SPARSE MODELING FOR HIGH-DIMENSIONAL MULTI-MANIFOLD DATA ANALYSIS

by

Ehsan Elhamifar

A dissertation submitted to The Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy.

Baltimore, Maryland

October, 2012

Abstract

High-dimensional data are ubiquitous in many areas of science and engineering, such as machine learning, signal and image processing, computer vision, pattern recognition, bioinformatics, etc. Often, high-dimensional data are not distributed uniformly in the ambient space; instead they lie in or close to a union of low-dimensional manifolds. Recovering such low-dimensional structures in the data helps to not only significantly reduce the computational cost and memory requirements of algorithms that deal with the data, but also reduce the effect of the high-dimensional noise in the data and improve the performance of inference and learning tasks.

There are three fundamental tasks related to the multi-manifold data: clustering, dimensionality reduction, and classification. While the area of machine learning has seen great advances in these areas, the applicability of current algorithms are limited due to several challenges. First, in many problems, manifolds are spatially close or even intersect, while existing methods work only when manifolds are sufficiently separated. Second, most algorithms require to know the dimensions or the number of manifolds a priori, while in real-world problems such quantities are often unknown.

ABSTRACT

Third, most existing algorithms have difficulty in effectively dealing with data nuisances, such as noise, outliers, and missing entries, as well as manifolds of different intrinsic dimensions.

In this thesis, we present new frameworks based on sparse representation techniques for the problems of clustering, dimensionality reduction and classification of multi-manifold data that effectively address the aforementioned challenges. The key idea behind the proposed algorithms is what we call the self-expressiveness property of the data. This property states that in an appropriate dictionary formed from the given data points in multiple manifolds, a sparse representation of a data point corresponds to selecting other points from the same manifold. Our goal is then to search for such sparse representations and use them in appropriate frameworks to cluster, embed, and classify multi-manifold data. We propose sparse optimization programs to find such desired representations and develop theoretical guarantees for the success of the proposed algorithms. By extensive experiments on synthetic and real data, we demonstrate that the proposed algorithms significantly improve the state-of-the-art results.

Dedication

To my mother, Effat, for her endless love, patience, support and encouragement.

Contents

List of Tables

LIST OF TABLES

List of Figures

Chapter 1

Introduction

1.1 Multi-manifold data

High-dimensional data are ubiquitous in many areas of science and engineering, such as machine learning, signal and image processing, computer vision, pattern recognition, bioinformatics, etc. Images consist of billions of pixels, videos can have millions of frames, text and web documents are associated with tens of thousands of features, and DNA microarray data represent the expression levels of thousands of genes. This high-dimensionality of the data not only increases the computational time and memory requirements of algorithms, but also adversely affects their performance due to the effect of the noise and insufficient number of samples with respect to the ambient space dimension, hence, the so called the "curse of dimensionality" [9].

However, high-dimensional data are not often distributed uniformly in the ambient

space, instead they lie in or close to low-dimensional manifolds. This is mainly due to the fact that real data are often generated or captured by processes or physical systems that have only a few degrees of freedom. For instance, images of an object or a face captured under varying illumination can be characterized by a few degrees of freedom of the light source, or videos of a static scene captured by a moving camera can be characterized by a few degrees of freedom corresponding to the motion parameters of the camera. In fact, recovering such low-dimensional structures in the data helps to not only significantly reduce the computational cost and memory requirements of algorithms that deal with the data, but also reduce the effect of the high-dimensional noise in the data and improve the performance of inference, learning, and recognition tasks.

In many real-world problems, we are dealing with high-dimensional data across multiple classes or categories where the data in each class lie in or close to a manifold. As a result, the collection of data points lie in a union of low-dimensional manifolds. Images of multiple classes of objects or scenes, videos of different activities, text and web data of different subjects, and biomedical data corresponding to different diseases can be modeled as lying in a union of manifolds. Roughly speaking, each manifold can be characterized by a mapping from a low-dimensional input space (latent space) to a high-dimensional output space (ambient space), see Figure 1.1. This mapping can be in general nonlinear leading to nonlinear manifolds, but it can also be linear leading to flat manifolds or subspaces. In fact, subspaces correspond to an important class of

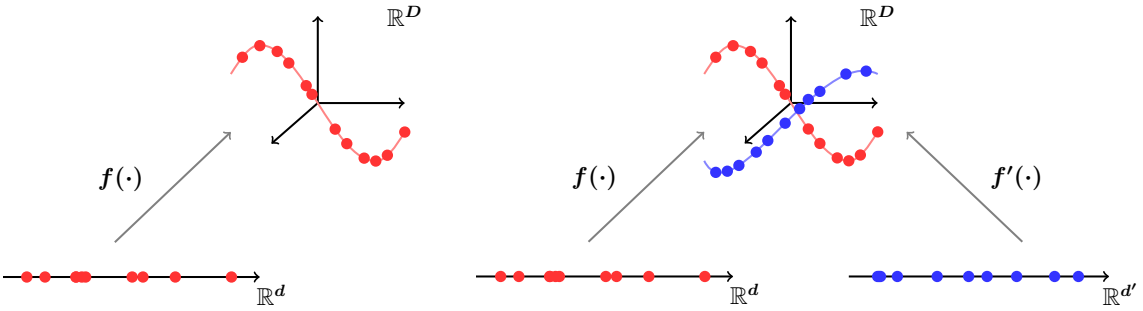


Figure 1.1: Left: a manifold is in general a mapping from a low-dimensional latent space to the high-dimensional ambient space. Right: In many problems, data across multiple classes lie in a union of manifolds.

manifolds that model well the distribution of the data in many real-world problems. Feature trajectories of multiple moving objects in a video [102], face images of multiple subjects captured under varying illumination [6], multiple instances of hand-written digits with different rotations, translations, and thicknesses [61], and human body sensor measurements corresponding to different activities [119] can be well modeled as lying in a union of low-dimensional subspaces of the ambient space.

There are three fundamental tasks related to the multi-manifold data: *clustering*, *dimensionality reduction* (*embedding*), and *classification*. While the area of machine learning has seen great advances in these areas, the applicability of current algorithms are limited due to several challenges. First, in many problems, manifolds are spatially close or even intersect, while current methods work only when manifolds are sufficiently separated. Second, most existing methods require to know the dimensions or the number of manifolds a priori, while in real-world problems such quantities are often unknown. Third, most existing algorithms have difficulty in effectively dealing with data nuisances, such as noise, outliers, and missing entries, as well as manifolds

of different intrinsic dimensions.

The goal of this thesis is to develop theoretically correct efficient algorithms for clustering, dimensionality reduction, and classification of multi-manifold data that can effectively address the aforementioned challenges. Next, we describe each of the three tasks and discuss existing approaches and their limitations.

1.1.1 Clustering

Given a collection of data points lying in multiple manifolds, the goal of clustering is to separate the data according to their underlying manifolds. In fact, clustering is one of the most fundamental problems in data analysis that serves as the first step for many important tasks such as learning, classification, recognition and inference. For instance, once we separate the data according to the manifolds they lie in, we can reduce the dimension of the data in each manifold (dimensionality reduction), recover the underlying mapping of each manifold (manifold learning), find the underlying manifold of a given query (classification), or perform other tasks that take advantage of the structural relationships of the data in a single manifold.

Depending on whether data lie in linear or nonlinear manifolds, clustering approaches are different in general.

1.1.1.1 Subspace clustering

Clustering of data in multiple subspaces, referred to as subspace clustering, is an important problem that finds numerous applications in image processing, e.g., image representation and compression [63], and computer vision, e.g., image segmentation [120], motion segmentation [25, 66], and temporal video segmentation [111]. As data in a subspace can be distributed arbitrarily and not around a centroid, standard central clustering methods [34] that take advantage of the spatial proximity of data in each cluster are not applicable in general to subspace clustering.

In fact, a variety of algorithms have been proposed for clustering of data in multiple subspaces [109] that are based on algebraic, statistical, or spectral clustering approaches (see Chapter 3 for a review). However, such methods often cannot effectively deal with important real-world situations, such as subspaces that intersect or have different dimensions, data points that are corrupted by noise and outliers or have missing entries. In addition, most existing algorithms require to know a priori the dimensions or the number of subspaces.

Spectral clustering-based approaches are among the most successful classes of subspace clustering algorithms. Such methods try to build a similarity graph whose nodes correspond to data points and whose edge weights represent similarity values between data points. The key problem is then to find a good similarity measure between data points such that points in the same subspace have high similarities and points in different subspaces have low similarities. Clustering of data into subspaces

is then obtained by finding groups of nodes (components) in the similarity graph that have high inter-component and low intra-component similarities.

Local spectral clustering-based approaches [55, 118, 122, 124] use local information around each point to compute a similarity between pairs of points. However, they have difficulties dealing with data points near the intersection of two subspaces, because the neighborhood of a point can contain points from different subspaces. In addition, they are sensitive to the right choice of the neighborhood size for computing the local information at each point. On the other hand, global spectral clustering-based approaches such as [22] try to resolve these issues by building better similarities between data points using global information. However, they often need to know a priori the number and the dimensions of the subspaces and typically assume that the subspaces have the same dimension. More importantly, their complexity of computing similarities grows exponentially as the dimensions of the subspaces increase.

1.1.1.2 Nonlinear manifold clustering

In general, data may lie in a union of nonlinear manifolds. Since in this case, there is no global linear relationship among the data points in the same subspace, standard subspace clustering algorithms are not in general applicable. In fact, manifolds can be very close to each other and they can have arbitrary dimensions, curvature and sampling, which make the manifold clustering problem be very challenging.

To cluster data in multiple manifolds, most existing algorithms assume that man-

Table 1.1: Considering all pairs of subjects in the Extended Yale B face dataset, the table shows the average percentage of images whose K nearest neighbors contain images from another subject. Note that for 6.0% of images, the nearest neighbor comes from another subject.

K	1	2	3	4	5	6	7	8	9	10
	6.0%	13.4%	22.5%	30.9%	38.9%	46.6%	53.8%	59.4%	64.2%	68.2%

ifolds are densely sampled and sufficiently separated [2, 55, 56, 87]. They build a similarity graph by connecting each point to its few nearest neighbors. As a result, ideally, the points in the same manifold get connected to each other while there are no connections among points in different manifolds. Hence, clustering of data is obtained by separating the components of the graph.

However, it is not often the case that manifolds are densely sampled or well separated. For instance, in a real dataset of face images of multiple subjects, it is often the case that some or all of the nearest neighbors of a face image of a particular subject come from face images of other subjects, see Table 1.1. Moreover, choosing the right number of the nearest neighbors of a point is critical for successfully separating the data using the obtained similarity graph. However, most existing methods assume that the right number of nearest neighbors is initially provided to the algorithm. Thus, their performance greatly depends on a good choice of this quantity.

1.1.2 Dimensionality reduction

Once we separate the data into their underlying manifolds, a subsequent fundamental task is to recover a low-dimensional representation for the data in each

manifold. Such low-dimensional representations can reduce not only the computational cost and memory requirements of algorithms for processing the data, but also the effect of noise and corruption in the data, hence improving the performance of such algorithms.

In general, dimensionality reduction techniques can be divided into two main categories of linear and nonlinear methods. Linear dimensionality reduction methods such as Principal Component Analysis (PCA) [65], Probabilistic PCA (PPCA) [101], Factor Analyzer (FA) [69], and Random Projections (RP) [4], reduce the dimension of the data using a linear mapping from the ambient space to a lower dimensional latent space. When the data lie in a subspace, one can use linear dimensionality reduction techniques to effectively find the low-dimensional representations of the data.

Nonlinear dimensionality reduction (NLDR) techniques, on the other hand, try to reduce the dimension of the data via nonlinear embedding techniques. A variety of NLDR algorithms have been proposed, most of which use the following three step procedure [7, 24, 91–93, 98, 114]. In the first step, they build a nearest neighbor graph by connecting each point to its few nearest neighbors. Second, they learn a set of weights for the edges of the graph that capture the similarities and dissimilarities between data points. Third, they embed data points into a lower-dimensional space such that the similarities and dissimilarities between low-dimensional representations are well-preserved.

Different NLDR algorithms have different schemes for learning the weights. How-

ever, they all share the same first step, which is to build the nearest neighbor graph. As a result, such algorithms often have difficulties in selecting the right neighborhood size that captures well the manifold geometry. This comes from the fact that a suitable choice of the neighborhood size requires prior knowledge about the intrinsic dimension of the manifold, which is not typically known in real-world problems. As a result, the performance of NLDR methods greatly depends on the proper choice of the neighborhood graph.

1.1.3 Classification

In data classification problems in which each class corresponds to a different manifold, a fundamental task is to classify a given query and determine the manifold, i.e., the class, it belongs to. Depending on how the manifolds are modeled and how the data are distributed in each manifold, different classification schemes have been proposed.

When the manifolds of different classes are well sampled, i.e., there are sufficiently many training samples from each manifold, Nearest Neighbor (NN) classification algorithms [34] assign a test sample to the class of its nearest neighbor or in general to the dominating class of its K nearest neighbors. However, in practice, it is not often the case that there are sufficiently many training samples from each class. Nearest Subspace (NS) algorithm [62] models the data in a manifold by a low-dimensional subspace and, instead of assigning a test example to the class of its nearest neigh-

bor, assigns the test sample to the class of its closest subspace. While NS does not require many samples from each class, it only works well when the manifold of a class corresponds to a single low-dimensional subspace. In addition, the classification performance can be sensitive to the choice of the dimension of the subspace that one fits to the training data in each class.

The Sparse Representation-based Classification (SRC) algorithm [117], on the other hand, uses directly the training data across all classes at the same time in order to classify a given query. More precisely, SRC assumes that a test sample can be represented linearly in terms of all training data across all classes. However, a sparse representation would correspond to a linear combination of a few training samples from the right class. Thus, a sparse representation can be used to determine the class of a given query. While SRC works quite well in practical problems, there is no theoretical understanding of the modeling assumptions and conditions on the data and the manifolds under which the algorithm succeeds. Moreover, in practical problems, there are often structural relationships among the data in each class, which are not used in the SRC model.

1.2 Thesis contributions

In this thesis, we present new frameworks for the problems of clustering, dimensionality reduction and classification of multi-manifold data. We propose algorithms

based on sparse representation techniques to effectively address these problems. The key idea behind the proposed algorithms is what we call the *self-expressiveness property* of the data, which is that in an appropriate dictionary formed from the given data points in multiple manifolds, a sparse representation of a data point corresponds to selecting other data points from the same manifold. Our goal is then to search for such sparse representations and use them in appropriate frameworks to cluster, embed, and classify multi-manifold data. We propose sparse optimization programs to find such desired sparse representation techniques and develop theoretical guarantees for the success of the proposed algorithms.

1.2.1 Clustering in a union of subspaces

In the first part of the thesis, we propose and study an algorithm based on sparse representation techniques, called Sparse Subspace Clustering (SSC), to cluster a collection of data points lying in a union of low-dimensional subspaces [40, 41, 45]. The underlying idea behind the algorithm is the *self-expressiveness* property of the data, which is that each data point in a union of subspaces can be efficiently represented as a linear or an affine combination of other points. Such a representation is not unique in general because there are infinitely many ways a data point can be expressed as a combination of other points. The key observation is that a *sparse* representation of a data point ideally corresponds to a combination of a few points from its own subspace. This motivates solving a global sparse optimization program whose solution is used

in a spectral clustering framework to obtain the clustering of the data. As a result, we can overcome the problems of local spectral clustering-based algorithms, such as choosing the right neighborhood size and dealing with points near the intersection of subspaces, since, for a given data point, the sparse optimization program automatically selects a few points that are not necessarily close to it but belong to the same subspace.

Since the sparse optimization program is in general NP-hard, we consider a convex relaxation. We show that, under mild conditions on the arrangement of subspaces and data distribution, the proposed convex minimization program recovers the desired solution, guaranteeing the success of the algorithm. Our theoretical analysis extends the sparse representation theory to the multi-subspace setting where the number of points in a subspace is arbitrary, possibly much larger than its dimension.

The proposed convex minimization program can be solved efficiently using convex programming tools [10, 14] and does not require initialization. Our algorithm can directly deal with data corruptions such as noise, sparse outlying entries, and missing entries as well as the more general class of affine subspaces by modifying the sparse optimization program to incorporate the corruption and the subspace model. Through experimental results, we show that our algorithm outperforms the state-of-the-art subspace clustering methods on two real-world problems of motion segmentation and face clustering.

1.2.2 Clustering and embedding in a union of nonlinear manifolds

In the second part of the thesis, we consider the more general problem of clustering and dimensionality reduction of data lying in a union of nonlinear manifolds. We propose an algorithm based on sparse representation techniques, called Sparse Manifold Clustering and Embedding (SMCE), for *simultaneous clustering and embedding* of data [43]. Unlike conventional methods that first build a neighborhood graph and then learn a set of weights for it, our method simultaneously builds the neighborhood graph and learns its weights. This leads to successful results even in challenging situations where the manifolds are spatially close to each other.

More specifically, we use the geometrically motivated assumption that for each data point there exists a small neighborhood in which only the points that come from the same manifold lie approximately in a low-dimensional affine subspace. We propose a sparse optimization program to select *a few neighbors* of each data point that span a *low-dimensional affine subspace* passing near that point. As a result, a few nonzero elements of the solution indicate the points that are on the same manifold. Hence, they can be used for clustering. In addition, the weights associated with the chosen neighbors indicate their similarities to the given data point, which can be used for dimensionality reduction. Clustering and embedding of the data into lower dimensions follows by taking the eigenvectors of the matrix of weights and its

submatrices, which are sparse, and hence can be stored and be operated on efficiently.

Thanks to the sparse representation framework employed by SMCE, we do not need to specify the number of nearest neighbors a priori. In fact, the optimization program selects the neighbors of each point automatically, where the number of neighbors of a data point depends on the local intrinsic dimensionality of the manifold at that point. Finally, SMCE has only one free parameter that, for a large range of values, results in a stable clustering and embedding, as we will show in the experimental results. To the best of our knowledge, SMCE is the only algorithm proposed to date that allows robust automatic selection of neighbors and simultaneous clustering and dimensionality reduction in a *unified manner*.

1.2.3 Classification in a union of subspaces

In the last part of the thesis, we consider the classification of multi-manifold data, where the training data in each class lie in a manifold characterized by a union of low-dimensional subspaces. We show that instead of looking for the sparsest representation of a test example in the dictionary of all the training data, a better criterion for classification is to look for a representation of the test example that involves the minimum number of blocks from the dictionary [42]. As a result, we study the problem of block-sparse recovery and consider two classes of block-sparse optimization programs to address the problem. We study conditions under which each class of the convex programs can recover the desired solution [42, 44].

To evaluate the classification performance of the two classes of convex programs, we consider synthetic experiments as well as the problem of automatic face recognition. By extensive experiments, we show that the methods based on block-sparse representation improve the state-of-the-art face recognition results for classifying both uncorrupted and corrupted data. More specifically, we show that the proposed convex programs improve face recognition results by 10% when the number of training data in each class is as small as the dimension of the face subspace. In addition, we show that the algorithms can efficiently handle corruptions and occlusions.

1.3 Thesis outline

The rest of the thesis is outlined as follows. In Chapter 2, we review the sparse representation theory for recovering sparse and group-sparse representations of signals/vectors in a given dictionary. We also review some of the most important manifold clustering and embedding techniques. In Chapter 3, we present our work for clustering of data in a union of subspaces based on sparse representation techniques. We study the theoretical guarantees of the proposed method and evaluate it on synthetic data as well as the real-world problems of motion segmentation and face clustering. In Chapter 4, we generalize the result to clustering and dimensionality reduction of data in a union of nonlinear manifolds. We present an algorithm that can deal well with spatially close manifolds as well as manifolds with non-uniform sampling and holes.

We demonstrate the effectiveness of the proposed algorithm on several synthetic and real examples. In Chapter 5, we present our work for classification of multi-manifold data where the manifold of each class is modeled as a union of low-dimensional subspaces. We study two classes of convex optimization programs whose solutions are used in order to classify a given query. We investigate the theoretical guarantees of the convex programs and evaluate them on synthetic data as well as the real-world problem of face recognition. Finally, we summarize the conclusions of this work in Chapter 6.

