\mathbf{y}}, \hat{\mathbf{y}} \rangle_{\mathcal{X}} = f^{-1}\left(K(\hat{\mathbf{y}}, \hat{\mathbf{y}})\right)$. Using

these inner products, $\|\mathbf{x}_i - \hat{\mathbf{y}}\|_{\mathcal{X}}^2$ can be computed and the P nearest training signals to $\hat{\mathbf{y}}$ can be estimated. MDS can be used to estimate the unknown signal $\hat{\mathbf{y}}$, from its nearest neighbors and their distances to $\hat{\mathbf{y}}$.

7.3 Optimized Measurements

It has been shown in [54] that compressive recovery using kernel random projections, perform substantially better than total variation or ℓ_1 minimization approaches for low to medium number of measurements in several datasets. However, optimized measurements have been shown to achieve significantly higher performance in linear compressive sensing, when compared to random projections, when the sparse representation dictionary is known. The idea is to ensure that the equivalent dictionary, which is the product of the measurement matrix and the sparse representation dictionary $\mathbf{\Psi}^T\mathbf{D}$, is as incoherent as possible. The measurement matrix $\mathbf{\Psi}$ is designed such that the Gram matrix of the equivalent dictionary, $\mathbf{D}^T\mathbf{\Psi}\mathbf{\Psi}^T\mathbf{D}$, is approximately equal to the identity matrix. Measurement vectors can be optimized in the case of kernel compressive sensing also in order to improve the recovery performance when compared to using same number of random measurement vectors.

From (7.3), it can be seen that the kernel measurements have two components in $\mathcal{F}$, one that lies in the span of $\mathbf{V}$ and one that lies in its orthogonal complement. However, as per our assumption $\phi(\mathbf{y})$ has most of its energy in the span of $\mathbf{V}$. If we design a measurement system such that $\{\phi(\boldsymbol{\psi}_n)\}_{n=1}^N$ almost entirely lies in the span of $\mathbf{V}$, we will able to perform much better recovery compared to using random measurements. Furthermore, in order to ensure that we extract maximum information, the measurement system $\phi(\mathbf{\Psi})$ should be of rank L in feature space. This problem can be formally stated as:

$$\min_{\mathbf{\Psi},\mathbf{\Omega}} \|\phi(\mathbf{\Psi}) - \mathbf{V}\mathbf{\Omega}\|_{\mathcal{F}}^2, \text{ subject to } \operatorname{rank}(\phi(\mathbf{\Psi})) = L. \tag{7.6}$$

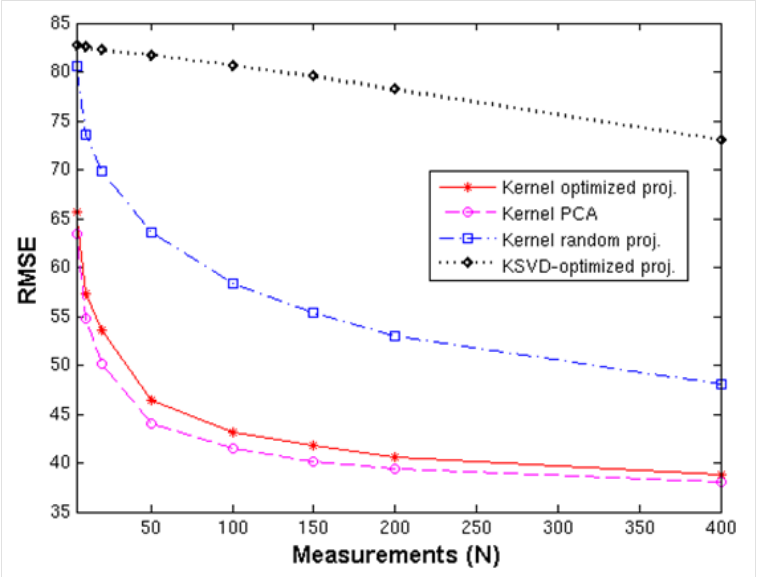
Here $\phi(\mathbf{\Psi})$ is matrix with N columns and $\mathbf{\Omega} \in \mathbb{R}^{L\times N}$. This problem has an infinite

number of exact solutions and a suitable solution can be chosen based on any regularization constraint. An important consideration is that we should be able to obtain the actual measurement system $\hat{\mathbf{\Psi}}$ from the computed $\phi(\mathbf{\Psi})$ accurately, which is not straightforward as the map $\phi(.)$ is not invertible.

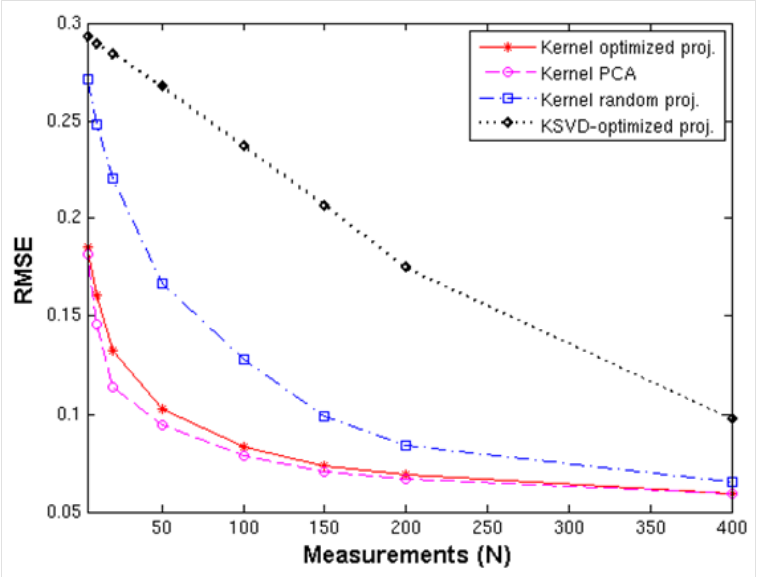
The solution to (7.6), given as $\phi(\hat{\mathbf{\Psi}}) = \mathbf{RV}$, where $\mathbf{R}$ is an orthogonal rotation and scaling matrix, is mathematically appealing. Exact preimages do not exist in general for such a $\phi(\hat{\mathbf{\Psi}})$ and compressive recovery results obtained using the approximate preimages of $\phi(\hat{\mathbf{\Psi}})$ computed using the MDS based method [93] were not encouraging. Since exact preimages can be guaranteed for the data samples $\mathbf{X}$ only, we propose the following procedure to compute the optimized measurement system. This scheme is illustrated in Figure 7.2 for a sample low dimensional scenario. Given the L principal dimensions in the feature space represented by $\mathbf{V}$, we compute the projections $\boldsymbol{\beta}_i$ of each feature sample $\phi(\mathbf{x}_i)$ on $\mathbf{V}$. The indices of N feature samples that have the highest projection energies are given by the set $\mathcal{C}$. The measurement vectors in the feature space are now given by $\phi(\hat{\mathbf{g}}_n) = \phi(\mathbf{x}_i), i = \mathcal{C}(n)$. Since every $\phi(\mathbf{x}_i)$ has an exact preimage given by $\mathbf{x}_i$, the measurement system is given by the N data samples indexed by the set $\mathcal{C}$ itself. Note that because of the rank constraint in (7.6), we choose $N > L$. By computing the rank of the matrix of projection components $[\boldsymbol{\beta}_i]_{i\in\mathcal{C}}$, we can ensure that the projections of $\{\phi(\mathbf{x}_i)\}_{i\in\mathcal{C}}$ completely span the subspace spanned by $\mathbf{V}$. The measurement system is then used to perform kernel compressive sensing and recovery is performed as described in Section 7.2.