Chapter 2

Background Material

Throughout this thesis, we denote vectors by boldface lowercase letters, such as $\boldsymbol{a}$, and denote matrices by boldface uppercase letters, such as $\boldsymbol{A}$. Moreover, throughout the thesis, we assume that vectors and matrices consist of real valued elements. $\boldsymbol{a} \in \mathbb{R}^m$ indicates a column vector that consists of m real-valued elements, and $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ indicates a matrix with m rows and n columns that consists of real-valued elements. The transposition operator is denoted by $[\,\cdot\,]^\top$. Specifically, $\boldsymbol{A}^\top \in \mathbb{R}^{n \times m}$ is the transpose of $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, where the element at each row i and each column j of $\boldsymbol{A}^\top$ is equal to the element at the row j and the column i of $\boldsymbol{A}$.

We denote by $\mathbf{1}_m \in \mathbb{R}^m$ a vector whose elements are all equal to one and denote by $\boldsymbol{I}_m$ the identity matrix in $\mathbb{R}^{m \times m}$, i.e., a diagonal matrix whose diagonal entries are all equal to one. We drop the subscript and use $\mathbf{1}$ and $\boldsymbol{I}$ whenever the dimensions are clear from the context.

2.1 Vector and matrix norms

Consider a vector $\boldsymbol{a} = \begin{bmatrix} a_1 & \dots & a_m \end{bmatrix}^\top \in \mathbb{R}^m$, which consists of m real valued elements, a_i. The ℓ_q-norm of $\boldsymbol{a}$, for $q > 0$, is defined as

$$\|\boldsymbol{a}\|_q \triangleq (\sum_{i=1}^{m} |a_i|^q)^{1/q}. \tag{2.1}$$

ℓ_∞-norm is defined by taking a limit as $q \to \infty$. It can be shown that ℓ_∞-norm is equal to the maximum absolute value of the elements of $\boldsymbol{a}$, i.e.,

$$\|\boldsymbol{a}\|_\infty = \max_i |a_i|. \tag{2.2}$$

For the specific case of $q = 0$, the ℓ_0-norm of $\boldsymbol{a}$, denoted by $\|\boldsymbol{a}\|_0$[1], is defined as the number of nonzero elements of $\boldsymbol{a}$, i.e.,

$$\|\boldsymbol{a}\|_0 = \sum_{i=1}^{n} \mathrm{I}(|a_i| > 0), \tag{2.3}$$

where $\mathrm{I}(\cdot)$ denotes the indicator function whose value is equal to one when its argument is true and equal to zero otherwise. The ℓ_0-norm can also be considered as the limit of the ℓ_q-norm of $\boldsymbol{a}$ when q approaches zero.

Given a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, the ℓ_q-norm of $\boldsymbol{A}$, for $q > 0$, is defined as the ℓ_q-norm of a vector that consists of all the elements of $\boldsymbol{A}$, i.e.,

$$\|\boldsymbol{A}\|_q = (\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^q)^{1/q}, \tag{2.4}$$

[1]Note that $\|\boldsymbol{a}\|_0$ is not in fact a real norm, since it does not satisfy the properties of a norm function. However, it is commonly referred to as a norm in the literature.

where a_{ij} denotes the element at the row i and the column j of $\boldsymbol{A}$. For example, $\|\boldsymbol{A}\|_1$ is the sum of the absolute values of the elements of $\boldsymbol{A}$. $\|\boldsymbol{A}\|_2$, known as the Frobenius norm and denoted also by $\|\boldsymbol{A}\|_F$, is the square root of the sum of the squared elements of $\boldsymbol{A}$.

For $p, q > 0$, the mixed ℓ_p/ℓ_q-norm of $\boldsymbol{A} = \begin{bmatrix} \boldsymbol{a}_1 & \dots & \boldsymbol{a}_n \end{bmatrix} \in \mathbb{R}^{m \times n}$ is defined as

$$\|\boldsymbol{A}\|_{p,q} = (\sum_{i=1}^{n} \|\boldsymbol{a}_i\|_q^p)^{1/p}. \tag{2.5}$$

For example, $\|\boldsymbol{A}\|_{\infty,2}$ corresponds to the maximum ℓ_2-norm of the columns of $\boldsymbol{A}$.

2.2 Sparse representation theory

In this section, we review the sparse representation theory as well as some of the main theoretical results, which we will make connection to later in this thesis.

2.2.1 Recovery of sparse signals

Sparse signal recovery has drawn increasing attention in many areas such as signal and image processing, computer vision, machine learning, and bioinformatics (see e.g., [20, 35, 86, 116] and the references therein). The key assumption behind sparse signal recovery is that an observed signal $\boldsymbol{y} \in \mathbb{R}^D$ can be written as a linear combination of a few atoms of a given dictionary $\boldsymbol{B} \in \mathbb{R}^{D \times N}$. More formally, consider an underdetermined system of linear equations of the form $\boldsymbol{y} = \boldsymbol{B}\boldsymbol{c}$, where $\boldsymbol{y}$ lies in the

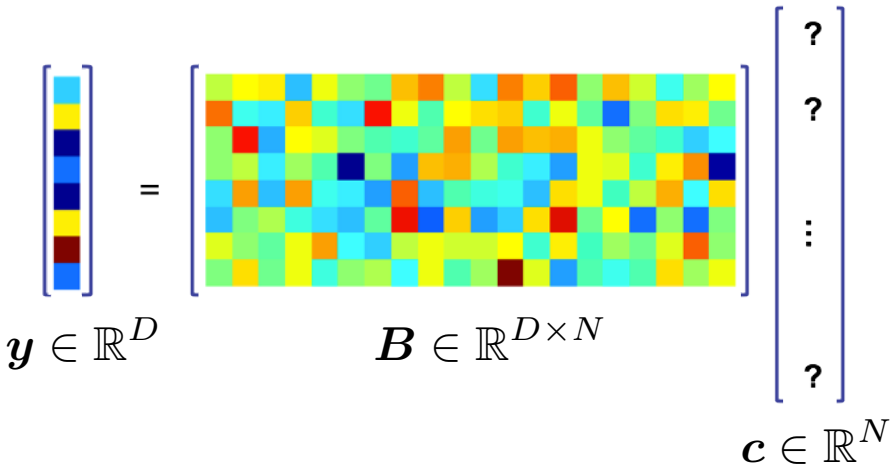


Figure 2.1: In an underdetermined dictionary $\boldsymbol{B} \in \mathbb{R}^{\boldsymbol{D} \times N}$, there are infinitely many representations $\boldsymbol{c}$ for a given $\boldsymbol{y}$. Sparse representation refers to a $\boldsymbol{c}$ that has only a few nonzero elements.

range-space of $\boldsymbol{B}$ that has more columns than rows ($N > D$), hence allowing infinitely many solutions for $\boldsymbol{y}$ (see Figure 2.1). Sparsity of the desired solution arises in many problems and can be used to restrict the set of possible solutions. In principle, the problem of finding the sparsest representation of a given signal can be cast as the following optimization program

$$P_{\ell_0} : \min \|\boldsymbol{c}\|_0 \quad \text{s.t.} \quad \boldsymbol{y} = \boldsymbol{B}\boldsymbol{c}. \tag{2.6}$$

We say that a vector $\boldsymbol{c}$ is k-sparse if it has at most k nonzero elements. While finding the sparse representation of a given signal using P_{ℓ_0} is in general NP-hard [1], the pioneering work of Donoho [33] and Candes [21] showed that, under appropriate conditions, this problem can be solved efficiently as

$$P_{\ell_1} : \min \|\boldsymbol{c}\|_1 \quad \text{s.t.} \quad \boldsymbol{y} = \boldsymbol{B}\boldsymbol{c}. \tag{2.7}$$

Since then, there has been an outburst of research articles investigating conditions under which the two optimization programs, P_{ℓ_1} and P_{ℓ_0}, are equivalent.

Nullspace property. The work of [30,58] derives necessary and sufficient conditions for the recovery of sparse signals. More specifically, Let Λ_k denote a set of k different indices from $\{1, \ldots, N\}$ and $\Lambda_{\widehat{k}}$ denote the remaining $N - k$ indices. Let $\boldsymbol{B}_{\Lambda_k} \in \mathbb{R}^{D \times k}$ and $\boldsymbol{B}_{\Lambda_{\hat{k}}} \in \mathbb{R}^{D \times N-k}$ be submatrices of $\boldsymbol{B}$ whose columns are selected from $\boldsymbol{B}$ according to Λ_k and $\Lambda_{\widehat{k}}$, respectively. The nullspace property states that P_{ℓ_1} and P_{ℓ_0} are equivalent and recover the k-sparse representation of any given signal if and only if

$$\forall \boldsymbol{c}_k \neq \mathbf{0}, \forall \boldsymbol{c}_{\widehat{k}} \neq \mathbf{0} \;\text{ such that }\; \boldsymbol{B}_{\Lambda_k}\boldsymbol{c}_k = \boldsymbol{B}_{\Lambda_{\widehat{k}}}\boldsymbol{c}_{\widehat{k}} \implies \|\boldsymbol{c}_k\|_1 < \|\boldsymbol{c}_{\widehat{k}}\|_1. \tag{2.8}$$

In other words, for every nonzero vector $\mathbf{x}$ that lives in the range-space of both $\boldsymbol{B}_{\Lambda_k}$ and $\boldsymbol{B}_{\Lambda_{\widehat{k}}}$, the ℓ_1-norm of the representation of $\mathbf{x}$ in $\boldsymbol{B}_{\Lambda_k}$ must be strictly smaller than the ℓ_1-norm of any representation of $\mathbf{x}$ in $\boldsymbol{B}_{\Lambda_{\widehat{k}}}$.

While the null-space property provides necessary and sufficient conditions for the equivalence of P_{ℓ_1} and P_{ℓ_0}, it is not possible to check the above condition for every Λ_k and for every $\mathbf{x}$ in the intersection of the ranges-spaces of $\boldsymbol{B}_{\Lambda_k}$ and $\boldsymbol{B}_{\Lambda_{\widehat{k}}}$. Moreover, the condition above does not explicitly characterize the relationships among the atoms of the dictionary under which P_{ℓ_1} and P_{ℓ_0} are equivalent. This has motivated investigating a series of sufficient conditions based on the notions of mutual/cumulative coherence [31, 104] and the restricted isometry property [17, 21], which we describe next. Throughout the section, we assume that the columns of $\boldsymbol{B}$ have unit Euclidean

norm.

Mutual/Cumulative coherence. The mutual coherence of a dictionary $\boldsymbol{B}$ is defined as

$$\mu \triangleq \max_{i \neq j} |\boldsymbol{b}_i^\top \boldsymbol{b}_j|, \tag{2.9}$$

where $\boldsymbol{b}_i$ denotes the i-th column of $\boldsymbol{B}$ of unit Euclidean norm. [104] and [31] show that if the sufficient condition

$$(2k-1)\mu < 1 \tag{2.10}$$

holds, then the optimization programs P_{ℓ_1} and P_{ℓ_0} are equivalent and recover the k-sparse representation of a given signal. While μ can be easily computed, it does not characterize a dictionary very well since it measures the most extreme correlations in the dictionary.

To better characterize a dictionary, cumulative coherence measures the maximum total coherence between a fixed atom and a collection of k other atoms. Specifically, the cumulative coherence associated with a positive integer k [104] is defined as

$$\zeta_k \triangleq \max_{\Lambda_k} \max_{i \notin \Lambda_k} \sum_{j \in \Lambda_k} |\boldsymbol{b}_i^\top \boldsymbol{b}_j|, \tag{2.11}$$

where Λ_k denotes a set of k different indices from $\{1, \ldots, N\}$. Note that for $k = 1$, we have $\zeta_1 = \mu$. Although cumulative coherence is, in general, more difficult to compute than mutual coherence, it provides sharper results for the equivalence of P_{ℓ_1} and P_{ℓ_0}. In particular, [104] shows that if

$$\zeta_k + \zeta_{k-1} < 1, \tag{2.12}$$

then the optimization programs P_{ℓ_1} and P_{ℓ_0} are equivalent and recover the k-sparse representation of a given signal. Using the definitions, one can verify that $\zeta_k \leq k\mu$ for all integers $k \geq 1$. As a result, (2.12) provides a weaker condition than (2.10) for sparse recovery of signals.

Restricted isometry property. An alternative sufficient condition for the equivalence between P_{ℓ_1} and P_{ℓ_0} is based on the so-called restricted isometry property (RIP) [17, 21]. For a positive integer k, the restricted isometry constant of a dictionary $\boldsymbol{B}$ is defined as the smallest constant δ_k for which

$$(1 - \delta_k)\|\boldsymbol{c}\|_2^2 \leq \|\boldsymbol{B}\boldsymbol{c}\|_2^2 \leq (1 + \delta_k)\|\boldsymbol{c}\|_2^2 \tag{2.13}$$

holds for all k-sparse vectors $\boldsymbol{c}$. [17] shows that if $\delta_{2k} < \sqrt{2} - 1$, then P_{ℓ_1} and P_{ℓ_0} are equivalent. The bound in this result has been further improved and [48] shows that if $\delta_{2k} < 0.4652$, then P_{ℓ_1} and P_{ℓ_0} are equivalent.

2.2.2 Recovery of block-sparse signals

Recently, there has been growing interest in recovering sparse representations of signals in a union of a large number of subspaces, under the assumption that the signals live in the direct sum of only a few subspaces. Such a representation whose nonzero elements appear in a few blocks is called a *block-sparse* representation. Block sparsity arises in various applications such as reconstructing multi-band signals [80, 81], measuring gene expression levels [86], face/digit/speech recognition [42, 52,

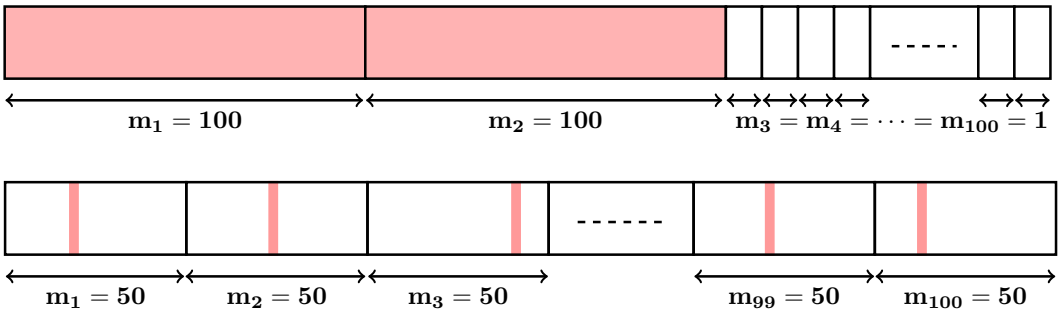


Figure 2.2: Top: a block-sparse vector is not necessarily sparse. In this example, 2 nonzero blocks out of 100 blocks correspond to 200 nonzero elements out of 298 elements. Bottom: a sparse vector is not necessarily block-sparse. In this example, all 100 blocks are nonzero each having one nonzero element. However, this gives rise to only 50 nonzero elements out of 5,000 elements.

53, 117], clustering of data on multiple subspaces [40, 41, 45], finding exemplars in datasets [39], multiple measurement vector recovery [23, 26, 70, 108], etc.

The recovery of block-sparse signals involves solving a system of linear equations of the form

$$\boldsymbol{y} = \boldsymbol{B}\boldsymbol{c} = \begin{bmatrix} \boldsymbol{B}[1] & \cdots & \boldsymbol{B}[n] \end{bmatrix} \boldsymbol{c}, \tag{2.14}$$

where $\boldsymbol{B} \in \mathbb{R}^{D \times m}$ consists of n blocks $\boldsymbol{B}[i] \in \mathbb{R}^{D \times m_i}$ and $m = \sum_{i=1}^{n} m_i$. The main difference with respect to classical sparse recovery is that the desired solution of (2.14) corresponds to a few nonzero *blocks* rather than a few nonzero *elements* of $\boldsymbol{B}$. We say that a vector $\boldsymbol{c}^\top = \begin{bmatrix} \boldsymbol{c}[1]^\top & \cdots & \boldsymbol{c}[n]^\top \end{bmatrix}$ is *k-block-sparse*, if at most k blocks $\boldsymbol{c}[i] \in \mathbb{R}^{m_i}$ are different from zero. Note that, in general, a block-sparse vector is not necessarily sparse and vice versa, as shown in Figure 2.2.

The problem of finding a representation of a signal $\boldsymbol{y}$ that uses the minimum

number of blocks of $\boldsymbol{B}$ can be cast as the following optimization program

$$P_{\ell_q/\ell_0}: \min \sum_{i=1}^{n} \mathrm{I}(\|\boldsymbol{c}[i]\|_q) \quad \text{s.t.} \quad \boldsymbol{y} = \boldsymbol{B}\boldsymbol{c}, \tag{2.15}$$

where $q \geq 0$ and $I(\cdot)$ is the indicator function, which is zero when its argument is zero and is one otherwise. In fact, the objective function in (2.15) counts the number of nonzero blocks of a solution. However, solving (2.15) is in general an NP-hard problem [1] as it requires searching exhaustively over all choices of a few blocks of $\boldsymbol{B}$ and checking whether they span the observed signal. The ℓ_1 relaxation of P_{ℓ_q/ℓ_0} has the following form

$$P_{\ell_q/\ell_1}: \min \sum_{i=1}^{n} \|\boldsymbol{c}[i]\|_q \quad \text{s.t.} \quad \boldsymbol{y} = \boldsymbol{B}\boldsymbol{c}. \tag{2.16}$$

For $q \geq 1$, the optimization program P_{ℓ_q/ℓ_1} is convex and can be solved efficiently using convex programming tools [14].

Remark 1 *For $q = 1$, the convex program P_{ℓ_1/ℓ_1} is the same as P_{ℓ_1} in (2.7) used for sparse recovery. In other words, while the ℓ_1 optimization program, under some conditions, can recover a sparse representation of a signal, it can also recover a block-sparse representation, under appropriate conditions, as we will discuss in the thesis.*

The work of [36,37,96] study conditions under which for the special case of $q = 2$, P_{ℓ_2/ℓ_1} and P_{ℓ_2/ℓ_0} are equivalent. These conditions are based on generalizations of the null-space property, mutual coherence and restricted isometry property, as we describe next.

Block-nullspace property. The block-nullspace property provides necessary and sufficient conditions for the equivalence of the two optimization programs P_{ℓ_2/ℓ_1} and P_{ℓ_2/ℓ_0} [96]. More specifically, Let Λ_k denote a set of k different indices from $\{1,\ldots,n\}$ and $\Lambda_{\widehat{k}}$ denote the remaining $n-k$ indices. Let $\boldsymbol{B}_{\Lambda_k}$ and $\boldsymbol{B}_{\Lambda_{\hat{k}}}$ be submatrices of $\boldsymbol{B}$ whose blocks are selected from $\boldsymbol{B}$ according to Λ_k and $\Lambda_{\widehat{k}}$, respectively. The block-nullspace property states that P_{ℓ_2/ℓ_1} and P_{ℓ_2/ℓ_0} are equivalent and recover the k-block-sparse representation of any given signal if and only if

$$\forall \boldsymbol{c}_k \neq \boldsymbol{0}, \forall \boldsymbol{c}_{\widehat{k}} \neq \boldsymbol{0} \;\text{ such that }\; \boldsymbol{B}_{\Lambda_k}\boldsymbol{c}_k = \boldsymbol{B}_{\Lambda_{\widehat{k}}}\boldsymbol{c}_{\widehat{k}} \implies \sum_{i=1}^{k} \|\boldsymbol{c}_k[i]\|_2 < \sum_{i=1}^{n-k} \|\boldsymbol{c}_{\widehat{k}}[i]\|_2. \tag{2.17}$$

In other words, for every nonzero vector $\mathbf{x}$ that lies in the range-space of both $\boldsymbol{B}_{\Lambda_k}$ and $\boldsymbol{B}_{\Lambda_{\widehat{k}}}$, the mixed ℓ_2/ℓ_1-norm of the representation of $\mathbf{x}$ in $\boldsymbol{B}_{\Lambda_k}$ must be strictly smaller than the mixed ℓ_2/ℓ_1-norm of any representation of $\mathbf{x}$ in $\boldsymbol{B}_{\Lambda_{\widehat{k}}}$.

In practice, it is not possible to check the condition above for every Λ_k and for every $\mathbf{x}$ in the intersection of the range-spaces of $\boldsymbol{B}_{\Lambda_k}$ and $\boldsymbol{B}_{\Lambda_{\widehat{k}}}$. Moreover, the condition above does not explicitly characterize the relationships among the atoms and the blocks of the dictionary under which P_{ℓ_2/ℓ_1} and P_{ℓ_2/ℓ_0} are equivalent. This has motivated investigating sufficient conditions based on generalizations of the mutual coherence and restricted isometry property in the conventional sparse recovery.

Block-coherence. The work of [36] assumes that the blocks have linearly independent columns and are of the same length d, i.e., for each i, $\text{rank}(\boldsymbol{B}[i]) = m_i = d$.

Under these assumptions, [36] defines the block-coherence of a dictionary $\boldsymbol{B}$ as

$$\mu_B \triangleq \max_{i \neq j} \sigma_1(\boldsymbol{B}[i]^\top \boldsymbol{B}[j]), \tag{2.18}$$

where $\sigma_1(\cdot)$ denotes the largest singular value of the given matrix. Also, the subcoherence of $\boldsymbol{B}$ is defined as $\nu \triangleq \max_i \mu_i$ where μ_i denotes the mutual coherence for the i-th block. [36] shows that if

$$(2k-1)\mu_B < 1 - (d-1)\nu, \tag{2.19}$$

then P_{ℓ_2/ℓ_1} and P_{ℓ_2/ℓ_0} are equivalent and recover the k-block-sparse representation of a given signal. Note that when $d = 1$, (2.19) reduces to (2.10).

Block-RIP. [37] assumes that the blocks have linearly independent columns, although their lengths need not be equal. Under this assumption, [37] defines the block restricted isometry constant of $\boldsymbol{B}$ as the smallest constant $\delta_{B,k}$ such that

$$(1 - \delta_{B,k})\|\boldsymbol{c}\|_2^2 \leq \|\boldsymbol{B}\boldsymbol{c}\|_2^2 \leq (1 + \delta_{B,k})\|\boldsymbol{c}\|_2^2 \tag{2.20}$$

holds for every k-block-sparse vector $\boldsymbol{c}$. Analogous to the conventional sparse recovery results, [37] shows that if $\delta_{B,2k} < \sqrt{2} - 1$, then P_{ℓ_2/ℓ_1} and P_{ℓ_2/ℓ_0} are equivalent.

The work of [96] proposes an alternative analysis framework for block-sparse recovery using P_{ℓ_2/ℓ_1} in the special case of Gaussian dictionaries. By analyzing the nullspace of the dictionary, it shows that if the blocks have linearly independent columns, perfect recovery is achieved with high probability as the length of the signal, D, grows to infinity.

An alternative approach to recover the block-sparse representation of a given signal is to solve the optimization program

$$P'_{\ell_q/\ell_0}\colon \min \sum_{i=1}^{n} \mathrm{I}(\|\boldsymbol{B}[i]\boldsymbol{c}[i]\|_q) \quad \text{s.t.} \quad \boldsymbol{y} = \boldsymbol{B}\boldsymbol{c}, \tag{2.21}$$

for $q \geq 0$. Notice that the solution to this problem coincides with that of P_{ℓ_q/ℓ_0} for blocks with linearly independent columns since $\|\boldsymbol{B}[i]\boldsymbol{c}[i]\|_q > 0$ if and only if $\|\boldsymbol{c}[i]\|_q > 0$. Nevertheless, P'_{ℓ_q/ℓ_0} is an NP-hard problem. In the case of $q \geq 1$, the following ℓ_1 relaxation

$$P'_{\ell_q/\ell_1}: \min \sum_{i=1}^{n} \|\boldsymbol{B}[i]\boldsymbol{c}[i]\|_q \quad \text{s.t.} \quad \boldsymbol{y} = \boldsymbol{B}\boldsymbol{c}, \tag{2.22}$$

is a convex program and can be solved efficiently. The work of [50] studies conditions under which, for the special case of $q = 2$, P'_{ℓ_2/ℓ_1} and P'_{ℓ_2/ℓ_0} are equivalent. The conditions are based on the notion of mutual subspace incoherence, as described next.

Mutual subspace coherence. The work of [50] introduces the notion of mutual subspace coherence of $\boldsymbol{B}$, which is defined as

$$\mu_S = \max_{i \neq j} \max_{\mathbf{x} \in \mathcal{S}_i, \boldsymbol{z} \in \mathcal{S}_j} \frac{|\mathbf{x}^\top \boldsymbol{z}|}{\|\mathbf{x}\|_2 \|\boldsymbol{z}\|_2}, \tag{2.23}$$

where $\mathcal{S}_i = \mathrm{span}(\boldsymbol{B}[i])$. Under the assumption that the blocks have linearly independent columns and the subspaces spanned by each block are disjoint, [50] shows that P'_{ℓ_2/ℓ_1} and P'_{ℓ_2/ℓ_0} are equivalent if

$$(2k-1)\mu_S < 1. \tag{2.24}$$

As mentioned above, the state-of-the-art block-sparse recovery methods [11, 36, 37, 50, 96] consider dictionaries whose blocks consist of linearly independent vectors which we refer to as *non-redundant blocks*. However, in signal/image processing, machine learning, and computer vision problems such as face recognition [42, 117] and motion segmentation [40, 88], blocks of a dictionary consist of data points and often the number of data points in each block exceeds the dimension of the underlying subspace. For example, in automatic face recognition, the number of training images in each block of the dictionary is often more than the dimension of the face subspace, known to be 9 under a fixed pose and varying illumination [6]. In fact, having more data in each block helps to better capture the underlying distribution of the data in each subspace, and hence increases the performance of tasks such as classification. However, to the best of our knowledge, existing theoretical results have not addressed recovery in dictionaries whose blocks have linearly dependent atoms, which we refer to as *redundant blocks*. Moreover, theoretical analysis for the equivalence between P_{ℓ_q/ℓ_1} and P_{ℓ_q/ℓ_0} as well as the equivalence between P'_{ℓ_q/ℓ_1} and P'_{ℓ_q/ℓ_0} has been restricted to only $q = 2$. Nevertheless, empirical studies in some applications [126], have shown better block-sparse recovery performance for $q \neq 2$. Therefore, there is a need for analyzing the performance of each class of convex programs for arbitrary $q \geq 1$.

Remark 2 *Another line of research that considers the problem of block-sparse recovery is the work of [3, 64, 85]. However, the modeling assumption on the dictionary is different from what we have reviewed so far and what we are interested in this thesis.*

Specifically, [3, 64, 85] consider the regularized least-squares problem of

$$\min_{\boldsymbol{c}} \ \frac{1}{2}\|\boldsymbol{y}-\boldsymbol{B}\boldsymbol{c}\|_2^2+\lambda\,\Omega(\boldsymbol{c}), \tag{2.25}$$

where $\Omega(\boldsymbol{c})$ is a block-sparsity inducing norm on the coefficient vector $\boldsymbol{c}$. However, they make assumptions about the rows of the matrix $\boldsymbol{B}$, while [36, 37] and the work in this thesis make assumptions about the columns of $\boldsymbol{B}$. In fact, the assumptions in [3, 64, 85] almost never hold in the setting considered in this thesis. More precisely, the assumption A2 in [3] or the result of Proposition 1 and Theorem 6 in [64] require the empirical covariance matrix formed by the rows of $\boldsymbol{B}$ to be full-rank. Similarly, [85] assumes that the rows of $\boldsymbol{B}$ are drawn in an i.i.d. manner from a zero mean Gaussian distributions with a positive definite covariance matrix. However, such assumptions are easily violated when the columns of $\boldsymbol{B}$ are drawn from a union of subspaces. For example, when the columns of $\boldsymbol{B}$ are drawn from a one-dimensional line, the covariance matrix formed by the rows of $\boldsymbol{B}$ is always rank deficient (is of rank one). As a result, neither the work of [3, 64] nor our work is a special case of the other and the obtained conditions in our work cannot be compared with the conditions in [3, 64] and vice versa.

The work of [121] also considers the optimization program in (2.25) *where the blocks of $\boldsymbol{B}$ consist of orthonormal columns. However, in our work, we consider the constraint optimization program in* (2.16) *instead of the unconstraint optimization in* (2.25) *and we study the more general case where the blocks of $\boldsymbol{B}$ can consist of linearly dependent columns.*

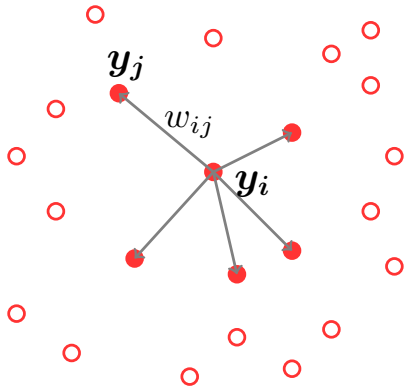


Figure 2.3: To build a neighborhood graph, one connects each node to its K nearest neighbors or to the nodes that are inside an ϵ-ball centered at node.

2.3 Manifold clustering and embedding

In this section, we review some of the most important algorithms for clustering and dimensionality reduction of data in manifolds. Given N data points $\{\boldsymbol{y}_i \in \mathbb{R}^D\}_{i=1}^N$ that lie in a union of n manifolds $\{\mathcal{M}_\ell\}_{\ell=1}^n$, the problem of manifold clustering refers to the problem of separating the data into their underlying manifolds. In other words, we would like to find a partition $\{\Lambda_\ell\}_{\ell=1}^n$ for the set $\{1, 2, \ldots, N\}$ such that data points indexed by Λ_ℓ belong to the same manifold $\mathcal{M}_\ell$. Once we cluster the data, the dimensionality reduction (embedding) problem aims at finding compact representations $\{\mathbf{x}_i \in \mathbb{R}^{d_\ell}\}_{i \in \Lambda_\ell}$ for the data in each manifold, where $\boldsymbol{d}_\ell < D$ denotes the embedding dimension of the data in cluster ℓ.

The majority of manifold clustering and embedding algorithms relies on building a neighborhood graph whose nodes represent the data points and whose edge weights encode information that can be used for clustering and embedding, see Figure 2.3.

Such algorithms often follow the following three step procedure:

1. Build a neighborhood graph with N nodes representing the N data points. Connect each point to its K nearest neighbors or to points that are inside an ϵ-ball centered at that point.

2. Learn a set of weights $\{w_{ij}\}$ describing the similarity between a node i and any node j, using the neighborhood graph.

3. Find clustering and low-dimensional representations for the data using the learned weights.

Most algorithms assume that a good value of K or ϵ is given, hence a good neighborhood graph is available. The main difference among algorithms is in their procedures for learning the weights.

2.3.1 Laplacian eigenmaps and spectral clustering

The Laplacian eigenmaps (LEM) algorithm [7] is motivated by the geometric idea that data points that are close in the high-dimensional space should remain close to each other in the low-dimensional embedding. As a result, LEM learns weights between a point and its neighbors and tries to preserve these weights in the low-dimensional representation of the data points. LEM proposes two approaches to choose the weights for the edges of the neighborhood graph:

- [Simple-minded] set $w_{ij} = 1$ if nodes i and j are connected by an edge in the neighborhood graph, otherwise set $w_{ij} = 0$.
- [Heat kernel] given a parameter $t > 0$, set $w_{ij} = e^{-\frac{\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2}{t}}$ if nodes i and j are connected by an edge in the neighborhood graph, otherwise set $w_{ij} = 0$.

Once the weights are learned, the last step is to find low-dimensional representations $\{\mathbf{x}_i \in \mathbb{R}^d\}_{i=1}^N$ for the data such that $\mathbf{x}_i$ and $\mathbf{x}_j$ are close if $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ are close, i.e., if nodes i and j are connected in the neighborhood graph. Such low-dimensional representations are obtained by finding the bottom eigenvectors of the normalized Laplacian matrix of the graph. More specifically, denoting the weight matrix as $\boldsymbol{W} = [w_{ij}] \in \mathbb{R}^{N \times N}$, LEM forms the normalized Laplacian matrix of the graph as

$$\boldsymbol{L} = \boldsymbol{I} - \boldsymbol{D}^{-1}\boldsymbol{W}, \tag{2.26}$$

where $\boldsymbol{D} = \mathrm{diag}(d_1, \ldots, d_N)$ is a diagonal matrix whose i-th element is equal to the sum of the weights of the edges connected to the node i, i.e., $d_i = \sum_{j=1}^N w_{ij}$. The Laplacian matrix is symmetric positive semidefinite [7]. Collecting the d eigenvectors of $\boldsymbol{L}$ corresponding to its second to the $(d+1)$-th smallest eigenvalues in a matrix $\boldsymbol{V} \in \mathbb{R}^{N \times d}$, the d-dimensional representation of the N data points are then given by the rows of $\boldsymbol{V}$.

When the data lie in n manifolds, under the assumption that the points in different manifolds are well-separated from each other, the graph will have n connected components, each component corresponding to a manifold. In such a case, one can

use the normalized Laplacian matrix of the graph, defined in (4.11), and find the components of the graph by applying the Kmeans algorithm [34] to the rows of a matrix $\boldsymbol{V} \in \mathbb{R}^{N \times n}$, which is composed of the n eigenvectors of $\boldsymbol{L}$ corresponding to its smallest eigenvalues. Once the data are separated into their underlying manifolds, in order to find a low-dimensional representation of the data in each manifold, one has to apply the LEM algorithm separately to the data points in each cluster.

2.3.2 Locally linear embedding and clustering

The Locally Linear Embedding (LLE) algorithm [91] uses the geometric idea that each point and its nearest neighbors lie in or close to a locally linear patch of the manifold. Thus, each data point, $\boldsymbol{y}_i$, can be approximately reconstructed using the affine combination of its nearest neighbors, $\{\boldsymbol{y}_j\}_{j \in \mathcal{N}_i}$, where $\mathcal{N}_i$ represents the indices of the neighbors of $\boldsymbol{y}_i$, suggesting to solve

$$\min_{\{w_{ij}\}_{j \in \mathcal{N}_i}} \left\| \boldsymbol{y}_i - \sum_{j \in \mathcal{N}_i} w_{ij} \boldsymbol{y}_j \right\|_2^2 \quad \text{s.t.} \quad \sum_{j \in \mathcal{N}_i} w_{ij} = 1. \tag{2.27}$$

It is shown in [91] that the obtained weights are invariant with respect to rotations and global translation of the data. Hence, the leaned weights $\{w_{ij}\}$ reflect intrinsic geometric properties of the data.

LLE then tries to reconstruct a neighborhood preserving mapping such that the low-dimensional representations of data points preserve the geometric relationships of the data in the high-dimensional space. More precisely, in the low-dimensional space,

one expects that $\mathbf{x}_i$ can be efficiently reconstructed from $\{\mathbf{x}_j\}_{j\in\mathcal{N}_i}$ using the learned weights. This suggests looking for $\mathbf{x}_i$ that minimize

$$\left\|\mathbf{x}_i - \sum_{j\in\mathcal{N}_i} w_{ij}\mathbf{x}_j\right\|_2^2, \tag{2.28}$$

using the learned weights. Collecting $\mathbf{x}_i$ as the columns of a matrix $\boldsymbol{X} \in \mathbb{R}^{d\times N}$, and building the weight matrix $\boldsymbol{W} = [w_{ij}]$, the LLE algorithm finds the embedding vectors by solving the minimization program

$$\min_{\boldsymbol{X}} \|\boldsymbol{X} - \boldsymbol{X}\boldsymbol{W}^\top\|_F^2 \quad \text{s.t.} \quad \boldsymbol{X}\mathbf{1} = \mathbf{0}, \;\; \frac{1}{N}\boldsymbol{X}\boldsymbol{X}^\top = \boldsymbol{I}. \tag{2.29}$$

The constraint $\boldsymbol{X}\mathbf{1} = \sum_{i=1}^N \mathbf{x}_i = \mathbf{0}$ removes the translational degree of freedom of the embedding vectors and centers them at the origin, while and the constraint $\frac{1}{N}\boldsymbol{X}\boldsymbol{X}^\top = \frac{1}{N}\sum_{i=1}^N \mathbf{x}_i\mathbf{x}_i^\top$ removes the arbitrary rotational and scaling degrees of freedom of the embedding vectors by requiring them to have unit covariance.

It turns out that the solution of the optimization program in (2.29) is given by the bottom eigenvectors of the following matrix

$$\boldsymbol{M} = (\boldsymbol{I} - \boldsymbol{W})^\top(\boldsymbol{I} - \boldsymbol{W}). \tag{2.30}$$

More precisely, collecting the d eigenvectors of $\boldsymbol{M}$ corresponding to its second to the $(d+1)$-th smallest eigenvalues in a matrix $\boldsymbol{V} \in \mathbb{R}^{N\times d}$, the d-dimensional representation of the N data points are given by the rows of $\boldsymbol{V}$.

When data lie in $n \geq 2$ manifolds that are sufficiently separated such that the points from different manifolds are not connected to each other in the neighborhood

graph, the dimension of the nullspace of $\boldsymbol{M}$ is at least n [87]. The work of [87] uses the bottom eigenvectors of $\boldsymbol{M}$ to find the components of the graph, i.e., the manifolds. However, the drawback of this approach is that there is no guarantee that the dimension of the nullspace can be in general greater than the umber of components of the graph, hence not all vectors in the nullspace of $\boldsymbol{M}$ are informative about the memberships of the data to the manifolds. [55] tries to resolve this issue in the case of flat manifolds by analyzing the variance of the vectors in the nullspace of $\boldsymbol{M}$, however, the presented result is not theoretically correct.

Chapter 3

Sparse Subspace Clustering

In this section, we consider the problem of clustering a collection of data points that lie in a union of subspaces. We propose an algorithm, called Sparse Subspace Clustering (SSC), to address this problem [40, 41, 45]. The key idea is that, among infinitely many possible representations of a data point in terms of other points, a sparse representation corresponds to selecting a few points from the same subspace. This motivates solving a sparse optimization program whose solution is used in a spectral clustering framework to infer the clustering of data into subspaces.

Since solving the sparse optimization program is in general NP-hard, we consider a convex relaxation and show that, under appropriate conditions on the arrangement of the subspaces and the distribution of data, the proposed minimization program succeeds in recovering the desired sparse representations. The proposed algorithm can be solved efficiently and can handle data points near the intersections of subspaces.

Figure 3.1: Motion segmentation: given feature points on multiple rigidly moving objects tracked in multiple frames of a video (top), the goal is to separate the feature trajectories according to the moving objects (bottom).

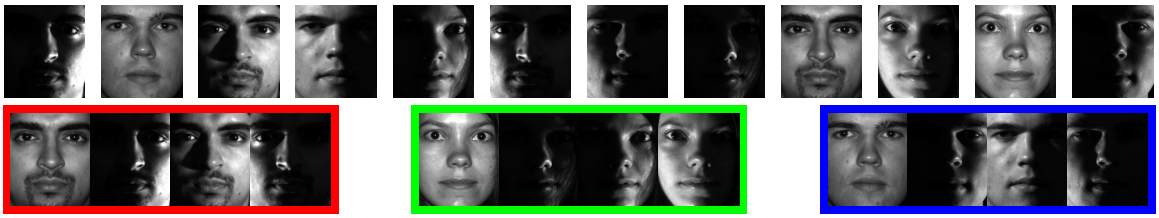


Figure 3.2: Face clustering: given face images of multiple subjects (top), the goal is to find images that belong to the same subject (bottom).

Another key advantage of the proposed algorithm with respect to the state of the art (see Section 3.1 for a brief review) is that it can deal with data nuisances, such as noise, sparse outlying entries, and missing entries, directly by incorporating the model of the data into the sparse optimization program. Moreover, it does not need to know the dimensions of the subspaces a priori. We demonstrate the effectiveness of the proposed algorithm through experiments on synthetic data as well as the two real-world problems of motion segmentation (Fig. 3.1) and face clustering (Fig. 3.2).

Before presenting the proposed framework, we review the existing methods for

subspace clustering.

3.1 A review of subspace clustering algorithms

Existing subspace clustering algorithms can be divided into three main categories: iterative/algebraic, statistical, and spectral clustering-based methods.

3.1.1 Iterative/algebraic methods

Iterative approaches, such as K-subspaces [15, 106] and median K-flats [123] alternate between assigning points to subspaces and fitting a subspace to each cluster. The main drawbacks of such approaches are that they generally require to know the number and dimensions of subspaces, and that they are sensitive to initialization. Factorization-based approaches such as [25, 51, 66] find an initial segmentation by thresholding the entries of a similarity matrix built from the factorization of the data matrix. These methods are provably correct when the subspaces are independent, but fail when this assumption is violated. In addition, they are sensitive to noise and outliers in the data.