7.4 Results

We compare the recovery performances of the non-linear kernel compressive sensing with random and optimized measurement systems, as well as with that of the optimized measurement system for linear compressive sensing. The two datasets used for testing are the MNIST handwritten digit 2 dataset [153] and the sculpted faces [22].



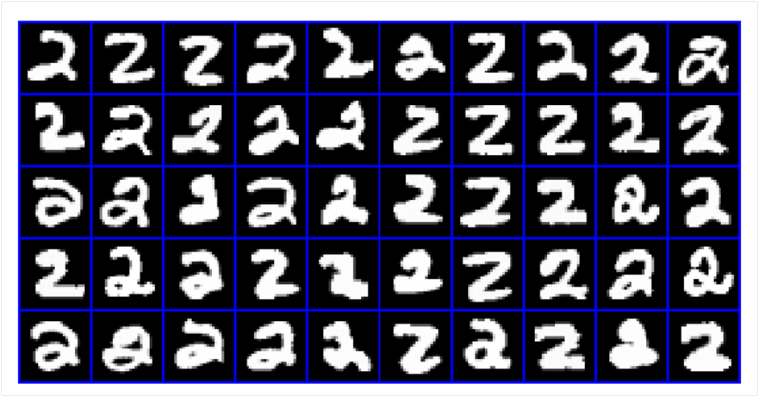
(a)



(b)

Figure 7.3: Kernel compressive recovery with random and optimized measurement systems for (a) digit 2 data, (b) sculpted faces data. Mean RMSE with projection of test data onto L kernel principal components is also shown.

The handwritten digit 2 had 5958 training images and 500 test images were randomly chosen from the test set. Each image was of size 28×28. The sculpted faces dataset had 698 images of size 64×64 of which 500 images were randomly chosen as training



(a)

(b)

Figure 7.4: The 50 optimized measurement vectors chosen from the training set for performing kernel compressive sensing ($L = 25$) (a) digit 2 data (b) sculpted faces data.

samples and the rest were used as test samples. Note that the vectorized training samples are denoted by $\mathbf{X}$ and each test sample is denoted by $\mathbf{y}$. A polynomial kernel of the form $K(\mathbf{x}_i, \mathbf{x}_j) = (c + \langle \mathbf{x}_i, \mathbf{x}_j \rangle_{\mathcal{X}})^p$ is used where $p = 5$ and c is computed as the mean of entries of the covariance of $\mathbf{X}$.

The random measurement vectors are obtained as realizations from i.i.d. Gaussian random variables of zero mean and unit variance. The optimized measurement system was computed using the procedure detailed in Section 7.3. For comparison, we also learned a dictionary using the training samples in each of the two datasets using the K-SVD algorithm and computed the optimized measurement systems for linear

compressive sensing using the procedure described in [136]. For kernel compressive sensing, the test data $\hat{\mathbf{y}}$ is recovered using the MDS-based preimage computation procedure and for linear compressive sensing we use the orthogonal matching pursuit algorithm to approximately solve for the coefficient vector $\boldsymbol{\beta}$. Figure 7.3 shows the recovery performance in terms of average root mean square error (RMSE) between the original and the recovered test samples, for the digit and sculpted faces dataset. For kernel compressive sensing, we set the number of measurements as twice the number of dominant principal components, $N = 2L$. For linear compressive sensing it was set to twice the assumed number of non-zero elements in the coefficient vector $\boldsymbol{\beta}$. For comparison, the average RMSE is calculated for test samples recovered by directly projecting on to the L kernel principal components. The proposed optimized measurement system performs significantly better than using random measurements for kernel compressive sensing as well as using optimized measurements for linear compressive sensing. It is close in performance to the direct projection using the kernel principal components.

Figure 7.4 illustrates the 50 optimized measurement vectors computed for the datasets. It can be seen that for the digit dataset, the measurement system covers a wide variety of strokes and for the sculpted faces, the measurement system covers a wide range of poses. For classes of data that can be well-represented using a few principal components in the feature space, good recovery performance in kernel compressive sensing can be obtained by using a carefully chosen set of training samples themselves as the measurement vectors.

7.5 Conclusions

We considered the problem of compressively sensing signals that are well represented in feature spaces obtained by a non-linear transformation of the input space. An optimized measurement system for performing non-linear compressive sensing was

proposed. It was shown that the measurement system can be designed using a carefully chosen set of samples obtained from the training data itself. Results with two different datasets showed that the proposed optimized measurements outperformed both the random measurement system as well as an optimized measurement system for linear compressive sensing.