Algebraic approaches such as Generalized Principal Component Analysis (GPCA) [78, 111], fit the data with a polynomial whose gradient at a point gives the normal

vector to the subspace containing that point. While GPCA can deal with subspaces of different dimensions, it is sensitive to noise and outliers, and its complexity increases exponentially in terms of the number and the dimensions of the subspaces.

3.1.2 Statistical methods

Iterative statistical approaches, such as Mixtures of Probabilistic PCA (MPPCA) [100], Multi-Stage Learning (MSL) [97], or [59], assume that the distribution of the data inside each subspace is Gaussian and alternate between data clustering and subspace estimation by applying the Expectation Maximization (EM) algorithm. The main drawbacks of these methods are that they generally need to know the number and dimensions of the subspaces, and that they are sensitive to initialization. Robust statistical approaches, such as Random Sample Consensus (RANSAC) [47], fit a subspace of dimension d to randomly chosen subsets of d points until the number of inliers is large enough. The inliers are then removed, and the process is repeated to find a second subspace, and so on. RANSAC can deal with noise and outliers, and does not need to know the number of subspaces. However, the dimensions of the subspaces must be known and equal. In addition, the complexity of the algorithm increases exponentially in the dimension of the subspaces.

Information-theoretic statistical approaches, such as Agglomerative Lossy Compression (ALC) [89], look for the segmentation of the data that minimizes the coding length needed to fit the points with a mixture of degenerate Gaussians up to a given

distortion. As this minimization problem is NP-hard, a suboptimal solution is found by first assuming that each point forms its own group, and then iteratively merging pairs of groups to reduce the coding length. ALC can handle noise and outliers in the data. While, in principle, it does not need to know the number and dimensions of the subspaces, the number of subspaces found by the algorithms is dependent on the choice of a distortion parameter. In addition, there is no theoretical proof for the optimality of the agglomerative algorithm.

3.1.3 Spectral clustering-based methods

Local spectral clustering-based approaches such as Local Subspace Affinity (LSA) [118], Locally Linear Manifold Clustering (LLMC) [55], Spectral Local Best-fit Flats (SLBF) [124], and [122] use local information around each point to build a similarity between pairs of points. The segmentation of the data is then obtained by applying spectral clustering [84, 113] to the similarity matrix. These methods have difficulties dealing with points near the intersection of two subspaces, because the neighborhood of a point can contain points from different subspaces. In addition, they are sensitive to the right choice of the neighborhood size to compute the local information at each point.

Global spectral clustering-based approaches try to resolve these issues by building better similarities between data points using global information. Spectral Curvature Clustering (SCC) [22] uses multi-way similarities that capture the curvature of a col-

lection of points within an affine subspace. SCC can deal with noisy data but requires to know the number and dimensions of subspaces and assumes that subspaces have the same dimension. In addition, the complexity of building the multi-way similarity grows exponentially with the dimensions of the subspaces, hence, in practice, a sampling strategy is employed to reduce the computational cost. Using advances in sparse [21,33,99] and low-rank [18,19,90] recovery algorithms, Sparse Subspace Clustering (SSC) [40,41,94], Low-Rank Recovery (LRR) [75–77], and Low-Rank Subspace Clustering (LRSC) [46] algorithms pose the clustering problem as finding a sparse or low-rank representation of the data in the dictionary of the data itself. The solution of the corresponding global optimization algorithm is then used to build a similarity graph from which the segmentation of the data is obtained. The advantages of these methods with respect to most state-of-the-art algorithms are that they can handle noise and outliers in data, and that they do not need to know the dimensions and, in principle, the number of subspaces a priori.

3.2 Sparse subspace clustering algorithm

In this section, we introduce the sparse subspace clustering (SSC) algorithm for clustering a collection of multi-subspace data using sparse representation techniques. We motivate and formulate the algorithm for data points that perfectly lie in a union of linear subspaces. In the next section, we generalize the algorithm to deal with data

nuisances such as noise, sparse outlying entries, and missing entries as well as the more general class of affine subspaces.

Let $\{\mathcal{S}_\ell\}_{\ell=1}^n$ be an arrangement of n linear subspaces of $\mathbb{R}^D$ of dimensions $\{d_\ell\}_{\ell=1}^n$. Consider a given collection of N noise-free data points $\{\boldsymbol{y}_i\}_{i=1}^N$ that lie in the union of the n subspaces. Denote the matrix containing all the data points as

$$\boldsymbol{Y} \triangleq \begin{bmatrix} \boldsymbol{y}_1 & \dots & \boldsymbol{y}_N \end{bmatrix} = \begin{bmatrix} \boldsymbol{Y}_1 & \dots & \boldsymbol{Y}_n \end{bmatrix} \boldsymbol{\Gamma}, \tag{3.1}$$

where $\boldsymbol{Y}_\ell \in \mathbb{R}^{D \times N_\ell}$ is the matrix of the $N_\ell > d_\ell$ points that lie in $\mathcal{S}_\ell$ and $\boldsymbol{\Gamma} \in \mathbb{R}^{N \times N}$ is an unknown permutation matrix. We assume that we do not know a priori the bases of the subspaces nor do we know which data points belong to which subspace. The *subspace clustering* problem refers to the problem of finding the number of subspaces, their dimensions, a basis for each subspace, and the segmentation of the data from $\boldsymbol{Y}$.

To address the subspace clustering problem, we propose an algorithm that consists of two steps. In the first step, for each data point, we find a few other points that belong to the same subspace. To do so, we propose a global sparse optimization program whose solution encodes information about the memberships of data points to the underlying subspace of each point. In the second step, we use this information in a spectral clustering framework to infer the clustering of the data.

3.2.1 Sparse optimization program

Our algorithm takes advantage of what we refer to as the *self-expressiveness property* of the data, i.e., *each data point in a union of subspaces can be efficiently reconstructed by a combination of other points in the dataset.* More precisely, $\boldsymbol{y}_i \in \mathcal{S}_\ell$ can be written as

$$\boldsymbol{y}_i = \boldsymbol{Y}\boldsymbol{c}_i, \quad c_{ii} = 0, \tag{3.2}$$

where $\boldsymbol{c}_i \triangleq \begin{bmatrix} c_{i1} & c_{i2} & \dots & c_{iN} \end{bmatrix}^\top$ and the constraint $c_{ii} = 0$ eliminates the trivial solution of writing a point as a linear combination of itself. In other words, the matrix of data points $\boldsymbol{Y}$ is a self-expressive dictionary with respect to which each point can be written as a linear combination of other points.

Notice that the representation of $\boldsymbol{y}_i$ in the dictionary $\boldsymbol{Y}$ is *not unique* in general. This comes from the fact that the number of data points in a subspace is assume be to be greater than its dimension, i.e., $N_\ell > d_\ell$. As a result, each $\boldsymbol{Y}_\ell$, and consequently $\boldsymbol{Y}$, has a non-trivial nullspace giving rise to infinitely many representations of each data point.The key observation in our proposed algorithm is that among all solutions of (3.2), *there exists a sparse representation, $\boldsymbol{c}_i$, whose nonzero elements correspond to data points from the same subspace as that of $\boldsymbol{y}_i$.* We refer to such a representation as a *subspace-sparse representation.*

More specifically, a data point $\boldsymbol{y}_i$ that lies in the d_ℓ-dimensional subspace $\mathcal{S}_\ell$ can be written as a linear combination of d_ℓ other points in general directions from $\mathcal{S}_\ell$. As a result, ideally, a sparse representation of a data point finds points from the same

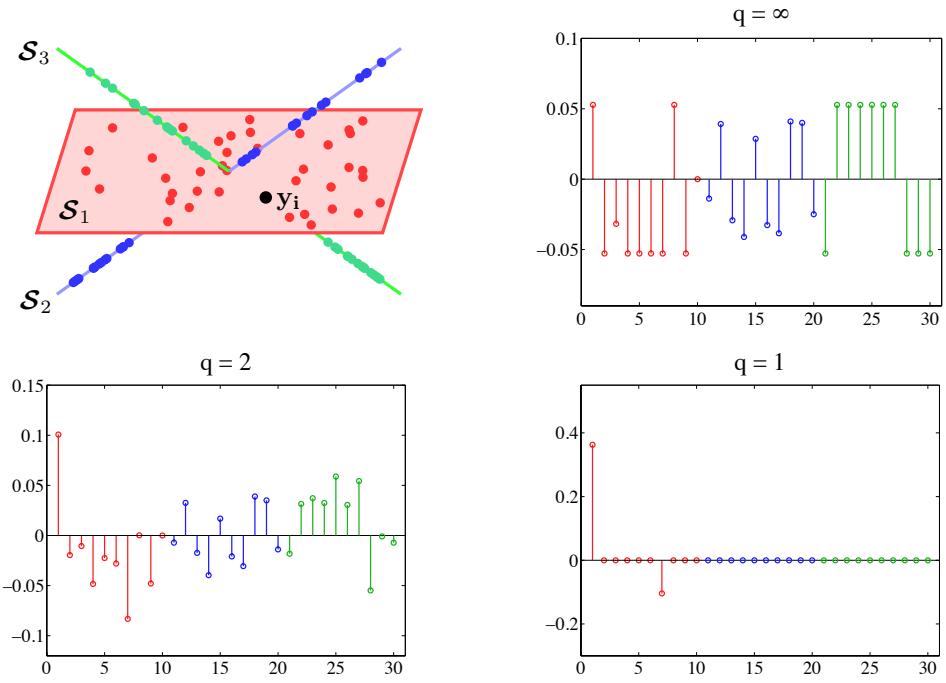


Figure 3.3: Three subspaces in $\mathbb{R}^3$ with 10 data points in each subspace, ordered such that the fist and the last 10 points belong to $\mathcal{S}_1$ and $\mathcal{S}_3$, respectively. The solution of the ℓ_q-minimization program in (3.3) for $\boldsymbol{y}_i$ lying in $\mathcal{S}_1$ for $q = 1, 2, \infty$ is shown. Note that as the value of q decreases, the sparsity of the solution increases. For $q = 1$, the solution corresponds to choosing two other points lying in $\mathcal{S}_1$.

subspace where the number of the nonzero elements corresponds to the dimensionality of the underlying subspace.

For a system of equations such as (3.2) with infinitely many solutions, one can restrict the set of solutions by minimizing an objective function such as the ℓ_q-norm of the solution[1] as

$$\min \|\boldsymbol{c}_i\|_q \quad \text{s.t.} \quad \boldsymbol{y}_i = \boldsymbol{Y}\boldsymbol{c}_i,\ c_{ii} = 0. \tag{3.3}$$

Different choices of q have different effects in the obtained solution. Typically, by decreasing the value of q from infinity toward zero, the sparsity of the solution in-

[1]Recall that the ℓ_q-norm of $\boldsymbol{c}_i \in \mathbb{R}^N$ is defined as $\|\boldsymbol{c}_i\|_q \triangleq (\sum_{j=1}^{N} |c_{ij}|^q)^{\frac{1}{q}}$.

creases, as shown in Figure 3.3. The extreme case of $q = 0$ corresponds to the general NP-hard problem [1] of finding the sparsest representation of the given point, as the ℓ_0-norm counts the number of nonzero elements of the solution. Since we are interested in efficiently finding a non-trivial sparse representation of $\boldsymbol{y}_i$ in the dictionary $\boldsymbol{Y}$, we consider minimizing the tightest convex relaxation of the ℓ_0-norm, i.e.,

$$\min \|\boldsymbol{c}_i\|_1 \quad \text{s.t.} \quad \boldsymbol{y}_i = \boldsymbol{Y}\boldsymbol{c}_i,\ c_{ii} = 0, \tag{3.4}$$

which can be solved efficiently using convex programming tools [13, 14, 68] and is known to prefer sparse solutions [21, 33, 99]. We can also rewrite the sparse optimization program (3.4) for all data points in the matrix form as

$$\min \|\boldsymbol{C}\|_1 \quad \text{s.t.} \quad \boldsymbol{Y} = \boldsymbol{Y}\boldsymbol{C}, \quad \operatorname{diag}(\boldsymbol{C}) = \boldsymbol{0}, \tag{3.5}$$

where $\boldsymbol{C} \in \mathbb{R}^{N \times N}$ is the matrix of the sparse coefficients whose i-th column corresponds to the sparse representation of $\boldsymbol{y}_i$, i.e., $\boldsymbol{C} \triangleq \begin{bmatrix} \boldsymbol{c}_1 & \boldsymbol{c}_2 & \dots & \boldsymbol{c}_N \end{bmatrix}$, and $\operatorname{diag}(\boldsymbol{C}) \in \mathbb{R}^N$ is the vector of the diagonal elements of $\boldsymbol{C}$.

In Section 3.5, we study conditions under which the solution of (3.5) corresponds to a subspace-sparse representation of each data point. Next, assuming that $\boldsymbol{C}$ is subspace-sparse, we use $\boldsymbol{C}$ to infer the clustering of the data.

3.2.2 Clustering using sparse coefficients

After solving the proposed optimization program in (3.5), we obtain a sparse representation for each data point whose nonzero elements ideally correspond to points

from the same subspace. The next step of the algorithm is to infer the segmentation of the data into different subspaces using the sparse coefficients.

To address this problem, we build a weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$, where $\mathcal{V}$ denotes the set of N nodes of the graph, which corresponds to the N data points, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes the set of the edges between the nodes. $\boldsymbol{W} \in \mathbb{R}^{N \times N}$ is a symmetric non-negative similarity matrix representing the weights of the edges, i.e., node i is connected to node j by an edge whose weight is equal to w_{ij}. An ideal similarity matrix $\boldsymbol{W}$, hence an ideal similarity graph $\mathcal{G}$, is one in which nodes that correspond to points from the same subspace are connected to each other and there are no edges between nodes that correspond to points in different subspaces.

Note that the sparse optimization program ideally recovers to a subspace-sparse representation of each point, i.e., a representation whose nonzero elements correspond to points from the same subspace of the given data point. This provides an immediate choice of the similarity matrix as $\boldsymbol{W} = |\boldsymbol{C}| + |\boldsymbol{C}|^\top$. In other words, each node i connects itself to a node j by and edge whose weight is equal to $|c_{ij}| + |c_{ji}|$. The reason for the symmetrization is that, in general, a data point $\boldsymbol{y}_i \in \mathcal{S}_\ell$ can write itself as a linear combination of some points including $\boldsymbol{y}_j \in \mathcal{S}_\ell$. However, $\boldsymbol{y}_j$ may not necessarily choose $\boldsymbol{y}_i$ in its sparse representation. By this particular choice of the weight, we make sure that nodes i and j get connected to each other if either $\boldsymbol{y}_i$ or $\boldsymbol{y}_j$ is in the sparse representation of the other.

The similarity graph built in this way has ideally n connected components corre-

sponding to the n subspaces, i.e.,

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{W}_1 & \cdots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{0} & \cdots & \boldsymbol{W}_n \end{bmatrix} \boldsymbol{\Gamma}, \tag{3.6}$$

where $\boldsymbol{W}_\ell$ is the similarity matrix of data points in $\mathcal{S}_\ell$. Clustering of data into subspaces follows then by applying the spectral clustering of [84] to $\mathcal{G}$. More specifically, we first compute the degree matrix of the graph as $\boldsymbol{D} = \text{diag}(\sum_{j=1}^{N} w_{1j}, \ldots, \sum_{j=1}^{N} w_{Nj})$. We then form the symmetric normalized Laplacian matrix of the graph as $\boldsymbol{L} = \boldsymbol{I} - \boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2}$and find the n eigenvectors of $\boldsymbol{L}$ associated to its smallest eigenvalues. Denoting the i-th bottom eigenvector as $\boldsymbol{v}_i$, we form the matrix $\boldsymbol{V} = \begin{bmatrix} \boldsymbol{v}_1 & \cdots & \boldsymbol{v}_n \end{bmatrix} \in \mathbb{R}^{N \times n}$, normalize the rows of $\boldsymbol{V}$ to have unit Euclidean norms, and finally obtain the clustering of the data into n subspaces by applying the Kmeans algorithm [34] to $\boldsymbol{V}$.

Remark 3 (Normalization of sparse coefficients) *An optional step that often improves the spectral clustering result is to normalize the sparse coefficients as* $\boldsymbol{c}_i \leftarrow \boldsymbol{c}_i / \|\boldsymbol{c}_i\|_\infty$ *prior to building the similarity graph. This helps to better deal with different norms of data points. More specifically, if a data point with a large Euclidean norm selects a few points with small Euclidean norms, then the values of the nonzero coefficients will generally be large. On the other hand, if a data point with a small Euclidean norm selects a few points with large Euclidean norms, then the values of the nonzero coefficients will generally be small. Since spectral clustering puts more*

Algorithm 1 : Sparse Subspace Clustering (SSC)

Input: A set of points $\{\boldsymbol{y}_i\}_{i=1}^N$ lying in a union of n linear subspaces $\{\mathcal{S}_i\}_{i=1}^n$.

1: Solve the sparse optimization program:

$$\boldsymbol{C}^* = \text{argmin}_{\boldsymbol{C}} \|\boldsymbol{C}\|_1 \quad \text{s.t.} \quad \boldsymbol{Y} = \boldsymbol{Y}\boldsymbol{C}, \quad \text{diag}(\boldsymbol{C}) = \boldsymbol{0}.$$

2: Normalize the columns of $\boldsymbol{C}^*$ as $\boldsymbol{c}_i^* \leftarrow \frac{\boldsymbol{c}_i^*}{\|\boldsymbol{c}_i^*\|_\infty}$.

3: Form a similarity graph with N nodes representing the data points. Set the weights on the edges between the nodes by $\boldsymbol{W} = |\boldsymbol{C}^*| + |\boldsymbol{C}^*|^\top$.

4: Apply the spectral clustering algorithm of [84] to the similarity graph with weights $\boldsymbol{W}$.

Output: Segmentation of the data: $\boldsymbol{Y}_1, \boldsymbol{Y}_2, \ldots, \boldsymbol{Y}_n$.

emphasis on keeping the stronger connections in the graph, by the normalization step we make sure that the largest edge weights for all the nodes are of the same scale.

In summary, the SSC algorithm for clustering of data points that perfectly lie in a union of linear subspaces is shown in Algorithm 1. Note that an advantage of spectral clustering, which will be shown in the experimental results, is that it provides robustness with respect to a few errors in the sparse representations of data points. In other words, as long as edges between points in different subspaces are weak, spectral clustering can find the correct segmentation of the data.

Remark 4 *In principle, SSC does not need to know the number of subspaces. More specifically, under the conditions of the theoretical results in Section 3.5, in the simi-*

larity graph there will be no connections between points in different subspaces. Thus, one can determine the number of subspaces by finding the number of connected components of the graph $\mathcal{G}$, which is given by $n = \dim(\ker(\boldsymbol{L}))$, as shown in [113]. However, when there are connections between points in different subspaces, other model selection techniques should be employed (see e.g., [16]).

3.3 Practical extensions

In real-world problems, data are often corrupted by noise and sparse outlying entries due to measurement/process noise and ad-hoc data collection techniques, hence, do not lie perfectly in a union of subspaces. For instance, in the motion segmentation problem, because of the malfunctioning of the tracker, feature trajectories can be corrupted by noise or can have entries with large errors [89]. Similarly, in clustering of human faces, images can be corrupted by errors due to specularities, cast shadows, and occlusions [117]. On the other hand, data points may have missing entries, e.g., when the tracker loses tracks of some feature points in a video due to occlusions [110]. Finally, data may lie in a more general class of a union of affine subspaces, which includes linear subspaces as a special case.

In this section, we generalize the SSC algorithm, which we introduced in the previous section for clustering of data lying perfectly in a union of linear subspaces, to deal with the aforementioned challenges. Unlike state-of-the-art methods, which

require to run a separate algorithm first to correct the errors in the data [89, 110], we deal with these problems in a unified framework by incorporating a model for the corruption into the sparse optimization program. Thus, the sparse coefficients again encode information about the membership of the data to the subspaces, which are used in a spectral clustering framework, as before.

3.3.1 Noise and sparse outlying entries

In this section, we consider clustering of data points that are contaminated with sparse outlying entries and noise. Let

$$\boldsymbol{y}_i = \boldsymbol{y}_i^0 + \boldsymbol{e}_i^0 + \boldsymbol{z}_i^0 \tag{3.7}$$

be the i-th data point that is obtained by corrupting an error-free point $\boldsymbol{y}_i^0$, which perfectly lies in a subspace, with a vector of sparse outlying entries $\boldsymbol{e}_i^0 \in \mathbb{R}^D$ that has only a few large nonzero elements, i.e., $\|\boldsymbol{e}_i^0\|_0 \leq k$ for some integer k, and with a noise $\boldsymbol{z}_i^0 \in \mathbb{R}^D$ whose norm is bounded as $\|\boldsymbol{z}_i^0\|_2 \leq \zeta$ for some $\zeta > 0$. Since error-free data points perfectly lie in a union of subspaces, using the self-expressiveness property, we can reconstruct $\boldsymbol{y}_i^0 \in \mathcal{S}_\ell$ in terms of other error-free points as

$$\boldsymbol{y}_i^0 = \sum_{j \neq i} c_{ij} \boldsymbol{y}_j^0. \tag{3.8}$$

Rewriting $\boldsymbol{y}_i^0$ using (3.7) in terms of the corrupted point $\boldsymbol{y}_i$, the sparse outlying entries vector $\boldsymbol{e}_i^0$, and the noise vector $\boldsymbol{z}_i^0$ and substituting it into (3.8), we obtain

$$\boldsymbol{y}_i = \sum_{j \neq i} c_{ij} \boldsymbol{y}_j + \boldsymbol{e}_i + \boldsymbol{z}_i, \tag{3.9}$$

where the vectors $\boldsymbol{e}_i \in \mathbb{R}^D$ and $\boldsymbol{z}_i \in \mathbb{R}^D$ are defined as

$$\boldsymbol{e}_i \triangleq \boldsymbol{e}_i^0 - \textstyle\sum_{j \neq i} c_{ij} \boldsymbol{e}_j^0, \tag{3.10}$$

$$\boldsymbol{z}_i \triangleq \boldsymbol{z}_i^0 - \textstyle\sum_{j \neq i} c_{ij} \boldsymbol{z}_j^0. \tag{3.11}$$

Notice that (3.9) has still a sparse solution $\boldsymbol{c}_i$, because $\boldsymbol{y}_i^0 \in \mathcal{S}_\ell$ can be expressed as a linear combination of $\{\boldsymbol{y}_j^0\}_{j \neq i}$ with at most d_ℓ nonzero entries, i.e., $\|\boldsymbol{c}_i\|_0 \leq d_\ell$. Moreover, $\boldsymbol{e}_i$ is also sparse, because $\|\boldsymbol{e}_i^0\|_0 \leq k$ and $\|\boldsymbol{c}_i\|_0 \leq \max_\ell d_\ell$, hence using (3.10), $\|\boldsymbol{e}_i\|_0 \leq k(1+\max_\ell d_\ell)$, which we assume to be much smaller than N. Similarly, $\boldsymbol{z}_i$ is a vector of noise since it is linear combination of at most $1+\max_\ell d_\ell$ noise vectors in (3.11).

Collecting $\boldsymbol{e}_i$ and $\boldsymbol{z}_i$ as columns of matrices $\boldsymbol{E}$ and $\boldsymbol{Z}$, respectively, we can rewrite (3.9) in the matrix form as

$$\boldsymbol{Y} = \boldsymbol{Y}\boldsymbol{C} + \boldsymbol{E} + \boldsymbol{Z}, \quad \operatorname{diag}(\boldsymbol{C}) = \boldsymbol{0}. \tag{3.12}$$

Our objective is then to find a solution $(\boldsymbol{C}, \boldsymbol{E}, \boldsymbol{Z})$ for (3.12), where $\boldsymbol{C}$ corresponds to a sparse coefficient matrix, $\boldsymbol{E}$ corresponds to a matrix of sparse outlying entries, and $\boldsymbol{Z}$ is a noise matrix. To do so, we propose to solve the following optimization

program

$$\begin{aligned} \min_{(\boldsymbol{C},\boldsymbol{E},\boldsymbol{Z})} \quad & \|\boldsymbol{C}\|_1 + \lambda_e \|\boldsymbol{E}\|_1 + \frac{\lambda_z}{2}\|\boldsymbol{Z}\|_F^2 \\ \text{s.t.} \quad & \boldsymbol{Y} = \boldsymbol{Y}\boldsymbol{C} + \boldsymbol{E} + \boldsymbol{Z}, \;\; \text{diag}(\boldsymbol{C}) = \boldsymbol{0}, \end{aligned} \tag{3.13}$$

where the ℓ_1-norm promotes sparsity of the columns of $\boldsymbol{C}$ and $\boldsymbol{E}$ while the Frobenius norm promotes having small entries in the columns of $\boldsymbol{Z}$. The two parameters $\lambda_e > 0$ and $\lambda_z > 0$ balance the three terms in the objective function. Note that the optimization program in (3.13) is convex with respect to the optimization variables $(\boldsymbol{C}, \boldsymbol{E}, \boldsymbol{Z})$, hence, can be solved using convex programming tools (see Section 3.4 for implementation details).

When data are corrupted only by noise, we can eliminate $\boldsymbol{E}$ from the optimization program in (3.13). On the other hand, when the data are corrupted only by sparse outlying entries, we can eliminate $\boldsymbol{Z}$ in (3.13). In practice, however, $\boldsymbol{E}$ can also deal with small errors due to noise. The following proposition suggests setting $\lambda_z = \alpha_z/\mu_z$ and $\lambda_e = \alpha_e/\mu_e$, where $\alpha_z, \alpha_e > 1$ and

$$\mu_z \triangleq \min_i \max_{j \neq i} |\boldsymbol{y}_i^\top \boldsymbol{y}_j|, \qquad \mu_e \triangleq \min_i \max_{j \neq i} \|\boldsymbol{y}_j\|_1. \tag{3.14}$$

The proof of the following proposition is provided in the appendix of this chapter.

Proposition 1 *Consider the optimization program* (3.13)*. Without the term* $\boldsymbol{Z}$*, if* $\lambda_e \leq 1/\mu_e$*, then there exists at least one data point* $\boldsymbol{y}_\ell$ *for which in the optimal solution we have* $(\boldsymbol{c}_\ell, \boldsymbol{e}_\ell) = (\boldsymbol{0}, \boldsymbol{y}_\ell)$*. Also, without the term* $\boldsymbol{E}$*, if* $\lambda_z \leq 1/\mu_z$*, then there exists at least one data point* $\boldsymbol{y}_\ell$ *for which* $(\boldsymbol{c}_\ell, \boldsymbol{z}_\ell) = (\boldsymbol{0}, \boldsymbol{y}_\ell)$*.*

After solving the proposed optimization programs (see Section 3.4 for details), we use $\boldsymbol{C}$ to build a similarity graph and infer the clustering of the data using spectral clustering. Thus, by incorporating the data corruption model into the sparse optimization program, we can deal with clustering of corrupted data, as before, without explicitly running a separate algorithm to correct the errors in the data [89, 110].

3.3.2 Missing entries

We consider now the clustering of incomplete data, where some of the entries of a subset of data points are missing. Note that when only a small fraction of entries of each data point is missing, clustering of incomplete data can be cast as clustering of data with sparse outlying entries. More precisely, one can fill in the missing entries of each data point with random values, hence obtain data points with sparse outlying entries. Then clustering of data follows by solving (3.13) and applying spectral clustering to the graph built using the obtained sparse coefficients. However, the drawback of this approach is that it disregards the fact that we know the locations of the missing entries in the data matrix.

It is possible, in some cases, to cast the clustering of data with missing entries as clustering of complete data. To see this, consider a collection of data points $\{\boldsymbol{y}_i\}_{i=1}^N$ in $\mathbb{R}^D$. Let $J_i \subset \{1, \ldots, D\}$ denote indices of the known entries of $\boldsymbol{y}_i$ and define $J \triangleq \bigcap_{i=1}^N J_i$. Thus, for every index in J, all data points have known entries. When the size of J, denoted by $|J|$, is not small relative to the ambient space dimension,

D, we can project the data, hence, the original subspaces, into a subspace spanned by the columns of the identity matrix indexed by J and apply the SSC algorithm to the obtained complete data. In other words, we can only keep the rows of $\boldsymbol{Y}$ indexed by J, obtain a new data matrix of complete data $\bar{\boldsymbol{Y}} \in \mathbb{R}^{|J| \times N}$, and solve the sparse optimization program (3.13). We can then infer the clustering of the data by applying spectral clustering to the graph built using the sparse coefficient matrix.

Note that the approach described above is based on the assumption that J is nonempty. Addressing the problem of subspace clustering with missing entries when J is empty or has a small size is the subject of the future research.

3.3.3 Affine subspaces

In some real-world problems, data lie in a union of affine rather than linear subspaces. For instance, the motion segmentation problem involves clustering of data that lie in a union of 3-dimensional affine subspaces [102, 110]. However, most subspace clustering algorithms deal with this problem as if the data lie in a union of 4-dimensional linear subspaces. This comes from the fact that a d_ℓ-dimensional affine subspace $\mathcal{S}_\ell$ can be considered as a subset of a $(d_\ell + 1)$-dimensional linear subspace that includes $\mathcal{S}_\ell$ and the origin. However, this has the drawback of increasing the dimension of the possible intersection of subspaces, which in some cases can result in indistinguishability of subspaces from each other. For example, two different lines $x = -1$ and $x = +1$ in the x-y plane form the same 2-dimensional linear subspace

after including the origin, hence become indistinguishable.

To directly deal with affine subspaces, we use the fact that any data point $\boldsymbol{y}_i$ in an affine subspace $\mathcal{S}_\ell$ of dimension d_ℓ can be written as an affine combination of $d_\ell+1$ other points from $\mathcal{S}_\ell$. In other words, a sparse solution of

$$\boldsymbol{y}_i = \boldsymbol{Y}\boldsymbol{c}_i, \quad \mathbf{1}^\top \boldsymbol{c}_i = 1, \;\; c_{ii} = 0, \tag{3.15}$$

corresponds to $d_\ell+1$ other points that belong to $\mathcal{S}_\ell$. Thus, to cluster data points lying close to a union of affine subspaces and contaminated by noise and sparse outlying entries as in (3.9), we propose to solve the sparse optimization program

$$\begin{aligned} \min \;\; & \|\boldsymbol{C}\|_1 + \lambda_e \|\boldsymbol{E}\|_1 + \frac{\lambda_z}{2}\|\boldsymbol{Z}\|_F^2 \\ \text{s.t.} \;\; & \boldsymbol{Y} = \boldsymbol{Y}\boldsymbol{C} + \boldsymbol{E} + \boldsymbol{Z}, \;\; \mathbf{1}^\top \boldsymbol{C} = \mathbf{1}^\top, \;\; \operatorname{diag}(\boldsymbol{C}) = \mathbf{0}. \end{aligned} \tag{3.16}$$

Notice that, in comparison to (3.13) for the case of linear subspaces, the optimization program in (3.16) includes additional linear equality constraints, namely $\mathbf{1}^\top \boldsymbol{C} = \mathbf{1}^\top$. Notice also that (3.16) can deal with linear subspaces as well since a linear subspace is also an affine subspace. In the next section, we address the implementation of the proposed optimization programs (3.13) and (3.16).

3.4 Solving the sparse optimization programs

Note that the convex programs, introduced in the previous sections, can be solved using generic convex solvers such as CVX[2]. However, since we need to solve for $O(PN)$ number of variables with $P \in \{N, N+D\}$ depending on the optimization program, generic solvers will have high computational costs as they do not scale well with the dimension, D, and the number of data points, N.

In this section, we study efficient implementations of the proposed sparse optimizations using an Alternating Direction Method of Multipliers (ADMM) method [13,49]. We fist consider the most general optimization program

$$\begin{aligned} &\min_{(\boldsymbol{C},\boldsymbol{E},\boldsymbol{Z})} \ \|\boldsymbol{C}\|_1 + \lambda_e \|\boldsymbol{E}\|_1 + \frac{\lambda_z}{2}\|\boldsymbol{Z}\|_F^2 \\ &\text{s.t.} \quad \boldsymbol{Y} = \boldsymbol{Y}\boldsymbol{C} + \boldsymbol{E} + \boldsymbol{Z}, \ \ \boldsymbol{C}^\top \mathbf{1} = \mathbf{1}, \quad \text{diag}(\boldsymbol{C}) = \mathbf{0}, \end{aligned} \tag{3.17}$$

and present an ADMM algorithm to solve it. We then derive implementations of other practical sparse optimizations, considered in the experiments in Section 3.8, using the presented algorithm.

First, note that using the equality constraint in (3.17), we can eliminate $\boldsymbol{Z}$ from

[2]CVX is a Matlab-based software for convex programming and can be downloaded from http://cvxr.com.

the optimization program and equivalently solve

$$\begin{aligned} \min_{(\boldsymbol{C},\boldsymbol{E})} \ & \|\boldsymbol{C}\|_1 + \lambda_e \|\boldsymbol{E}\|_1 + \frac{\lambda_z}{2}\|\boldsymbol{Y} - \boldsymbol{Y}\boldsymbol{C} - \boldsymbol{E}\|_F^2 \\ \text{s.t.} \quad & \boldsymbol{C}^\top \mathbf{1} = \mathbf{1}, \quad \operatorname{diag}(\boldsymbol{C}) = \mathbf{0}. \end{aligned} \tag{3.18}$$

The overall procedure of the ADMM algorithm is to introduce appropriate auxiliary variables into the optimization program, augment the constraints into the objective function, and iteratively minimize the Lagrangian with respect to the primal variables and maximize it with respect to the Lagrange multipliers. With an abuse of notation, throughout this section, we denote by $\operatorname{diag}(\boldsymbol{C})$ both a vector whose elements are the diagonal entries of $\boldsymbol{C}$ and a diagonal matrix whose diagonal elements are the diagonal entries of $\boldsymbol{C}$.

To start, we introduce an auxiliary matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ and consider the optimization program

$$\begin{aligned} \min_{(\boldsymbol{C},\boldsymbol{E},\boldsymbol{A})} \ & \|\boldsymbol{C}\|_1 + \lambda_e \|\boldsymbol{E}\|_1 + \frac{\lambda_z}{2}\|\boldsymbol{Y} - \boldsymbol{Y}\boldsymbol{A} - \boldsymbol{E}\|_F^2 \\ \text{s.t.} \quad & \boldsymbol{A}^\top \mathbf{1} = \mathbf{1}, \quad \boldsymbol{A} = \boldsymbol{C} - \operatorname{diag}(\boldsymbol{C}). \end{aligned} \tag{3.19}$$

whose solution for $(\boldsymbol{C}, \boldsymbol{E})$ coincides with the solution of (3.18). As we will see shortly, introducing $\boldsymbol{A}$ helps to obtain efficient updates on the optimization variables. Next, using a parameter $\rho > 0$, we add to the objective function of (3.19) two penalty terms corresponding to the constraints $\boldsymbol{A}^\top \mathbf{1} = \mathbf{1}$ and $\boldsymbol{A} = \boldsymbol{C} - \operatorname{diag}(\boldsymbol{C})$ and consider the

following optimization program

$$\begin{aligned}\min_{(\boldsymbol{C},\boldsymbol{E},\boldsymbol{A})} \ & \|\boldsymbol{C}\|_1 + \lambda_e\|\boldsymbol{E}\|_1 + \frac{\lambda_z}{2}\|\boldsymbol{Y}-\boldsymbol{Y}\boldsymbol{A}-\boldsymbol{E}\|_F^2 + \frac{\rho}{2}\|\boldsymbol{A}^\top\mathbf{1}-\mathbf{1}\|_2^2 \\ & + \frac{\rho}{2}\|\boldsymbol{A}-(\boldsymbol{C}-\operatorname{diag}(\boldsymbol{C}))\|_F^2 \\ \text{s.t.} \ & \boldsymbol{A}^\top\mathbf{1}=\mathbf{1}, \quad \boldsymbol{A}=\boldsymbol{C}-\operatorname{diag}(\boldsymbol{C}).\end{aligned} \tag{3.20}$$

Note that adding the penalty terms to (3.19) does not change its optimal solution, i.e., both (3.19) and (3.20) have the same solutions, since for any feasible solution of (3.20) that satisfies the constraints, the penalty terms vanishes. However, adding the penalty terms makes the objective function strictly convex in terms of the optimization variables $(\boldsymbol{C},\boldsymbol{E},\boldsymbol{A})$, which allows using the ADMM approach.

Introducing a vector $\boldsymbol{\delta}\in\mathbb{R}^N$ and a matrix $\boldsymbol{\Delta}\in\mathbb{R}^{N\times N}$ of Lagrange multipliers for the two equality constraints in (3.20), we can write the Lagrangian function of (3.20) as

$$\begin{aligned}\mathcal{L}(\boldsymbol{C},\boldsymbol{A},\boldsymbol{E},\boldsymbol{\delta},\boldsymbol{\Delta}) = \ & \|\boldsymbol{C}\|_1 + \lambda_e\|\boldsymbol{E}\|_1 + \frac{\lambda_z}{2}\|\boldsymbol{Y}-\boldsymbol{Y}\boldsymbol{A}-\boldsymbol{E}\|_F^2 + \frac{\rho}{2}\|\boldsymbol{A}^\top\mathbf{1}-\mathbf{1}\|_2^2 \\ & + \frac{\rho}{2}\|\boldsymbol{A}-(\boldsymbol{C}-\operatorname{diag}(\boldsymbol{C}))\|_F^2 + \boldsymbol{\delta}^\top(\boldsymbol{A}^\top\mathbf{1}-\mathbf{1}) + \operatorname{tr}(\boldsymbol{\Delta}^\top(\boldsymbol{A}-\boldsymbol{C}+\operatorname{diag}(\boldsymbol{C}))),\end{aligned} \tag{3.21}$$

where $\operatorname{tr}(\cdot)$ denotes the trace operator of a given matrix. The ADMM approach is an iterative procedure that proceeds as follows. Denote by $(\boldsymbol{C}^{(k)},\boldsymbol{E}^{(k)},\boldsymbol{A}^{(k)})$ the optimization variables at iteration k, and by $(\boldsymbol{\delta}^{(k)},\boldsymbol{\Delta}^{(k)})$ the Lagrange multipliers at iteration k. These variables are updated as:

- Obtain $\boldsymbol{A}^{(k+1)}$ by minimizing $\mathcal{L}$ with respect to $\boldsymbol{A}$, while $(\boldsymbol{C}^{(k)},\boldsymbol{E}^{(k)},\boldsymbol{\delta}^{(k)},\boldsymbol{\Delta}^{(k)})$ are fixed. Note that computing the derivative of $\mathcal{L}$ with respect to $\boldsymbol{A}$ and setting

it to zero, we obtain

$$(\lambda_z \boldsymbol{Y}^\top \boldsymbol{Y} + \rho \boldsymbol{I} + \rho \mathbf{1}\mathbf{1}^\top)\boldsymbol{A}^{(k+1)} = \lambda_z \boldsymbol{Y}^\top (\boldsymbol{Y} - \boldsymbol{E}^{(k)}) + \rho(\mathbf{1}\mathbf{1}^\top + \boldsymbol{C}^{(k)}) - \mathbf{1}\boldsymbol{\delta}^{(k)\top} - \boldsymbol{\Delta}^{(k)}. \tag{3.22}$$

In other words, $\boldsymbol{A}^{(k+1)}$ is obtained by solving an $N \times N$ system of linear equations. When N is not very large, one can use matrix inversion to obtain $\boldsymbol{A}^{(k+1)}$ from (3.22). For large values of N, more efficient approaches such as the conjugate gradient method [14] should be employed to solve for $\boldsymbol{A}^{(k+1)}$.