Chapter 8

SUMMARY AND FUTURE WORK

The research presented in this dissertation covered a range of sparse models and their interaction with manifold, ensemble and graph-based methods. The use of the proposed methods in data representation, recovery and inference applications were analyzed. We began by focusing on the paradigm of graph embedding, and proposed a general sparse learning framework that incorporates graph embedding constraints. The applications of this framework in several machine learning applications were discussed. Following this, we shifted our interest to ensemble models and proposed algorithms to represent data using a combination of several sparse models. The utility of these models in image analysis applications were discussed. Subsequently, two models that combine traditional sparse coding framework and manifold projection for recovering data from their low-dimensional corrupted observations were proposed, and used for recovering degraded images. We then performed a theoretical study on an important class of sparse models, the non-negative representations, and proceeded to develop a new paradigm of sparse representations - the combined sparse model, where a part of the coefficient support is known to be non-negative and the rest is unconstrained. Finally, we analyzed a non-linear compressive sensing framework using kernels, that provides improved recovery performance for certain classes of data and proposed an optimized measurement system that resulted in a much improved recovery performance. A detailed summary and possible research directions for the future are presented below.

8.1 Summary

Using the paradigm of graph embedding, relationship between data can be encoded as undirected graphs. This was integrated with the traditional sparse modeling paradigm and algorithms to generate graph-embedded sparse codes were proposed.

The optimization problem is non-convex and non-differentiable, and hence methods from non-convex optimization theory were adapted to compute these codes. Since optimum sparse codes cannot be obtained without dictionaries, an alternating procedure to learn dictionaries and sparse codes was used. If discriminative graph-embedding constraints are posed, codes that provide an improved discrimination between classes can be obtained. The sparse coefficients were used in unsupervised, supervised and semi-supervised learning frameworks, with standard datasets, and results showed that the proposed algorithm performs better than several baseline procedures.

Ensemble models are a powerful class of models in machine learning, where multiple weak hypotheses are combined to obtain the final inference. We proposed to learn an ensemble of sparse models from the data using an adaptive procedure that incorporates the knowledge of degradation of data. A method to restore the data from degraded observations was also proposed. The ensemble model was used to recover compressively sensed data and results showed that better performance was obtained, when compared to using traditional learned dictionaries. In superresolution, the proposed ensemble models performed comparably to several recent sparse coding-based approaches. A notable advantage with the proposed ensemble model is that a knowledge of only the form of degradation is sufficient, and it is not necessary that we know the actual degradation operator. We also demonstrated the utility of the proposed models in unsupervised clustering.

Two generative models that combine manifold projection using examples, with sparse coding using a predefined dictionary were proposed. The manifold projection was implemented using a non-negative sparse representation of examples in the neighborhood of the test observation. The first model regularized sparse coding using manifold samples and the second model performed a direct combination of manifold projection and sparse coding. Experiments with image inpainting showed that the proposed models perform better than using predefined dictionaries with sparse cod-

ing in extreme missing/corrupted data conditions. Compressive sensing with random measurements and inpainting of standard images was performed using the second model and results show that in all cases of compressive sensing and most cases of inpainting, the proposed model performed better recovery than dictionaries learned using the state-of-the-art procedures.

The problem of recovering a combined non-negative and general sparse representation from an arbitrary dictionary with known coherence parameters was studied. Sparsity thresholds for the cases when either or both the non-negative and general coefficient supports were unknown, were derived. In particular, a greedy algorithm, the COMB-OMP, was proposed for recovering non-negative and general coefficients when both their supports are unknown and its performance was analyzed. Since the sparsity thresholds derived were pessimistic, the performance of the COMB-BP, BP, COMB-OMP and the OMP algorithms in recovering the sparse coefficient vectors was studied through simulations, and results showed that imposing non-negativity constraints aid in coefficient recovery.

Compressive sensing can be performed in a feature space instead of the input space leading to improved recovery performance for certain classes of data. Such data can be well-represented in feature spaces obtained using a non-linear transformation of the input space. Non-linear compressive sensing is performed by transforming the linear measurements from the input space to the feature space. A method to optimize the measurement system to the class of training data for non-linear compressive sensing was proposed. The optimized measurement vectors were a set of carefully chosen training vectors themselves. Significant improvement in recovery performance was obtained compared to using random measurements as well as using optimized measurements for linear compressive sensing with two different datasets.

8.2 Future Directions

Sparse codes that obey graph constraints result in similarity kernels that incorporate the class relationships, as demonstrated in Chapter 3. Although the graph is sparse, the similarity kernels have a block-diagonal structure, since the graph relationships are propagated across the whole class. In the problem of label or tag propagation, each data is assigned multiple tags and based on the similarity between the data samples and they are propagated [154, 155]. Let us assume that the data samples lie in a union of low-dimensional linear subspaces, and the tag vectors have similarities defined using a graph. The proposed approach can propagate the tag information by performing graph-embedded sparse coding, which considers both the tag similarities and sparse subspace structure of the data. In fact, the proposed approach can be used in several applications where different kinds of similarities encoded using various graphs should be fused. Apart from exploring new applications, analyzing the theoretical characteristics of the proposed graph embedded sparse modeling approach is also a promising research direction. It is well-known that samples from smooth manifolds can be represented using graph Laplacians [24], and samples from Grassmannian manifolds can be represented using ℓ_1 graphs [131]. Since we are combining both the manifold priors in our approach, it is worthwhile to understand the type of manifolds that can be well-represented using the proposed sparse coding approach. From the standpoint of implementation, computing graph-embedded codes can be performed efficiently using "gossip protocol" approaches in a distributed manner, and its convergence can be analyzed.

The ensemble sparse models proposed in Chapter 4 form a new paradigm in representing and recovering data, and hence there are interesting future research directions that exist. Ensemble representations that use ideas from *bagging* can be beneficial in the case of ultra-small datasets, where dictionary learning usually leads

to a trivial solution. In the case of ultra-large datasets, a divide-and-conquer strategy is used to reduce the complexity in dictionary learning and representation. Ensemble approaches that use 1−sparse representations are amenable to *fine-grain parallelism*, and can be efficiently implemented using *Graphical Processor Units*. As we move towards increasingly parallel architectures, these types of models promise to adapt and scale well. Furthermore, from the theoretical end, the Random Example Selection and Averaging (*RandExAv*) scheme can be analyzed to derive performance guarantees. This scheme chooses a random subset of dictionary atoms and averages the multiple 1−sparse approximations obtained. Using approaches similar to those in [125], the estimation performance of the proposed scheme with various forms of degradations can be analyzed.