- Obtain $\boldsymbol{C}^{(k+1)}$ by minimizing $\mathcal{L}$ with respect to $\boldsymbol{C}$, while $(\boldsymbol{A}^{(k)}, \boldsymbol{E}^{(k)}, \boldsymbol{\delta}^{(k)}, \boldsymbol{\Delta}^{(k)})$ are fixed. Note that the update on $\boldsymbol{C}$ also has a closed-form solution given by

$$\boldsymbol{C}^{(k+1)} = \boldsymbol{J} - \operatorname{diag}(\boldsymbol{J}), \quad \text{where } \boldsymbol{J} \triangleq \mathcal{T}_{\frac{1}{\rho}}(\boldsymbol{A}^{(k+1)} + \boldsymbol{\Delta}^{(k)}/\rho), \tag{3.23}$$

where $\mathcal{T}_\eta(\cdot)$ is the shrinkage-thresholding operator acting on each element of the given matrix, and is defined as

$$\mathcal{T}_\eta(v) = (|v| - \eta)_+ \operatorname{sgn}(v). \tag{3.24}$$

The operator $(\cdot)_+$ returns its argument if it is non-negative and returns zero otherwise.

- Obtain $\boldsymbol{E}^{(k+1)}$ by minimizing $\mathcal{L}$ with respect to $\boldsymbol{E}$, while $(\boldsymbol{C}^{(k+1)}, \boldsymbol{A}^{(k+1)}, \boldsymbol{\delta}^{(k)}, \boldsymbol{\Delta}^{(k)})$ are fixed. The update on $\boldsymbol{E}$ can also be computed in closed-form as

$$\boldsymbol{E}^{(k+1)} = \mathcal{T}_{\frac{\lambda_e}{\lambda_z}}(\boldsymbol{Y}\boldsymbol{A}^{(k+1)} - \boldsymbol{Y}), \tag{3.25}$$

- Having $(\boldsymbol{C}^{(k+1)}, \boldsymbol{A}^{(k+1)}, \boldsymbol{E}^{(k+1)}$ fixed, perform a gradient ascent update with the step size of ρ on the Lagrange multipliers as

$$\boldsymbol{\delta}^{(k+1)} = \boldsymbol{\delta}^{(k)} + \rho\,(\boldsymbol{A}^{(k+1)\top}\mathbf{1} - \mathbf{1}), \tag{3.26}$$

$$\boldsymbol{\Delta}^{(k+1)} = \boldsymbol{\Delta}^{(k)} + \rho\,(\boldsymbol{A}^{(k+1)} - \boldsymbol{C}^{(k+1)}). \tag{3.27}$$

These three steps are repeated until convergence is achieved or the number of iterations exceeds a maximum iteration number. Convergence is achieved when we have $\|\boldsymbol{A}^{(k)\top}\mathbf{1}-\mathbf{1}\|_\infty \leq \epsilon$, $\|\boldsymbol{A}^{(k)}-\boldsymbol{C}^{(k)}\|_\infty \leq \epsilon$, $\|\boldsymbol{A}^{(k)}-\boldsymbol{A}^{(k-1)}\|_\infty \leq \epsilon$ and $\|\boldsymbol{E}^{(k)}-\boldsymbol{E}^{(k-1)}\|_\infty \leq \epsilon$, where ϵ denotes the error tolerance for the primal and dual residuals. In practice, the choice of $\epsilon \in \{10^{-4}, 10^{-3}\}$ works well in real experiments. In summary, Algorithm 2 shows the updates for the ADMM implementation of the optimization program (3.17).

In some applications, we may need to solve a smaller version of the optimization program (3.17). For example, in the motion segmentation problem, which we discuss in details in the experiments, data are only corrupted by noise and there are no sparse outlying entries in the data. Hence, we do not need to have $\boldsymbol{E}$ in the optimization program and we will need to solve

$$\begin{aligned} &\min_{(\boldsymbol{C},\boldsymbol{Z})} \ \|\boldsymbol{C}\|_1 + \frac{\lambda_z}{2}\|\boldsymbol{Z}\|_F^2 \\ &\text{s.t.} \quad \boldsymbol{Y} = \boldsymbol{Y}\boldsymbol{C} + \boldsymbol{Z}, \ \ \boldsymbol{C}^\top\mathbf{1} = \mathbf{1}, \quad \operatorname{diag}(\boldsymbol{C}) = \mathbf{0}. \end{aligned} \tag{3.28}$$

In that case, the ADMM updates can be obtained from the previous derivations by simply eliminating the term $\boldsymbol{E}$ and its updating step in Algorithm 2. Also, in the face

Algorithm 2 : Solving (3.17) **via an ADMM Algorithm**

Initialization: Set maxIter $= 10^4$, $k = 0$, and Terminate $\leftarrow$ False. Initialize $\boldsymbol{C}^{(0)}, \boldsymbol{A}^{(0)}, \boldsymbol{E}^{(0)}, \boldsymbol{\delta}^{(0)}$, and $\boldsymbol{\Delta}^{(0)}$ to zero.

1: **while** (Terminate == False) **do**

2: update $\boldsymbol{A}^{(k+1)}$ by solving the following system of linear equations

$$(\lambda_z \boldsymbol{Y}^\top \boldsymbol{Y} + \rho \boldsymbol{I} + \rho \mathbf{1}\mathbf{1}^\top)\boldsymbol{A}^{(k+1)} = \lambda_z \boldsymbol{Y}^\top (\boldsymbol{Y} - \boldsymbol{E}^{(k)}) + \rho(\mathbf{1}\mathbf{1}^\top + \boldsymbol{C}^{(k)}) - \mathbf{1}\boldsymbol{\delta}^{(k)\top} - \boldsymbol{\Delta}^{(k)},$$

3: update $\boldsymbol{C}^{(k+1)}$ as $\boldsymbol{C}^{(k+1)} = \boldsymbol{J} - \text{diag}(\boldsymbol{J})$, where $\boldsymbol{J} \triangleq \mathcal{T}_{\frac{1}{\rho}}(\boldsymbol{A}^{(k+1)} + \boldsymbol{\Delta}^{(k)}/\rho)$,

4: update $\boldsymbol{E}^{(k+1)}$ as $\boldsymbol{E}^{(k+1)} = \mathcal{T}_{\frac{\lambda_e}{\lambda_z}}(\boldsymbol{Y} - \boldsymbol{Y}\boldsymbol{A}^{(k+1)})$,

5: update $\boldsymbol{\delta}^{(k+1)}$ as $\boldsymbol{\delta}^{(k+1)} = \boldsymbol{\delta}^{(k)} + \rho\,(\boldsymbol{A}^{(k+1)\top}\mathbf{1} - \mathbf{1})$,

6: update $\boldsymbol{\Delta}^{(k+1)}$ as $\boldsymbol{\Delta}^{(k+1)} = \boldsymbol{\Delta}^{(k)} + \rho\,(\boldsymbol{A}^{(k+1)} - \boldsymbol{C}^{(k+1)})$,

7: $k \leftarrow k + 1$,

8: **if** ($\|\boldsymbol{A}^{(k)\top}\mathbf{1} - \mathbf{1}\|_\infty \leq \epsilon$ and $\|\boldsymbol{A}^{(k)} - \boldsymbol{C}^{(k)}\|_\infty \leq \epsilon$ and $\|\boldsymbol{A}^{(k)} - \boldsymbol{A}^{(k-1)}\|_\infty \leq \epsilon$ and $\|\boldsymbol{E}^{(k)} - \boldsymbol{E}^{(k-1)}\|_\infty \leq \epsilon$ or ($k \geq$ maxIter) **then**

9: Terminate $\leftarrow$ True

10: **end if**

11: **end while**

Output: Optimal sparse coefficient matrix $\boldsymbol{C}^* = \boldsymbol{C}^{(k)}$.

clustering problem, which we will discuss in details in the experiments, faces lie in a linear subspace and there is no need for the affine constraint of $\boldsymbol{C}^\top \mathbf{1} = \mathbf{1}$. Moreover, since the data are corrupted by sparse outlying entries, we need to have the term

$\boldsymbol{E}$ in the optimization program, which in practice also deals with noise in the data. Thus, we will need to solve

$$\begin{aligned} &\min_{(\boldsymbol{C},\boldsymbol{E})} \ \|\boldsymbol{C}\|_1 + \lambda_e \|\boldsymbol{E}\|_1 \\ &\text{s. t.} \quad \boldsymbol{Y} = \boldsymbol{Y}\boldsymbol{C} + \boldsymbol{E}, \ \ \operatorname{diag}(\boldsymbol{C}) = \boldsymbol{0}. \end{aligned} \tag{3.29}$$

Using derivations similar to the case of solving (3.17) we can obtain the updates for solving the optimization program (3.29), summarized in Algorithm 3.

3.5 Subspace-sparse recovery theory

The underlying assumptions for the success of the SSC algorithm are that 1) the proposed optimization program recovers a subspace-sparse representation of each data point, i.e., a representation whose nonzero elements correspond to the subspace of the given point, and 2) the subspace-sparse representations are such that all the points in the same subspace form a connected component of the similarity graph. We defer the issue of connectedness of the points within each subspace to Section 3.6 and investigate, in this section, conditions under which, for data points that lie in a union of linear subspaces, the sparse optimization program in (3.4) recovers subspace-sparse representations of data points. We investigate recovery conditions for two classes of subspace arrangements: *independent* and *disjoint* subspace models [41].

Definition 1 A collection of subspaces $\{\mathcal{S}_i\}_{i=1}^n$ is said to be *independent* if $\dim(\oplus_{i=1}^n \mathcal{S}_i) = \sum_{i=1}^n \dim(\mathcal{S}_i)$, where $\oplus$ denotes the direct sum operator.

Algorithm 3 : Solving (3.29) **via an ADMM Algorithm**

Initialization: Set maxIter $= 10^4$, $k = 0$, and Terminate $\leftarrow$ False. Initialize $\boldsymbol{C}^{(0)}, \boldsymbol{A}^{(0)}, \boldsymbol{E}^{(0)}, \boldsymbol{\Delta}_1^{(0)}$, and $\boldsymbol{\Delta}_2^{(0)}$ to zero.

while (Terminate == False) **do**

update $\boldsymbol{A}^{(k+1)}$ by solving the following system of linear equations

$$(\boldsymbol{Y}^\top \boldsymbol{Y} + \rho \boldsymbol{I})\boldsymbol{A}^{(k+1)} = \boldsymbol{Y}^\top (\boldsymbol{Y} - \boldsymbol{E}^{(k)} + \boldsymbol{\Delta}_1^{(k)}/\rho) + \rho\, \boldsymbol{C}^{(k)} - \boldsymbol{\Delta}_2^{(k)},$$

update $\boldsymbol{C}^{(k+1)}$ as $\boldsymbol{C}^{(k+1)} = \boldsymbol{J} - \text{diag}(\boldsymbol{J})$, where $\boldsymbol{J} \triangleq \mathcal{T}_{\frac{1}{\rho}}(\boldsymbol{A}^{(k+1)} + \boldsymbol{\Delta}_2^{(k)}/\rho)$,

update $\boldsymbol{E}^{(k+1)}$ as $\boldsymbol{E}^{(k+1)} = \mathcal{T}_{\frac{\lambda_e}{\rho}}(\boldsymbol{Y} - \boldsymbol{Y}\boldsymbol{A}^{(k+1)} + \boldsymbol{\Delta}_1/\rho)$,

update $\boldsymbol{\Delta}_1^{(k+1)}$ as $\boldsymbol{\Delta}_1^{(k+1)} = \boldsymbol{\delta}^{(k)} + \rho\,(\boldsymbol{Y} - \boldsymbol{Y}\boldsymbol{A}^{(k+1)} - \boldsymbol{E}^{(k+1)})$,

update $\boldsymbol{\Delta}_2^{(k+1)}$ as $\boldsymbol{\Delta}_2^{(k+1)} = \boldsymbol{\Delta}_2^{(k)} + \rho\,(\boldsymbol{A}^{(k+1)} - \boldsymbol{C}^{(k+1)})$,

$k \leftarrow k + 1$,

if ($\|\boldsymbol{A}^{(k)} - \boldsymbol{C}^{(k)}\|_\infty \leq \epsilon$ and $\|\boldsymbol{A}^{(k)} - \boldsymbol{A}^{(k-1)}\|_\infty \leq \epsilon$ and $\|\boldsymbol{E}^{(k)} - \boldsymbol{E}^{(k-1)}\|_\infty \leq \epsilon$ or ($k \geq$ maxIter) **then**

Terminate $\leftarrow$ True

end if

end while

Output: Optimal sparse coefficient matrix $\boldsymbol{C}^* = \boldsymbol{C}^{(k)}$.

As an example, the three 1-dimensional subspaces shown in Figure 3.4 (left) are independent since they span the 3-dimensional space and the summation of their dimensions is also 3. On the other hand, the subspaces shown in Figure 3.4 (right)

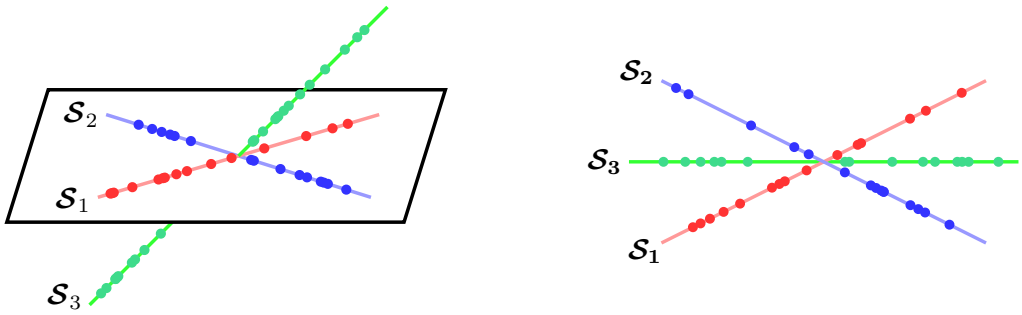


Figure 3.4: Left: the three 1-dimensional subspaces are independent as they span the 3-dimensional space and the summation of their dimensions is also 3. Right: the three 1-dimensional are disjoint as any two subspaces intersect at the origin.

are not independent since they span a 2-dimensional space while the summation of their dimensions is 3.

Definition 2 A collection of subspaces $\{\mathcal{S}_i\}_{i=1}^n$ is said to be *disjoint* if every pair of subspaces intersect only at the origin. In other words, for every pair of subspaces we have $\dim(\mathcal{S}_i \oplus \mathcal{S}_j) = \dim(\mathcal{S}_i) + \dim(\mathcal{S}_j)$.

As an example, both subspace arrangements shown in Figure 3.4 are disjoint since each pair of subspaces intersect at the origin. As a result, based on the above definitions, the notion of disjointness is weaker than independence because an independent subspace model is always disjoint while the converse is not necessarily true.

An important notion that characterizes the relationship of two disjoint subspaces is the smallest principal angle, defined as follows.

Definition 3 The *smallest principal angle* between two subspaces $\mathcal{S}_i$ and $\mathcal{S}_j$, denoted

by θ_{ij}, is defined as

$$\cos(\theta_{ij}) \triangleq \max_{\boldsymbol{v}_i \in \mathcal{S}_i, \boldsymbol{v}_j \in \mathcal{S}_j} \frac{\boldsymbol{v}_i^\top \boldsymbol{v}_j}{\|\boldsymbol{v}_i\|_2 \|\boldsymbol{v}_j\|_2}. \tag{3.30}$$

Note that two disjoint subspaces intersect at the origin, hence their smallest principal angle is greater than zero and $\cos(\theta) \in [0, 1)$.

3.5.1 Independent subspace model

In this section, we consider data points that lie in a union of independent subspaces, which is the underlying model of many subspace clustering algorithms. We show that the ℓ_1-minimization program in (3.4) and more generally the ℓ_q-minimization in (3.3) for $q < \infty$ always recover subspace-sparse representations of data points. More specifically, we show the following result.

Theorem 1 Consider a collection of data points drawn from n independent subspaces $\{\mathcal{S}_i\}_{i=1}^n$ of dimensions $\{d_i\}_{i=1}^n$. Let $\boldsymbol{Y}_i$ denote N_i data points in $\mathcal{S}_i$, where $\text{rank}(\boldsymbol{Y}_i) = d_i$, and let $\boldsymbol{Y}_{-i}$ denote data points in all subspaces except $\mathcal{S}_i$. Then, for every $\mathcal{S}_i$ and every nonzero $\boldsymbol{y}$ in $\mathcal{S}_i$, the ℓ_q-minimization program

$$\begin{bmatrix} \boldsymbol{c}^* \\ \boldsymbol{c}_-^* \end{bmatrix} = \text{argmin} \left\| \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{c}_- \end{bmatrix} \right\|_q \quad \text{s.t.} \quad \boldsymbol{y} = [\boldsymbol{Y}_i \;\; \boldsymbol{Y}_{-i}] \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{c}_- \end{bmatrix}, \tag{3.31}$$

for $q < \infty$, recovers a subspace-sparse representation, i.e., $\boldsymbol{c}^* \neq \boldsymbol{0}$ and $\boldsymbol{c}_-^* = \boldsymbol{0}$.

Proof. For the sake of contradiction assume that $\boldsymbol{c}^*_- \neq 0$. It follows from (3.31) that

$$\boldsymbol{y} - \boldsymbol{Y}_i \boldsymbol{c}^* = \boldsymbol{Y}_{-i} \boldsymbol{c}^*_-. \tag{3.32}$$

Note that the left hand side of equation (3.32) corresponds to a point in the subspace $\mathcal{S}_i$, while the right hand side of (3.32) corresponds to a point in the subspace $\oplus_{j \neq i} \mathcal{S}_j$. By the independence assumption, the two subspaces $\mathcal{S}_i$ and $\oplus_{j \neq i} \mathcal{S}_j$ are also independent hence disjoint and intersect only at the origin. Thus, from (3.32) we must have $\boldsymbol{Y}_{-i} \boldsymbol{c}^*_- = \boldsymbol{0}$ and we obtain $\boldsymbol{y} = \boldsymbol{Y}_i \boldsymbol{c}^*$. In other words, $\begin{bmatrix} \boldsymbol{c}^{*\top} & \boldsymbol{0}^\top \end{bmatrix}^\top$ is a feasible solution of the optimization problem (3.31). Finally, from the assumption that $\boldsymbol{c}^*_- \neq 0$, we have

$$\left\| \begin{bmatrix} \boldsymbol{c}^* \\ \boldsymbol{0} \end{bmatrix} \right\|_q < \left\| \begin{bmatrix} \boldsymbol{c}^* \\ \boldsymbol{c}^*_- \end{bmatrix} \right\|_q, \tag{3.33}$$

which contradicts the optimality of $\begin{bmatrix} \boldsymbol{c}^{*\top} & \boldsymbol{c}^*_- \end{bmatrix}^\top$. Thus, we must have $\boldsymbol{c}^* \neq \boldsymbol{0}$ and $\boldsymbol{c}^*_- = \boldsymbol{0}$, obtaining the desired result. ∎

Note that the subspace-sparse recovery holds without any assumption on the distribution of the data points in each subspace other than $\text{rank}(\boldsymbol{Y}_i) = d_i$. This comes at the price of having a more restrictive model for the subspace arrangements, i.e., an independent subspace model. Next, we will show that for the more general class of disjoint subspaces, under appropriate conditions on the relative configuration of the subspaces as well as the distribution of the data in each subspace, the ℓ_1-minimization in (3.4) recovers subspace-sparse representations of the data points.

3.5.2 Disjoint subspace model

We consider now the more general class of disjoint subspaces and investigate conditions under which the optimization program in (3.4) recovers a subspace-sparse representation of each data point. To that end, we consider a vector $\mathbf{x}$ in the intersection of $\mathcal{S}_i$ with $\oplus_{j\neq i}\mathcal{S}_j$ and let the optimal solution of the ℓ_1-minimization when we restrict the dictionary to data points from $\mathcal{S}_i$ be

$$\boldsymbol{a}_i = \text{argmin}_{\boldsymbol{a}} \|\boldsymbol{a}\|_1 \quad \text{s.t.} \quad \mathbf{x} = \boldsymbol{Y}_i\, \boldsymbol{a}. \tag{3.34}$$

We also let the optimal solution of the ℓ_1-minimization when we restrict the dictionary to points from all subspaces except $\mathcal{S}_i$ be[3]

$$\boldsymbol{a}_{-i} = \text{argmin}_{\boldsymbol{a}} \|\boldsymbol{a}\|_1 \quad \text{s.t.} \quad \mathbf{x} = \boldsymbol{Y}_{-i}\, \boldsymbol{a}. \tag{3.35}$$

We show that if for every nonzero $\mathbf{x}$ in the intersection of $\mathcal{S}_i$ with $\oplus_{j\neq i}\mathcal{S}_j$, the ℓ_1-norm of the solution of (3.34) is strictly smaller than the ℓ_1-norm of the solution of (3.35), i.e.,

$$\forall\, \mathbf{x} \in \mathcal{S}_i \cap (\oplus_{j\neq i}\mathcal{S}_j), \mathbf{x} \neq \mathbf{0} \implies \|\boldsymbol{a}_i\|_1 < \|\boldsymbol{a}_{-i}\|_1, \tag{3.36}$$

then the SSC algorithm recovers subspace-sparse representations of all the data points in $\mathcal{S}_i$. More precisely, we show the following result.

Theorem 2 Consider a collection of data points drawn from n disjoint subspaces $\{\mathcal{S}_i\}_{i=1}^n$ of dimensions $\{d_i\}_{i=1}^n$. Let $\boldsymbol{Y}_i$ denote N_i data points in $\mathcal{S}_i$, where $\text{rank}(\boldsymbol{Y}_i) =$

[3]Notice that $\boldsymbol{a}_i$ and $\boldsymbol{a}_{-i}$ depend on $\mathbf{x}$, $\boldsymbol{Y}_i$, and $\boldsymbol{Y}_{-i}$. Since this dependence is clear from the context, we drop the arguments in $\boldsymbol{a}_i(\mathbf{x}, \boldsymbol{Y}_i)$ and $\boldsymbol{a}_{-i}(\mathbf{x}, \boldsymbol{Y}_{-i})$.

d_i, and let $\boldsymbol{Y}_{-i}$ denote data points in all subspaces except $\mathcal{S}_i$. The ℓ_1-minimization

$$\begin{bmatrix} \boldsymbol{c}^* \\ \boldsymbol{c}^*_- \end{bmatrix} = \operatorname{argmin} \left\| \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{c}_- \end{bmatrix} \right\|_1 \quad \text{s. t.} \quad \boldsymbol{y} = [\boldsymbol{Y}_i \;\; \boldsymbol{Y}_{-i}] \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{c}_- \end{bmatrix}, \tag{3.37}$$

recovers a subspace-sparse representation of every nonzero $\boldsymbol{y}$ in $\mathcal{S}_i$, i.e., $\boldsymbol{c}^* \neq \boldsymbol{0}$ and $\boldsymbol{c}^*_- = \boldsymbol{0}$, if and only if (3.36) holds.

Proof. ($\Longleftarrow$) We prove the result using contradiction. Assume $\boldsymbol{c}^*_- \neq \boldsymbol{0}$ and define

$$\mathbf{x} \triangleq \boldsymbol{y} - \boldsymbol{Y}_i \boldsymbol{c}^* = \boldsymbol{Y}_{-i} \boldsymbol{c}^*_-. \tag{3.38}$$

Since $\boldsymbol{y}$ lies in $\mathcal{S}_i$ and $\boldsymbol{Y}_i \boldsymbol{c}^*$ is a linear combination of points in $\mathcal{S}_i$, from the first equality in (3.38) we have that $\mathbf{x}$ is a vector in $\mathcal{S}_i$. Let $\boldsymbol{a}_i$ be the solution of (3.34) for $\mathbf{x}$. We have

$$\mathbf{x} = \boldsymbol{y} - \boldsymbol{Y}_i \boldsymbol{c}^* = \boldsymbol{Y}_i \boldsymbol{a}_i \implies \boldsymbol{y} = \boldsymbol{Y}_i (\boldsymbol{c}^* + \boldsymbol{a}_i). \tag{3.39}$$

On the other hand, since $\boldsymbol{Y}_{-i} \boldsymbol{c}^*_-$ is a linear combination of points in all subspaces except $\mathcal{S}_i$, from the second equality in (3.38) we have that $\mathbf{x}$ is a vector in $\oplus_{j \neq i} \mathcal{S}_j$. Let $\boldsymbol{a}_{-i}$ be the solution of (3.35) for $\mathbf{x}$. We have

$$\mathbf{x} = \boldsymbol{Y}_{-i} \boldsymbol{c}^*_- = \boldsymbol{Y}_{-i} \boldsymbol{a}_{-i} \implies \boldsymbol{y} = \boldsymbol{Y}_i \boldsymbol{c}^* + \boldsymbol{Y}_{-i} \boldsymbol{a}_{-i}. \tag{3.40}$$

Note that the left hand side of (3.40) together with the fact that $\boldsymbol{a}_{-i}$ is the optimal solution of (3.35) imply that

$$\|\boldsymbol{a}_{-i}\|_1 \leq \|\boldsymbol{c}^*_-\|_1. \tag{3.41}$$

From (3.39) and (3.40) we have that $\begin{bmatrix} \boldsymbol{c}^* + \boldsymbol{a}_i \\ \mathbf{0} \end{bmatrix}$ and $\begin{bmatrix} \boldsymbol{c}^* \\ \boldsymbol{a}_{-i} \end{bmatrix}$ are feasible solutions of the original optimization program in (3.37). Thus, we have

$$\left\| \begin{bmatrix} \boldsymbol{c}^* + \boldsymbol{a}_i \\ \mathbf{0} \end{bmatrix} \right\|_1 \leq \|\boldsymbol{c}^*\|_1 + \|\boldsymbol{a}_i\|_1 < \|\boldsymbol{c}^*\|_1 + \|\boldsymbol{a}_{-i}\|_1 \leq \left\| \begin{bmatrix} \boldsymbol{c}^* \\ \boldsymbol{c}^*_- \end{bmatrix} \right\|_1, \tag{3.42}$$

where the first inequality follows from the triangle inequality, the second strict inequality follows from the sufficient condition in (3.36), and the last inequality follows from (3.41). This contradicts the optimality of $\begin{bmatrix} \boldsymbol{c}^{*\top} & \boldsymbol{c}^{*\top}_- \end{bmatrix}^\top$ for the original optimization program in (3.37), hence proving the desired result.

($\Longleftarrow$) We prove the result using contradiction. Assume the condition in (3.36) does not hold, i.e., there exists a nonzero $\mathbf{x}$ in the intersection of $\mathcal{S}_i$ and $\oplus_{j \neq i} \mathcal{S}_j$ for which we have $\|\boldsymbol{a}_{-i}\|_1 \leq \|\boldsymbol{a}_i\|_1$. As a result, for $\boldsymbol{y} = \mathbf{x}$, a solution of the ℓ_1-minimization program (3.37) corresponds to selecting points from all subspaces except $\mathcal{S}_i$, which contradicts the subspace-sparse recovery assumption. ∎

While the necessary and sufficient condition in (3.36) guarantees a successful subspace-sparse recovery via the ℓ_1-minimization program, it does not explicitly show the relationship between the subspace arrangements and the data distribution for the success of the ℓ_1-minimization program. To establish such a relationship, we show that $\|\boldsymbol{a}_i\|_1 \leq \beta_i$, where β_i depends on the singular values of the data points in $\mathcal{S}_i$, and $\beta_{-i} \leq \|\boldsymbol{a}_{-i}\|_1$, where β_{-i} depends on the subspace angles between $\mathcal{S}_i$ and other subspaces. Then, the sufficient condition $\beta_i < \beta_{-i}$ establishes the relationship be-

tween the subspace angles and the data distribution under which the ℓ_1-minimization recovers subspace sparse representations of the data points in $\mathcal{S}_i$, since it implies

$$\|\boldsymbol{a}_i\|_1 \leq \beta_i < \beta_{-i} \leq \|\boldsymbol{a}_{-i}\|_1, \tag{3.43}$$

i.e., the condition of Theorem 2 holds.

Remark 5 *For independent subspaces, the intersection of a subspace with the direct sum of other subspaces is the origin, hence, the condition in* (3.36) *always holds. As a result, from Theorem 2, the ℓ_1-minimization always recovers subspace-sparse representations of data points in independent subspaces.*

Theorem 3 Consider a collection of data points drawn from n disjoint subspaces $\{\mathcal{S}_i\}_{i=1}^n$ of dimensions $\{d_i\}_{i=1}^n$. Let $\mathbb{W}_i$ be the set of all full-rank submatrices $\tilde{\boldsymbol{Y}}_i \in \mathbb{R}^{D\times d_i}$ of $\boldsymbol{Y}_i$, where $\text{rank}(\boldsymbol{Y}_i) = d_i$. If the condition

$$\max_{\tilde{\boldsymbol{Y}}_i \in \mathbb{W}_i} \sigma_{d_i}(\tilde{\boldsymbol{Y}}_i) > \sqrt{d_i}\, \|\boldsymbol{Y}_{-i}\|_{\infty,2} \max_{j\neq i} \cos(\theta_{ij}) \tag{3.44}$$

holds, then for every nonzero $\boldsymbol{y}$ in $\mathcal{S}_i$, the ℓ_1-minimization in (3.37) recovers a subspace-sparse solution, i.e., $\boldsymbol{c}^* \neq \boldsymbol{0}$ and $\boldsymbol{c}^*_- = \boldsymbol{0}$.[4]

Proof. We prove the result in two steps. In step 1, we show that $\|\boldsymbol{a}_i\|_1 \leq \beta_i$. In step 2, we show that $\beta_{-i} \leq \|\boldsymbol{a}_{-i}\|_1$. Then, the sufficient condition $\beta_i < \beta_{-i}$ establishes the result of the theorem, since it implies (3.43), i.e., the condition of Theorem 2 holds.

[4]Recall that $\|\boldsymbol{Y}_{-i}\|_{\infty,2}$ denotes the maximum ℓ_2-norm of the columns of $\boldsymbol{Y}_{-i}$.

Step 1: upper bound on the ℓ_1-norm of (3.34). Let $\mathbb{W}_i$ be the set of all submatrices $\tilde{\boldsymbol{Y}}_i \in \mathbb{R}^{D\times d_i}$ of $\boldsymbol{Y}_i$ that are full column rank. We can write the vector $\mathbf{x} \in \mathcal{S}_i \cap (\oplus_{j\neq i}\mathcal{S}_j)$

$$\mathbf{x} = \tilde{\boldsymbol{Y}}_i \tilde{\boldsymbol{a}} \implies \tilde{\boldsymbol{a}} = (\tilde{\boldsymbol{Y}}_i^\top \tilde{\boldsymbol{Y}}_i)^{-1}\tilde{\boldsymbol{Y}}_i^\top \mathbf{x}. \tag{3.45}$$

Using vector and matrix norm properties, we have

$$\begin{aligned}\|\tilde{\boldsymbol{a}}\|_1 \leq \sqrt{d_i}\|\tilde{\boldsymbol{a}}\|_2 &= \sqrt{d_i}\,\|(\tilde{\boldsymbol{Y}}_i^\top \tilde{\boldsymbol{Y}}_i)^{-1}\tilde{\boldsymbol{Y}}_i^\top \mathbf{x}\|_2 \\ &\leq \sqrt{d_i}\,\|(\tilde{\boldsymbol{Y}}_i^\top \tilde{\boldsymbol{Y}}_i)^{-1}\tilde{\boldsymbol{Y}}_i^\top\|_{2,2}\|\mathbf{x}\|_2 = \frac{\sqrt{d_i}}{\sigma_{d_i}(\tilde{\boldsymbol{Y}}_i)}\,\|\mathbf{x}\|_2, \end{aligned} \tag{3.46}$$

where $\sigma_{d_i}(\tilde{\boldsymbol{Y}}_i)$ denotes the d_i-th largest singular value of $\tilde{\boldsymbol{Y}}_i$. Thus, for the solution of the optimization problem in (3.34), we have

$$\|\boldsymbol{a}_i\|_1 \leq \min_{\tilde{\boldsymbol{Y}}_i\in\mathbb{W}_i}\|\tilde{\boldsymbol{a}}\|_1 \leq \min_{\tilde{\boldsymbol{Y}}_i\in\mathbb{W}_i}\frac{\sqrt{d_i}}{\sigma_{d_i}(\tilde{\boldsymbol{Y}}_i)}\|\mathbf{x}\|_2 \triangleq \beta_i, \tag{3.47}$$

which established the upper bound on the ℓ_1-norm of the solution of the optimization program in (3.34).

Step 2: lower bound on the ℓ_1-norm of (3.35) For the solution of (3.35) we have $\mathbf{x} = \boldsymbol{Y}_{-i}\boldsymbol{a}_{-i}$. If we multiply both sides of this equation from left by $\mathbf{x}^\top$, we get

$$\|\mathbf{x}\|_2^2 = \mathbf{x}^\top\mathbf{x} = \mathbf{x}^\top\boldsymbol{Y}_{-i}\boldsymbol{a}_{-i}. \tag{3.48}$$

Applying the Holder's inequality ($|\boldsymbol{u}^\top\boldsymbol{v}| \leq \|\boldsymbol{u}\|_\infty\|\boldsymbol{v}\|_1$) to the above equation, we obtain

$$\|\mathbf{x}\|_2^2 \leq \|\boldsymbol{Y}_{-i}^\top\mathbf{x}\|_\infty\|\boldsymbol{a}_{-i}\|_1. \tag{3.49}$$

By recalling the definition of the smallest principal angle between two subspaces, we can write

$$\|\mathbf{x}\|_2^2 \leq \max_{j \neq i} \cos(\theta_{ij}) \, \|\boldsymbol{Y}_{-i}\|_{\infty,2} \, \|\mathbf{x}\|_2 \, \|\boldsymbol{a}_{-i}\|_1, \tag{3.50}$$

where θ_{ij} is the first principal angle between $\mathcal{S}_i$ and $\mathcal{S}_j$. We can rewrite (3.50) as

$$\beta_{-i} \triangleq \frac{\|\mathbf{x}\|_2}{\max_{j \neq i} \cos(\theta_{ij}) \, \|\boldsymbol{Y}_{-i}\|_{\infty,2}} \leq \|\boldsymbol{a}_{-i}\|_1 \tag{3.51}$$

which establishes the lower bound on the ℓ_1 norm of the solution of the optimization program in (3.35). ∎

Loosely speaking, the sufficient condition in Theorem 3 states that if the smallest principal angle between each $\mathcal{S}_i$ and any other subspace is above a certain value that depends on the data distribution in $\mathcal{S}_i$, then the subspace-sparse recovery holds. Notice that this bound can be rather high when the norms of data points are oddly distributed, e.g., when the maximum norm of data points in $\mathcal{S}_i$ is much smaller than the maximum norm of data points in all other subspaces. Since the segmentation of data does not change when data points are scaled, we can apply SSC to linear subspaces after normalizing data points to have unit Euclidean norms. In this case, the sufficient condition in (3.44) reduces to

$$\max_{\tilde{\boldsymbol{Y}}_i \in \mathbb{W}_i} \sigma_{d_i}(\tilde{\boldsymbol{Y}}_i) > \sqrt{d_i} \, \max_{j \neq i} \cos(\theta_{ij}). \tag{3.52}$$

Remark 6 *The condition in* (3.36) *is closely related to the nullspace property in the sparse recovery literature [27, 70, 96, 108]. The key difference, however, is that we*

only require the inequality in (3.36) *to hold for the optimal solutions of* (3.34) *and* (3.35) *instead of any feasible solution. Thus, while the inequality can be violated for many feasible solutions, it can still hold for the optimal solutions, guaranteeing successful subspace-sparse recovery from Theorem 2. Thus, our result can be thought of as a generalization of the nullspace property to the multi-subspace setting where the number of points in each subspace is arbitrary.*

3.5.3 Geometric interpretation

In this section, we provide a geometric interpretation of the subspace-sparse recovery conditions in (3.36) and (3.44). To do so, it is necessary to recall the relationship between the ℓ_1-norm of the optimal solution of

$$\min \|\boldsymbol{a}\|_1 \quad \text{s. t.} \quad \mathbf{x} = \boldsymbol{B}\boldsymbol{a}, \tag{3.53}$$

and the symmetrized convex polytope of the columns of $\boldsymbol{B}$ [32]. More precisely, if we denote the columns of $\boldsymbol{B}$ by $\boldsymbol{b}_i$ and define the symmetrized convex hull of the columns of $\boldsymbol{B}$ by

$$\mathcal{P} \triangleq \text{conv}(\pm\boldsymbol{b}_1, \pm\boldsymbol{b}_2, \cdots), \tag{3.54}$$

then the ℓ_1-norm of the optimal solution of (3.53) corresponds to the smallest $\alpha > 0$ such that the scaled polytope $\alpha\mathcal{P}$ reaches $\mathbf{x}$ [32]. Let us denote the symmetrized convex polytopes of $\boldsymbol{Y}_i$ and $\boldsymbol{Y}_{-i}$ by $\mathcal{P}_i$ and $\mathcal{P}_{-i}$, respectively. Then the condition in (3.36) has the following geometric interpretation: *subspace-sparse recovery in $\mathcal{S}_i$ holds*

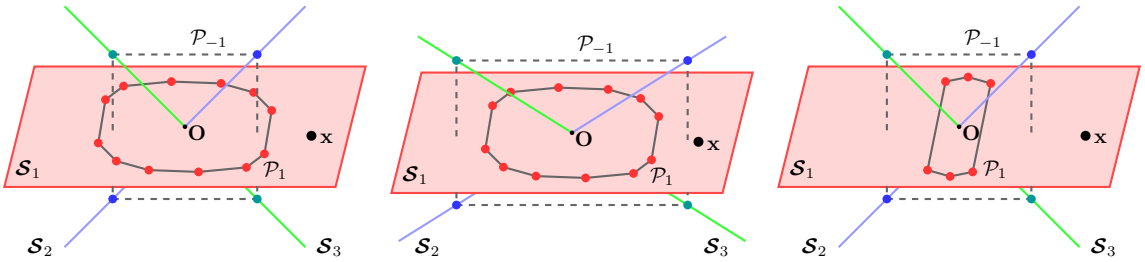


Figure 3.5: Left: for any nonzero $\mathbf{x}$ in the intersection of $\mathcal{S}_1$ and $\mathcal{S}_2 \oplus \mathcal{S}_3$, the polytope $\alpha\mathcal{P}_1$ reaches $\mathbf{x}$ for a smaller α than $\alpha\mathcal{P}_{-1}$, hence, subspace-sparse recovery holds. Middle: when the subspace angle decreases, the polytope $\alpha\mathcal{P}_{-1}$ reaches $\mathbf{x}$ for a smaller α than $\alpha\mathcal{P}_1$. Right: when the distribution of the data in $\mathcal{S}_1$ becomes nearly degenerate, in this case close to a line, the polytope $\alpha\mathcal{P}_{-1}$ reaches $\mathbf{x}$ for a smaller α than $\alpha\mathcal{P}_1$. In both cases, in the middle and right, the subspace-sparse recovery does not hold for points at the intersecion.

if and only if for any nonzero $\mathbf{x}$ *in the intersection of* $\mathcal{S}_i$ *and* $\oplus_{j\neq i}\mathcal{S}_j$, $\alpha\mathcal{P}_i$ *reaches* $\mathbf{x}$ *for a smaller* α *than* $\alpha\mathcal{P}_{-i}$.

As shown in the left plot of Figure 3.5, for $\mathbf{x}$ in the intersection of $\mathcal{S}_1$ and $\mathcal{S}_2 \oplus \mathcal{S}_3$, the polytope $\alpha\mathcal{P}_1$ reaches $\mathbf{x}$ for a smaller α than $\alpha\mathcal{P}_{-1}$, hence the subspace-sparse recovery condition holds. On the other hand, when the principal angles between $\mathcal{S}_1$ and other subspaces decrease, as shown in the middle plot of Figure 3.5, the subspace-sparse recovery condition does not hold since the polytope $\alpha\mathcal{P}_{-1}$ reaches $\mathbf{x}$ sooner than $\alpha\mathcal{P}_1$ does. Also, as shown in the right plot of Figure 3.5, when the distribution of the data in $\mathcal{S}_1$ becomes nearly degenerate, in this case close to a 1-dimensional subspace orthogonal to the direction of $\mathbf{x}$, then the subspace-sparse recovery condition does not hold since $\alpha\mathcal{P}_{-1}$ reaches $\mathbf{x}$ sooner than $\alpha\mathcal{P}_1$. Note that the sufficient condition in (3.44) translates the relationship between the polytopes, mentioned above, explicitly in terms of relationship between the subspace angles and the singular values of the

data.