Turning to the theory of sparse representations in Chapter 6, where we analyze the *deterministic* sparsity thresholds for non-negative and combined representations, the natural extension is to study the *robust* or *probabilistic* sparsity thresholds, which are much more useful in practical scenarios. The robust thresholds for general representations [66] use standard tools from Banach space probability such as *Khintchine* inequalities, based on assumptions on the sign patterns of the coefficient vector. However, for non-negative representations, these tools cannot be directly used, since the sign patterns do not change. An intuitive understanding of the problem of non-negative representations will help us identify the tools necessary to compute the robust thresholds. This can be then extended for the problem of combined representations as well. Although the mathematical details are not entirely clear, empirical experiments give us the hope that such an analysis is possible, and improvements in robust thresholds can be obtained in comparison to general representations.

Linear discriminant analysis [156] is a well-known procedure that can be used to improve separability between different classes of data and hence lead to an improved classification performance. In the non-linearly transformed feature space, this

can be extended to compute discriminant subspaces of the feature space, using the generalized discriminant analysis (GDA) procedure [92]. Since GDA is the discriminative counterpart of kernel PCA, classification can be performed by projecting the non-linearly transformed data onto the directions identified and classification can be performed. The non-linear compressive random measurements of a data sample can be used to identify its projection onto the GDA directions, computed using the training data, and this can be directly used in a classifier. Furthermore, by designing an optimized measurement system, the projections computed onto the discriminant directions can be improved. Measurement systems for representation and discrimination use a subset of samples from the training examples themselves. If K optimized measurements are obtained from the training set of T_{tr}, where $K \ll T_{tr}$ and the test set contains T_{te} samples, the complexity of computing kernel matrices for test data is $O(KT_{te})$, instead of $O(T_{tr}T_{te})$. Hence, the optimized system, by virtue of providing a good task-specific representation, can be useful in dimensionality reduction and subsequently in reducing the computational complexity of algorithms in kernel space.

REFERENCES

[1] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *Image Processing, IEEE Transactions on*, vol. 19, no. 11, pp. 2861–2873, 2010.

[2] J. Yang, J. Wright, T. Huang, and Y. Ma, "Image super-resolution as sparse representation of raw image patches," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on.* IEEE, 2008, pp. 1–8.

[3] "USPS dataset," Available at ftp://ftp.kyb.tuebingen.mpg.de/pub/bs/data/.

[4] C. Lawson and R. Hanson, *Solving Least Squares Problems.* New Jersey: Prentice Hall Inc., 1974.

[5] S. Mallat, *A Wavelet Tour of Signal Processing.* Academic: New York, 1999.

[6] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Transactions on Computers*, pp. 90–93, 1974.

[7] J. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3445–3462, Dec. 1993.

[8] M. Marcellin, M. Gormish, A. Bilgin, and M. Boliek, "An overview of JPEG-2000," *Proceedings of the IEEE Data Compression Conference*, pp. 523–541, Mar. 2000.

[9] M. J. Fadili, J. L. Starck, and F. Murtagh, "Inpainting and zooming using sparse representations," *The Computer Journal*, vol. 52, pp. 64–79, 2009.

[10] Z. Xu and J. Sun, "Image inpainting by patch propagation using patch sparsity." *IEEE Transactions on Image Processing*, vol. 19, no. 5, pp. 1153–65, May 2010.

[11] M. Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing.* Springer, 2010.

[12] J. Thiagarajan, K. Ramamurthy, and A. Spanias, "Multilevel dictionary learning for sparse representation of images," in *IEEE DSPE Workshop*, 2011, pp. 271–276.

[13] M. Aharon, M. Elad, and A. Bruckstein, "The K-SVD: an algorithm for designing of overcomplete dictionaries for sparse representation," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[14] M. Aharon and M. Elad, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Transactions on Image Processing*, vol. 15, no. 12, pp. 3736–3745, 2006.

[15] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 210–227, 2009.

[16] K. Yu, T. Zhang, and Y. Gong, "Nonlinear learning using local coordinate coding," *Advances in Neural Information Processing Systems*, vol. 22, pp. 2223–2231, 2009.

[17] J. J. Thiagarajan, K. N. Ramamurthy, P. Knee, and A. Spanias, "Sparse representations for automatic target classification in SAR images," in *Proceedings of ISCCSP*, 2010.

[18] J. J. Thiagarajan, K. N. Ramamurthy, and A. Spanias, "Sparse representations for pattern classification using learned dictionaries," *Proceedings of Twenty-eighth SGAI International Conference on Artificial Intelligence*, 2008.

[19] B. Cheng, J. Yang, S. Yan, Y. Fu, and T. Huang, "Learning with ℓ_1-graph for image analysis." *IEEE Transactions on Image Processing*, vol. 19, no. 4, pp. 858–66, 2010.

[20] R. Raina, A. Battle, H. Lee, B. Packer, and A. Ng, "Self-taught learning: Transfer learning from unlabeled data," in *Proceedings of ICML*, 2007.

[21] I. Jolliffe, *Principal component analysis*. Wiley Online Library, 2005.

[22] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction." *Science*, vol. 290, no. 5500, pp. 2319–23, 2000.

[23] S. Roweis and L. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, pp. 2323–2326, 2000.

[24] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," *Advances in Neural Information Processing Systems*, vol. 14, pp. 585–591, 2001.

[25] D. Donoho and C. Grimes, "Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data," *Proceedings of the National Academy of Sciences*, vol. 100, no. 10, pp. 5591–5596, 2003.

[26] K. Fukunaga, *Introduction to Statistical Pattern Recognition*. Academic Press, 1990.

[27] H. Chen, H. Chang, and T. Liu, "Local discriminant embedding and its variants," in *IEEE CVPR*, vol. 2, 2005, pp. 846–853.

[28] D. Cai, X. He, and J. Han, "Semi-supervised discriminant analysis," in *IEEE ICCV*, 2007.

[29] R. Polikar, "Ensemble based systems in decision making," *Circuits and Systems Magazine, IEEE*, vol. 6, no. 3, pp. 21–45, 2006.

[30] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[31] Y. Freund, R. Schapire, and N. Abe, "A short introduction to boosting," *Journal-Japanese Society For Artificial Intelligence*, vol. 14, no. 771-780, p. 1612, 1999.

[32] S. Kwok and C. Carter, *Multiple decision trees*. Basser Department of Computer Science, University of Sydney, 1988.

[33] N. Duffy and D. Helmbold, "Boosting methods for regression," *Machine Learning*, vol. 47, no. 2, pp. 153–200, 2002.

[34] A. Bruckstein, M. Elad, and M. Zibulevsky, "On the uniqueness of nonnegative sparse solutions to underdetermined systems of equations," *IEEE Transactions on Information Theory*, vol. 54, no. 11, pp. 4813–4820, 2008.

[35] D. L. Donoho and J. Tanner, "Counting the faces of randomly-projected hypercubes and orthants, with applications," *Discrete & computational geometry*, vol. 43, Apr. 2010.

[36] M. Wang, W. Xu, and A. Tang, "A unique nonnegative solution to an underdetermined system: From vectors to matrices," *IEEE Transactions on Signal Processing*, vol. 59, no. 3, pp. 1007 –1016, march 2011.

[37] D. L. Donoho and J. Tanner, "Sparse nonnegative solution of underdetermined linear equations by linear programming," *Proceedings of the National Academy of Sciences*, vol. 102, no. 3, Jun. 2005.

[38] M. Khajehnejad, A. G. Dimakis, W. Xu, and B. Hassibi, "Sparse recovery of nonnegative signals with minimal expansion," *IEEE Transactions on Signal Processing*, vol. 59, no. 1, pp. 196–208, 2011.

[39] K. N. Ramamurthy, J. J. . Thiagarajan, and A. Spanias, "Improved sparse coding using manifold projections," in *IEEE ICIP*, 2011.

[40] J. Gemmeke, T. Virtanen, and A. Hurmalainen, "Exemplar-based sparse representations for noise robust automatic speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 7, pp. 2067 –2080, 2011.

[41] M. Slawski and M. Hein, "Sparse recovery for protein mass spectrometry data," in *NIPS Workshop on Practical Application of Sparse Modeling: Open Issues and New Directions*, 2010.

[42] L. Benaroya, L. M. Donagh, F. Bimbot, and R. Gribonval, "Non negative sparse representation for wiener based source separation with a single sensor," in *Proc. IEEE ICASSP*, vol. 6. IEEE, 2003, pp. VI–613.

[43] R. He and W.-S. Zheng, "Nonnegative sparse coding for discriminative semi-supervised learning," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2011.

[44] P. O. Hoyer, "Non-negative sparse coding," in *Proc. IEEE workshop on neural networks for signal processing*, 2002, pp. 557–565.

[45] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *In NIPS*. MIT Press, 2001, pp. 556–562.

[46] P. O. Hoyer, "Non-negative matrix factorization with sparseness constraints," *Journal of Machine Learning Research*, vol. 5, pp. 1457âĂŞ–1469, 2004.

[47] D. L. Donoho and V. C. Stodden, "When does non-negative matrix factorization give a correct decomposition into parts?" in *Advances in Neural Information Processing Systems*, 2004.

[48] B. Schölkopf, A. Smola, and K.-R. Müller, "Kernel principal component analysis," *Artificial Neural Networks – ICANN*, pp. 583–588, 1997.

[49] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[50] J. Ye, Z. Zhao, and M. Wu, "Discriminative k-means for clustering," in *Advances in Neural Information Processing Systems*, 2007.

[51] S. Yan, D. Xu, B. Zhang, H. Zhang, Q. Yang, and S. Lin, "Graph embedding and extensions: A general framework for dimensionality reduction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 1, pp. 40–51, 2007.

[52] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising with block-matching and 3d filtering," in *Proc. Comput. Imaging IV SPIE Electronic Imaging*, Jan. 2006.

[53] W. Dong, X. Li, and L. Zhang, "Sparsity-based image denoising via dictionary learning and structural clustering," *Computer Vision and Pattern Recognition*, 2011.

[54] H. Qi and S. Hughes, "Using the kernel trick in compressive sensing: accurate signal recovery from fewer measurements," in *Proceedings of IEEE ICASSP*, 2011.

[55] K. N. Ramamurthy, J. J. . Thiagarajan, P. Sattigeri, and A. Spanias, "Learning dictionaries with graph embedding constraints," in *Proc. Asilomar Conference on Sig., Sys., and Comp.*, 2012.

[56] K. N. Ramamurthy, J. J. Thiagarajan, and A. Spanias, "Boosted dictionaries for image restoration based on sparse representations," in *IEEE ICASSP (accepted)*, 2013.

[57] K. N. Ramamurthy, J. J. Thiagarajan, P. Sattigeri, and A. Spanias, "Ensemble sparse models for image analysis and restoration," *IEEE Transactions on Image Processing*, 2013 (submitted). [Online]. Available: http://arxiv.org/abs/1302.6957

[58] K. N. Ramamurthy, J. J. Thiagarajan, and A. Spanias, "Recovering non-negative and combined sparse representations," *Digital Signal Processing*, 2013 (Submitted).

[59] K. Ramamurthy and A. Spanias, "Optimized measurements for kernel compressive sensing," in *Proc. Asilomar Conference on Signals, Systems and Computers*, 2011.

[60] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society*, vol. 67, no. 2, pp. 301–320, 2005.

[61] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 68, no. 1, pp. 49–67, 2005.

[62] P. Kuppinger, G. Durisi, and H. Bölcskei, "Uncertainty relations and sparse signal recovery for pairs of general signal sets," *IEEE Transactions on Information Theory*, vol. PP, no. 99, p. 1, 2011.

[63] C. Studer, P. Kuppinger, G. Pope, and H. Bolcskei, "Recovery of sparsely corrupted signals," *IEEE Transactions on Information Theory*, vol. 58, no. 5, pp. 3115–3130, 2012.

[64] D. L. Donoho and M. Elad, "Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ_1 minimization." *Proceedings of the National Academy of Sciences of the United States of America*, vol. 100, no. 5, pp. 2197–202, Mar. 2003.

[65] D. Donoho, "Neighborly polytopes and sparse solution of underdetermined linear equations," Stanford University, Tech. Rep., 2005.

[66] J. Tropp, "On the conditioning of random subdictionaries," *Applied and Computational Harmonic Analysis*, vol. 25, no. 1, pp. 1–24, 2008.

[67] D. Donoho and J. Tanner, "Precise undersampling theorems," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 913–924, 2010.

[68] D. Donoho, "Neighborly polytopes and sparse solutions of underdetermined linear equations," Stanford University, Tech. Rep., 2005.

[69] J. D. Blanchard, C. Cartis, J. Tanner, and A. Thompson, "Phase transitions for greedy sparse approximation algorithms," *Applied and Computational Harmonic Analysis*, vol. 30, no. 2, pp. 188–203, 2011.

[70] A. Maleki and D. L. Donoho, "Freely available, optimally tuned iterative thresholding algorithms for compressed sensing," in *Proc. Signal Processing with Adaptive Sparse Structured Representations*, 2009.

[71] K. Engan, S. O. Aase, and J. H. Husoy, "Method of optimal directions for frame design," in *Proceedings of IEEE ICASSP*, 1999.

[72] R. Rubinstein, A. Bruckstein, and M. Elad, "Dictionaries for sparse representation modeling," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1045–1057, 2010.

[73] L. Bar and G. Sapiro, "Hierarchical dictionary learning for invariant classification," in *IEEE ICASSP*, 2010, pp. 3578–3581.

[74] J. Mairal, G. Sapiro, and M. Elad, "Multiscale sparse image representation with learned dictionaries," in *Proceedings of IEEE ICIP*, 2007.

[75] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *Journal of Machine Learning Research*, vol. 11, no. 1, pp. 19–60, 2009.

[76] J. J. Thiagarajan, K. N. Ramamurthy, and A. Spanias, "Shift-invariant sparse representation of images using learned dictionaries," *Proc. IEEE Workshop on MLSP*, pp. 145–150, 2008.

[77] R. Vidal, Y. Ma, and S. Sastry, "Generalized principal component analysis (GPCA)," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 12, pp. 1945–1959, 2005.

[78] J. R. Munkres, *Elementary differential topology*. Princeton University Press, 1966.

[79] E. Saucan, E. Appleboim, and Y. Zeevi, "Sampling and reconstruction of surfaces and higher dimensional manifolds," *Journal of Mathematical Imaging and Vision*, vol. 30, no. 1, pp. 105–123, 2008.

[80] G. Carlsson, T. Ishkhanov, V. De Silva, and A. Zomorodian, "On the local behavior of spaces of natural images," *International Journal of Computer Vision*, vol. 76, no. 1, pp. 1–12, 2008.

[81] X. Hei and P. , "Locality preserving projections," *Advances in Neural Information Processing Systems*, vol. 16, pp. 153–160, 2004.

[82] Y. Jia, F. Nie, and C. Zhang, "Trace ratio problem revisited," *IEEE Transactions on Neural Networks*, vol. 20, no. 4, pp. 729–735, 2009.

[83] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural computation*, vol. 4, no. 1, pp. 1–58, 1992.

[84] R. Maclin and D. Opitz, "Popular ensemble methods: An empirical study," *arXiv preprint arXiv:1106.0257*, 2011.

[85] L. Hansen and P. Salamon, "Neural network ensembles," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 12, no. 10, pp. 993–1001, 1990.

[86] T. Dietterich, "Ensemble methods in machine learning," *Multiple classifier systems*, pp. 1–15, 2000.

[87] E. Bauer and R. Kohavi, "An empirical comparison of voting classification algorithms: Bagging, boosting, and variants," *Machine learning*, vol. 36, no. 1, pp. 105–139, 1999.

[88] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting," *The annals of statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[89] J. Zhu, S. Rosset, H. Zou, and T. Hastie, "Multi-class adaboost," *Statistics and its Interface*, vol. 2, pp. 349–360, 2009.

[90] T. G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," *Machine learning*, vol. 40, no. 2, pp. 139–157, 2000.

[91] M. Girolami, "Mercer kernel-based clustering in feature space," *IEEE Transactions on Neural Networks*, vol. 13, pp. 780–784, 2002.

[92] G. Baudat and F. Anouar, "Generalized discriminant analysis using a kernel approach," *Neural Computation*, vol. 12, pp. 2385–2404, October 2000.

[93] J. Kwok and I. Tsang, "The pre-image problem in kernel methods." *IEEE Transactions on Neural Networks*, vol. 15, no. 6, pp. 1517–25, 2004.

[94] P. Honeine and R. Cédric, "Preimage problem in kernel-based machine learning," *IEEE Signal Processing Magazine*, vol. 28, pp. 77–88, 2011.

[95] D. Huang, Y. Tian, and F. DelaTorre, "Local isomorphism to solve the pre-image problem in kernel methods," in *Proc. IEEE CVPR*, 2011.

[96] J. Nocedal and S. Wright, *Numerical optimization*. Springer Verlag, 1999.

[97] H. Lee, A. Battle, R. Raina, and A. Ng, "Efficient sparse coding algorithms," *Advances in Neural Information Processing Systems*, vol. 19, p. 801, 2007.

[98] S. Gao, I. Tsang, L. Chia, and P. Zhao, "Local features are not lonely–Laplacian sparse coding for image classification," in *Proc. IEEE CVPR*, 2010, pp. 3555–3561.

[99] M. Zheng, J. Bu, C. Chen, C. Wang, L. Zhang, G. Qiu, and D. Cai, "Graph regularized sparse coding for image representation," *IEEE Transactions on Image Processing*, vol. 20, no. 5, pp. 1327–1336, 2011.

[100] J. Thiagarajan, K. Ramamurthy, P. Sattigeri, and A. Spanias, "Supervised local sparse coding of sub-image features for image retrieval," in *Proc. IEEE ICIP*, 2012.

[101] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *The Annals of Statistics*, vol. 32, no. 2, pp. 407–499, 2004.

[102] A. L. Yuille and A. Rangarajan, "The concave-convex procedure," *Neural Computation*, vol. 15, no. 4, pp. 915–936, 2003.

[103] Y.-F. Guo, S.-J. Li, J.-Y. Yang, T.-T. Shu, and L.-D. Wu, "A generalized foley–sammon transform based on generalized fisher discriminant criterion and its application to face recognition," *Pattern Recognition Letters*, vol. 24, no. 1, pp. 147–158, 2003.

[104] H. Wang, S. Yan, D. Xu, X. Tang, and T. Huang, "Trace ratio vs. ratio trace for dimensionality reduction," in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on.* IEEE, 2007, pp. 1–8.

[105] A. Y. Ng, M. I. Jordan, Y. Weiss *et al.*, "On spectral clustering: Analysis and an algorithm," *Advances in Neural Information Processing Systems*, vol. 2, pp. 849–856, 2002.

[106] "UCI machine learning repository," Available at http://archive.ics.uci.edu/ml/datasets.html.

[107] S. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, 1993.

[108] J. A. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Transactions on Information Theory*, vol. 50, no. 10, pp. 2231–2242, October 2004.

[109] S. Cotter, R. Adler, R. Rao, and K. Kreutz-Delgado, "Forward sequential algorithms for best basis selection," in *IEE Proceedings - Vision, Image and Signal Processing*, vol. 146, no. 5. IET, 1999, pp. 235–244.

[110] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Review*, vol. 43, no. 1, pp. 129–159, 2001.

[111] I. F. Gorodnitsky and B. D. Rao, "Sparse signal reconstruction from limited data using FOCUSS: A re-weighted norm minimization algorithm," *IEEE Transactions on Signal Processing*, vol. 45, no. 3, pp. 600–616, 1997.

[112] M. Elad, "Why simple shrinkage is still relevant for redundant representations?" *IEEE Transactions on Information Theory*, vol. 52, no. 12, pp. 5559 –5569, December 2006.

[113] M. Elad, B. Matalon, J. Shtok, and M. Zibulevsky, "A wide-angle view at iterated shrinkage algorithms," in *SPIE (Wavelet XII)*, 2007.

[114] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online dictionary learning for sparse coding," in *Proc. ICML*, 2009, pp. 689–696.

[115] H. Lee, A. Battle, R. Raina, and A. Ng, "Efficient sparse coding algorithms," *Advances in Neural Information Processing Systems*, vol. 19, p. 801, 2007.

[116] K. Huang and S. Aviyente, "Sparse representation for signal classification," in *Proc. of Advances in Neural Information Processing Systems*, 2006.

[117] I. Ramirez, P. Sprechmann, and G. Sapiro, "Classification and clustering via dictionary learning with structured incoherence and shared features," in *IEEE CVPR*, 2010, pp. 3501–3508.

[118] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *IEEE CVPR*, 2009, pp. 1794–1801.

[119] G. Yu, G. Sapiro, and S. Mallat, "Image modeling and enhancement via structured sparse model selection," in *Proc. of IEEE ICIP*, 2010, pp. 1641 –1644.

[120] Q. Zhang and B. Li, "Discriminative K-SVD for dictionary learning in face recognition," in *IEEE CVPR*, 2010.

[121] J. J. Thiagarajan and A. Spanias, "Learning dictionaries for local sparse coding in image classification," in *Proc. of Asilomar SSC*, 2011.

[122] J. J. Thiagarajan, K. N. Ramamurthy, P. Sattigeri, and A. Spanias, "Supervised local sparse coding of sub-image features for image retrieval," in *IEEE ICIP*, 2012.

[123] W. Zhang, A. Surve, X. Fern, and T. Dietterich, "Learning non-redundant codebooks for classifying complex objects," in *Proc. ICML*, 2009, pp. 1241–1248.

[124] J. Wang, Y. Li, Y. Zhang, H. Xie, and C. Wang, "Boosted learning of visual word weighting factors for bag-of-features based medical image retrieval," in *International Conference on Image and Graphics*, 2011, pp. 1035–1040.

[125] M. Elad and I. Yavneh, "A plurality of sparse representations is better than the sparsest one alone," *IEEE Transactions on Information Theory*, vol. 55, no. 10, pp. 4701–4714, 2009.

[126] "ScSR - matlab codes for image super-resolution," Available at http://www.ifp.illinois.edu/∼jyang29/resources.html.

[127] "Berkeley segmentation dataset," Available at http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/.

[128] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii, "Scalable k-means++," *Proceedings of the VLDB Endowment*, vol. 5, no. 7, pp. 622–633, 2012.

[129] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proc. ACM-SIAM symposium on Discrete algorithms*, 2007, pp. 1027–1035.

[130] D. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[131] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *arXiv preprint arXiv:1203.1005*, 2012.

[132] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 1.21," http://cvxr.com/cvx, October 2010.

[133] O. G. Guleryuz, "Nonlinear approximation based image recovery using adaptive sparse reconstructions and iterated denoising: Part I - theory," *IEEE Transactions on Image Processing*, vol. 15, pp. 539–554, 2006.

[134] ——, "Nonlinear approximation based image recovery using adaptive sparse reconstructions and iterated denoising: Part II - adaptive algorithms," *IEEE Transactions on Image Processing*, vol. 15, pp. 555–571, 2006.

[135] R. Baraniuk, "Compressive sensing [lecture notes]," *IEEE Signal Processing Magazine*, vol. 24, no. 4, pp. 118–121, 2007.

[136] J. Duarte-Carvajalino and G. Sapiro, "Learning to sense sparse signals: simultaneous sensing matrix and sparsifying dictionary optimization," *IEEE Transactions on Image Processing*, vol. 18, no. 7, pp. 1395–1408, 2009.

[137] M. Elad, "Optimized projections for compressed sensing," *IEEE Transactions on Signal Processing*, vol. 1, no. 1, pp. 1–22, 2007.

[138] E. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Inform. Theory*, vol. 52, no. 2, pp. 489–509, Feb. 2006.

[139] J. Haupt and R. Nowak, "Signal reconstruction from noisy random projections," *IEEE Transactions on Information Theory*, vol. 52, no. 9, pp. 4036 – 4048, 2006.

[140] D. Takhar *et.al.*, "A compressed sensing camera: New theory and an implementation using digital micromirrors," in *Proc. Comput. Imaging IV SPIE Electronic Imaging*, Jan. 2006.

[141] J. Tropp and A. Gilbert, "Signal recovery from partial information via orthogonal matching pursuit," 2005. [Online]. Available: http://www.math.lsa.umich.edu/∼annacg/papers/TG05-Signal-Recovery.pdf

[142] R. Baraniuk, M. Davenport, R. DeVore, and M. Wakin, "A Simple Proof of the Restricted Isometry Property for Random Matrices," *Constructive Approximation*, vol. 28, no. 3, pp. 253–263, 2008.

[143] M. Stojnic, W. Xu, and B. Hassibi, "Compressed sensing - probabilistic analysis of a null-space characterization," in *Proc. IEEE International Conference on Acoustic, Speech and Signal Processing*, 2008.

[144] J. Bardsley, "Covariance-preconditioned iterative methods for nonnegatively constrained astronomical imaging," *SIAM Journal on Matrix Analysis and Applications*, vol. 27, no. 4, pp. 1184–1197, 2006.

[145] D. Donoho, I. Johnstone, and J. Hoch, "Maximum entropy and the nearly black object," *Journal of the Royal Statistical Society*, vol. 54, no. 1, pp. 41–81, 1992.

[146] J. A. Tropp, "Topics in Sparse Approximation," Ph.D. dissertation, University of Texas at Austin, 2004.

[147] M. Slawski and M. Hein, "Robust sparse recovery with non-negativity constraints," in *Workshop on Signal Processing with Adaptive Sparse Structured Representations*, 2011.

[148] C. Studer and R. G. Baraniuk, "Stable restoration and separation of approximately sparse signals," *arXiv preprint arXiv:1107.0420*, 2011.

[149] A. Berman and N. Shaked-Monderer, *Completely positive matrices*. World Scientific, 2003.

[150] J. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Transactions on Information Theory*, vol. 50, no. 10, pp. 2231–2242, 2004.

[151] E. Kreyszig, *Introductory Functional Analysis With Applications*. Wiley, John & Sons, Incorporated, Mar. 1989.

[152] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge University Press, 1990.

[153] "The MNIST database of handwritten digits," Available online at http://yann.lecun.com/exdb/mnist/.

[154] M. Guillaumin, T. Mensink, J. Verbeek, and C. Schmid, "Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation," in

Computer Vision, 2009 IEEE 12th International Conference on. IEEE, 2009, pp. 309–316.

[155] C. Wang, S. Yan, L. Zhang, and H.-J. Zhang, "Multi-label sparse coding for automatic image annotation," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on.* IEEE, 2009, pp. 1643–1650.

[156] C. M. Bishop, *Pattern Recognition and Machine Learning.* Springer, 2007.

APPENDIX A

THE COMB-OMP ALGORITHM FOR GREEDY PURSUIT OF COMBINED SPARSE REPRESENTATIONS

Goal

Recover the $ML0$ solution from $\mathbf{y} = \mathbf{G}\boldsymbol{\delta}$ such that $\mathbf{I}_{\bar{\mathcal{X}}}\boldsymbol{\delta} \geq \mathbf{0}$.

Input

$\mathbf{y}$, the input vector.

$\mathbf{G} = [\mathbf{X} \quad \mathbf{D}]$, the combined dictionary.

T, the desired number of iterations.

ϵ, error tolerance.

Initialization

- Iteration count, $t = 0$.

- Solution, $\boldsymbol{\delta}_t = 0$.

- Residual, $\mathbf{r}_t = \mathbf{y} - \mathbf{G}\boldsymbol{\delta}_t = \mathbf{y}$.

- Active coefficient supports, $\mathcal{X}_t = \{\}$, $\mathcal{D}_t = \{\}$, $\mathcal{G}_t = \{\}$.

- All coefficient supports, $\bar{\mathcal{X}} = \{i\}_{i=1}^{K_x}$, $\bar{\mathcal{D}} = \{i\}_{i=K_x+1}^{K_g}$, $\bar{\mathcal{G}} = \bar{\mathcal{X}} \cup \bar{\mathcal{D}}$.

- Non-active coefficient supports, $\mathcal{X}_t^c = \bar{\mathcal{X}}$, $\mathcal{D}_t^c = \bar{\mathcal{D}}$, $\mathcal{G}_t^c = \bar{\mathcal{G}}$.

Algorithm

Loop until $t \leq T$ OR $\|\mathbf{r}_t\|_2 > \epsilon$

- **Compute correlations:**

  $\pi(i) = \frac{\mathbf{r}_t^T \mathbf{g}_i}{\|\mathbf{g}_i\|_2}$ for $1 \leq i \leq K_g$.

- **Update support:**

  $\hat{i} = \underset{i \in \mathcal{X}_t^c}{\operatorname{argmax}} [\pi(i)]^+$.

  $\hat{j} = \underset{j \in \mathcal{D}_t^c}{\operatorname{argmax}} |\pi(j)|$.

  $\hat{k} = \operatorname{argmax}([\pi(\hat{i})]^+, |\pi(\hat{j})|)$.

  If $\hat{k} \in \mathcal{X}_t^c$, then $\mathcal{X}_{t+1} = \mathcal{X}_t \cup \{\hat{k}\}$, else $\mathcal{D}_{t+1} = \mathcal{D}_t \cup \{\hat{k}\}$.

$\mathcal{G}_{t+1} = \mathcal{X}_{t+1} \cup \mathcal{D}_{t+1}$.

- **Update solution:**

$\boldsymbol{\delta}_{t+1} = \text{argmin}_{\boldsymbol{\delta}} \|\mathbf{y} - \mathbf{G}\boldsymbol{\delta}\|_2$ subject to $\mathit{support}(\boldsymbol{\delta}) = \mathcal{G}_{t+1}$, $\mathbf{I}_{\mathcal{X}_t}\boldsymbol{\delta} \geq \mathbf{0}$.

- **Update residual:** $\mathbf{r}_{t+1} = \mathbf{y} - \mathbf{G}\boldsymbol{\delta}_{t+1}$.

- **Update support sets:**

$\mathcal{G}^c_{t+1} = \bar{\mathcal{G}} - \mathcal{G}_{t+1}, \mathcal{X}^c_{t+1} = \bar{\mathcal{X}} - \mathcal{X}_{t+1}, \mathcal{D}^c_{t+1} = \bar{\mathcal{D}} - \mathcal{D}_{t+1}$.

- **Update iteration count:** $t = t + 1$.

end

Debias to compute final $\boldsymbol{\delta}$:

$\boldsymbol{\delta}_t = \text{argmin}_{\boldsymbol{\delta}} \|\mathbf{y} - \mathbf{G}\boldsymbol{\delta}\|_2$ subject to $\mathit{support}(\boldsymbol{\delta}) = \mathcal{G}_t$, $\mathbf{I}_{\bar{\mathcal{X}}}\boldsymbol{\delta} \geq \mathbf{0}$.