3.6 Graph connectivity

In the previous section, we studied conditions under which the proposed ℓ_1-minimization program recovers subspace-sparse representations of data points in a union of subspaces. As a result, in the similarity graph, the points that lie in different subspaces do not get connected to each other. On the other hand, our extensive experimental results on synthetic and real data show that data points in the same subspace always form a connected component of the graph, hence, for n subspaces the similarity graph has n connected components. [83] has theoretically verified the connectivity of points in the same subspace for 2 and 3 dimensional subspaces. However, it has shown that, for subspaces of dimensions greater than or equal to 4, under odd distribution of the data, it is possible that points in the same subspace form multiple components of the graph.

In this section, we consider a regularization term in the sparse optimization program that promotes connectivity of points in each subspace.[5] We use the idea that if data points in each subspace choose a few *common* points from the same subspace in their sparse representations, then they form a single component of the similarity

[5]Another approach to deal with the connectivity issue is to analyze the subspaces corresponding to the components of the graph and merge the components whose associated subspaces have a small distance from each other, i.e., have a small principal angle. However, the result can be sensitive to the choice of the dimension of the subspaces to fit to each component as well as the threshold value on the principal angles to merge the subspaces.

graph. Thus, we add to the sparse optimization program the regularization term

$$\|\boldsymbol{C}\|_{r,0} \triangleq \sum_{i=1}^{N} \mathrm{I}(\|\boldsymbol{c}^i\|_2 > 0), \tag{3.55}$$

where $\mathrm{I}(\cdot)$ denotes the indicator function and $\boldsymbol{c}^i$ denotes the i-th row of $\boldsymbol{C}$. Hence, minimizing (3.55) corresponds to minimizing the number of nonzero rows of $\boldsymbol{C}$ [39, 64, 105], i.e., choosing a few common data points in the sparse representation of each point (see [38, 39] for closely related applications of row-sparsity for the problem of finding exemplars in datasets). Since a minimization problem that involves (3.55) is in general NP-hard, we consider its convex relaxation as

$$\|\boldsymbol{C}\|_{r,1} \triangleq \sum_{i=1}^{N} \|\boldsymbol{c}^i\|_2. \tag{3.56}$$

Thus, to increase the connectivity of data points from the same subspace in the similarity graph, we propose to solve

$$\min \|\boldsymbol{C}\|_1 + \lambda_r \|\boldsymbol{C}\|_{r,1} \quad \text{s. t.} \quad \boldsymbol{Y} = \boldsymbol{Y}\boldsymbol{C}, \ \ \mathrm{diag}(\boldsymbol{C}) = \boldsymbol{0}, \tag{3.57}$$

where $\lambda_r > 0$ sets the trade-off between the sparsity of the solution and the connectivity of the graph. Figure 3.6 shows how adding this regularization term promotes selecting common points in sparse representations. The following example gives a reason for using the row-sparsity term as a regularizer but not as an objective function instead of the ℓ_1-norm.

Example 1 *Consider the three* 1*-dimensional subspaces in* $\mathbb{R}^2$*, shown in Figure 3.7, where data points have unit Euclidean norms and the angle between* $\mathcal{S}_1$ *and* $\mathcal{S}_2$ *as*

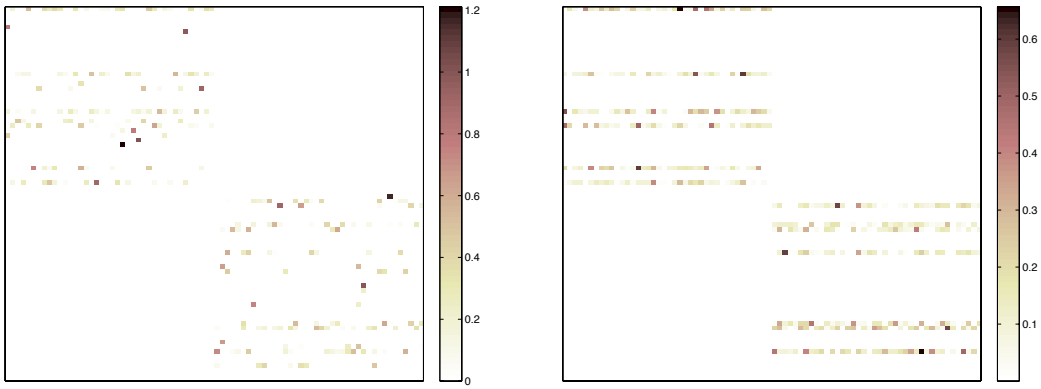
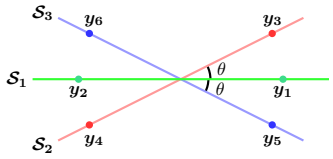


Figure 3.6: Coefficient matrix obtained from the solution of (3.57) for data points in two subspaces. Left: $\lambda_r = 0$. Right: $\lambda_r = 10$. Increasing λ_r results in concentration of the nonzero elements in a few rows of the coefficient matrix, hence choosing a few common data points.

well as $\mathcal{S}_1$ and $\mathcal{S}_3$ is equal to θ. Note that in this example, the sufficient condition in (3.44) *holds for all values of $\theta \in (0, \frac{\pi}{2})$. As a result, the solution of* (3.57) *with $\lambda_r = 0$ recovers a subspace-sparse representation for each data point, which in this example is uniquely given by $\boldsymbol{C}_1$ shown in Figure 3.7. Hence, the similarity graph has exactly* 3 *connected components corresponding to the data points in each subspace. Another feasible solution of* (3.57) *is given by $\boldsymbol{C}_2$, shown in Figure 3.7, where the points in $\mathcal{S}_1$ choose points from $\mathcal{S}_2$ and $\mathcal{S}_3$ in their representations. Hence, the similarity graph has only one connected component. Note that for a large range of subspace angles $\theta \in (0, \frac{4\pi}{10})$ we have*

$$\|\boldsymbol{C}_2\|_{r,1} = \sqrt{16 + 2/\cos^2(\theta)} < \|\boldsymbol{C}_1\|_{r,1} = 6. \tag{3.58}$$

As a result, for large values of λ_r, i.e., when only the second term of the objective function in (3.57) *is minimized, we cannot recover subspace-sparse representations of data points. This suggests using the row-sparsity regularizer with a small value of λ_r.*



$$\boldsymbol{C}_1 = \left(\begin{array}{cc|cc|cc} 0 & -1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 \end{array}\right) \quad \boldsymbol{C}_2 = \left(\begin{array}{cc|cc|cc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline +\frac{1}{4\cos\theta} & -\frac{1}{4\cos\theta} & 0 & -1 & 0 & 0 \\ -\frac{1}{4\cos\theta} & +\frac{1}{4\cos\theta} & -1 & 0 & 0 & 0 \\ \hline +\frac{1}{4\cos\theta} & -\frac{1}{4\cos\theta} & 0 & 0 & 0 & -1 \\ -\frac{1}{4\cos\theta} & +\frac{1}{4\cos\theta} & 0 & 0 & -1 & 0 \end{array}\right)$$

Figure 3.7: Left: three 1-dimensional subspaces in $\mathbb{R}^2$ with normalized data points. Middle: $\boldsymbol{C}_1$ corresponds to the solution of (3.57) for $\lambda_r = 0$. The similarity graph of $\boldsymbol{C}_1$ has three components corresponding to the three subspaces. Right: $\boldsymbol{C}_2$ corresponds to the solution of (3.57) for $\lambda_r \to +\infty$ and $\theta \in (0, \frac{4\pi}{10})$. The similarity graph of $\boldsymbol{C}_2$ has only one connected component.

3.7 Experiments with synthetic data

In this section, we consider synthetic data and verify the effect of the subspace angles and the data distribution on the success of SSC. Moreover, we evaluate the performance of SSC for dealing with different numbers and classes of subspaces with different dimensions.

3.7.1 Subspace angle and data distribution effect

In Section 3.5, we showed that the success of the ℓ_1-minimization for subspace-sparse recovery depends on the principal angles between subspaces and the distribution of data in each subspace. In this section, we verify this relationship through experiments on synthetic data.

We consider three disjoint subspaces $\{\mathcal{S}_i\}_{i=1}^3$ of the same dimension d embedded in the D-dimensional ambient space. To make the problem hard enough so that every data point in a subspace can also be reconstructed as a linear combination of points in

other subspaces, we generate subspace bases $\{\boldsymbol{U}_i \in \mathbb{R}^{D\times d}\}_{i=1}^{3}$ such that each subspace lies in the direct sum of the other two subspaces, i.e., $\text{rank}(\begin{bmatrix}\boldsymbol{U}_1 & \boldsymbol{U}_2 & \boldsymbol{U}_3\end{bmatrix}) = 2d$. In addition, we generate the subspaces such that the smallest principal angles θ_{12} and θ_{23} are equal to θ. Thus, we can verify the effect of the smallest principal angle in the subspace-sparse recovery by changing the value of θ.

To investigate the effect of the data distribution in subspace-sparse recovery, we generate the same number of data points, N_g, in each subspace at random and change the value of N_g. Typically, as the number of data points in a subspace increases, the probability of the data being close to a degenerate subspace decreases.[6]

After generating three d-dimensional subspaces associated to (θ, N_g), we solve the ℓ_1-minimization program in (3.4) for each data point and measure two different errors. First, denoting the sparse representation of $\boldsymbol{y}_i \in \mathcal{S}_{k_i}$ by $\boldsymbol{c}_i^\top \triangleq \begin{bmatrix}\boldsymbol{c}_{i1}^\top & \boldsymbol{c}_{i2}^\top & \boldsymbol{c}_{i3}^\top\end{bmatrix}$, with $\boldsymbol{c}_{ij}$ corresponding to points in $\mathcal{S}_j$, we measure the *subspace-sparse recovery error* by

$$\text{ssr error} = \frac{1}{3N_g}\sum_{i=1}^{3N_g}(1 - \frac{\|\boldsymbol{c}_{ik_i}\|_1}{\|\boldsymbol{c}_i\|_1}) \in [0,1], \tag{3.59}$$

where each term inside the summation indicates the fraction of the ℓ_1-norm of $\boldsymbol{c}_i$ that comes from points in other subspaces. The error being zero corresponds to $\boldsymbol{y}_i$ choosing points only in its own subspace, while the error being equal to one corresponds to $\boldsymbol{y}_i$ choosing points from other subspaces. Second, after building the similarity graph using the sparse coefficients and applying spectral clustering, we measure the *subspace*

[6]To remove the effect of different scalings of data points, i.e., to consider only the effect of the principal angle and number of points, we normalize the data points.

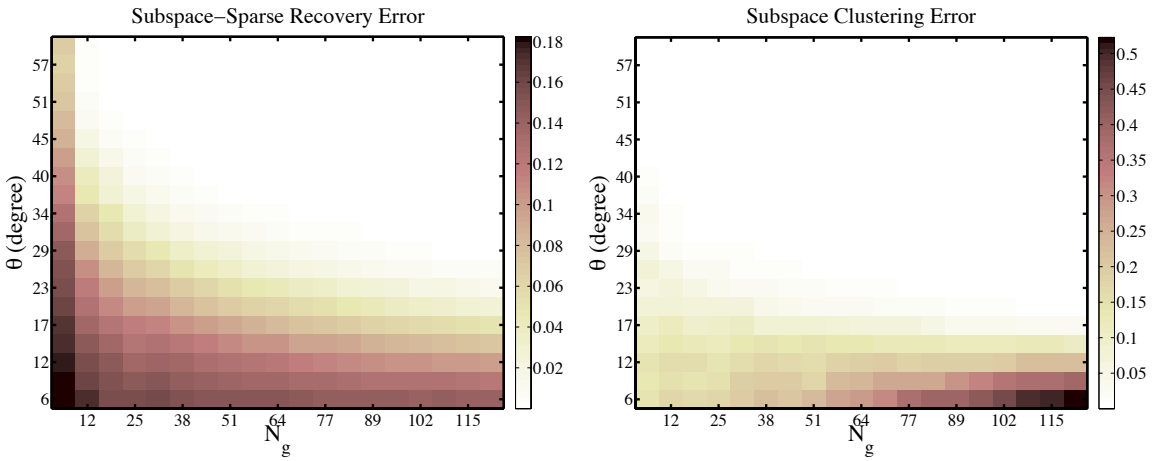


Figure 3.8: Subspace-sparse recovery error (left) and subspace clustering error (right) for three disjoint subspaces. Increasing the number of points or smallest principal angle decreases the errors.

clustering error by

$$\text{subspace clustering error} = \frac{\#\text{ of misclassified points}}{\text{total }\#\text{ of points}}. \tag{3.60}$$

In our experiments, we set the dimension of the ambient space to $D = 50$. We change the smallest principal angle between subspaces as $\theta \in [6, 60]$ degrees and change the number of points in each subspace as $N_g \in [d{+}1, 32d]$. For each pair (θ, N_g) we compute the average of the errors in (3.59) and (3.60) over 100 trials (randomly generated subspaces and data points). The results for $d = 4$ are shown in Figure 3.8. Note that when either θ or N_g is small, both the subspace-sparse recovery error and the clustering error are large, as predicted by our theoretical analysis. On the other hand, when θ or N_g increases, the errors decrease, and for (θ, N_g) sufficiently large we obtain zero errors. The results also verify that the success of the clustering relies on the success of the ℓ_1-minimization in recovering subspace-sparse representations

of data points. Note that for small θ as we increase N_g, the subspace-sparse recovery error is large and slightly decreases, while the clustering error increases. This is due to the fact that increasing the number of points, the number of undesirable edges between different subspaces in the similarity graph increases, making the spectral clustering more difficult. Note also that, for the values of (θ, N_g) where the subspace-sparse recovery error is zero, i.e., points in different subspaces are not connected to each other in the similarity graph, the clustering error is also zero. This implies that, in such cases, the similarity graph has exactly three connected components, i.e., data points in the same subspace form a single component of the graph.

3.7.2 Effect of different numbers, dimensions, and models of subspaces

In this section, we investigate the effect of the two classes of independent and disjoint subspace models as well as the effect of the number and the dimensions of subspaces on different subspace clustering methods. We compare SSC with the best state-of-the-art subspace clustering algorithms: LSA [118], SCC [22], LRR [76], and LRSC [46].

For the state-of-the-art algorithms, we use the codes provided by their authors. Note that the LRR algorithm according to [76], similar to SSC, applies spectral clustering to a similarity graph built directly from the solution of its proposed optimiza-

tion program. However, the code of the algorithm applies a heuristic post-processing step, similar to [71], to the low-rank solution prior to building the similarity graph, as also discussed in [75]. Thus, to compare the effectiveness of sparse versus low-rank objective function and to investigate the effect of the post-processing step of LRR, we report the results for both cases of without (LRR) and with (LRR-H) the post-processing step.

As LSA and SCC need to know the number of subspaces a priori and also estimating the number of subspaces from the eigenspectrum of the graph Laplacian in the noisy setting is often unreliable, in order to have a fair comparison, we provide the number of subspaces as an input to all the algorithms.

We generate random bases for independent and disjoint subspaces and randomly generate data points in each subspaces. We consider subspaces with the same dimensions and with different dimensions and also change the number of subspaces. We denote by $d = (d_1, \cdots, d_n)$ the collection of n subspaces, where subspace i has dimension d_i. We consider noise-free data as well as noisy data, where we add to each noise-free data point, $\boldsymbol{y}_i^0$, in a subspace, a noise vector, $\boldsymbol{z}_i$, orthogonal to the direction of the subspace such that $\|\boldsymbol{z}_i\|_2 \leq \sigma \|\boldsymbol{y}_i^0\|_2$ for a fixed $\sigma > 0$. For each fixed $(d_1, \cdots, d_n)$, a fixed subspace and a noise model (noise-free or noisy model), we generate n random subspaces in $\mathbb{R}^D$ and $10\, d_i$ random data points in each subspace $\mathcal{S}_i$. We apply different subspace clustering methods and compute the average and median clustering errors for each algorithm over 100 random trials.

Table 3.1: Clustering error (%) of different algorithms on synthetic noise-free data for different dimensions and number of subspaces as well as different subspace models.

Algorithms	LSA	SCC	LRR	LRR-H	LRSC	SSC
independent subspaces						
$d = (3, 3, 3)$						
Mean	2.23	0.00	0.00	0.00	0.00	0.00
Median	2.22	0.00	0.00	0.00	0.00	0.00
$d = (2, 3, 5)$						
Mean	2.17	0.81	0.10	0.00	0.00	0.00
Median	2.00	1.00	0.00	0.00	0.00	0.00
$d = (4, 4, 4, 4, 4)$						
Mean	0.63	0.00	0.00	0.00	0.00	0.00
Median	0.50	0.00	0.00	0.00	0.00	0.00
$d = (1, 2, 3, 4, 5)$						
Mean	5.57	3.78	0.83	0.00	0.17	0.00
Median	2.00	2.00	0.67	0.00	0.00	0.00
disjoint subspaces						
$d = (3, 3, 3)$						
Mean	9.77	0.00	15.09	7.38	11.41	0.97
Median	8.89	0.00	13.33	5.56	8.89	0.00
$d = (2, 3, 5)$						
Mean	5.98	0.88	7.13	1.21	4.64	0.11
Median	6.00	1.00	4.50	0.00	4.00	0.00
$d = (4, 4, 4, 4, 4)$						
Mean	18.98	0.00	42.73	32.03	39.46	2.46
Median	18.50	0.00	43.00	32.50	40.50	2.00
$d = (1, 2, 3, 4, 5)$						
Mean	5.23	6.97	27.15	5.84	23.49	0.95
Median	4.67	4.33	28.33	4.67	24.33	0.00

Tables 3.1 and 3.2 show the clustering errors for the noise-free and noisy data points, respectively, for $\sigma = 0.1$ and $D = 30$. From the results, we make the following conclusions:

– The performance of LRR and LRSC depend on the subspace model. For independent subspaces, they obtain very small clustering errors, which is expected since

Table 3.2: Clustering error (%) of different algorithms on synthetic noisy data for different dimensions and number of subspaces as well as different subspace models.

Algorithms	LSA	SCC	LRR	LRR-H	LRSC	SSC
independent subspaces						
$d = (3, 3, 3)$						
Mean	2.48	0.00	0.00	0.04	0.00	0.00
Median	2.22	0.00	0.00	0.00	0.00	0.00
$d = (2, 3, 5)$						
Mean	1.76	0.21	0.36	0.00	0.03	0.02
Median	1.00	0.00	0.00	0.00	0.00	0.00
$d = (4, 4, 4, 4, 4)$						
Mean	0.70	0.00	0.00	0.00	0.00	0.00
Median	0.50	0.00	0.00	0.00	0.00	0.00
$d = (1, 2, 3, 4, 5)$						
Mean	0.85	27.86	1.45	0.02	0.67	0.02
Median	0.67	29.00	0.67	0.00	0.00	0.00
disjoint subspaces						
$d = (3, 3, 3)$						
Mean	9.92	0.00	13.79	7.76	9.70	0.81
Median	8.89	0.00	12.22	5.56	7.78	0.00
$d = (2, 3, 5)$						
Mean	6.69	0.28	6.15	1.34	3.92	0.12
Median	5.00	0.00	4.00	0.00	2.00	0.00
$d = (3, 3, 3, 3, 3)$						
Mean	19.25	0.00	42.50	31.68	38.15	5.71
Median	19.50	0.00	43.25	31.50	38.50	3.00
$d = (1, 2, 3, 4, 5)$						
Mean	5.51	30.85	24.35	5.57	21.93	3.19
Median	5.31	33.67	24.67	4.00	22.00	0.67

these algorithms have theoretical guarantees for independent subspaces. On the other hand, for disjoint subspaces, LRR and LRSC, which work under the low-rank coefficient matrix criterion, obtain large clustering errors, suggesting that they cannot work well beyond the independent subspace model.

– For independent subspaces, SSC obtains very small clustering errors, which is ex-

pected since SSC always works under the independent subspace model as we showed in the theoretical analysis. For disjoint subspaces, the clustering errors of SSC slightly increase but still are small. This is expected, since for disjoint subspaces, SSC works under some conditions on the subspace angles and the data distribution, which can be violated for some data points or subspaces.

– The clustering performance of SCC does not depend on the subspace models, i.e., being independent or disjoint. However, it depends on whether subspaces have the same or different dimensions. More specifically, SCC obtains very low clustering errors when the subspaces have the same or very close dimensions for both independent and disjoint subspaces. On the other hand, the clustering error of SCC is large when the subspaces have very different dimensions. This comes from the fact that SCC uses the maximum dimension, $d_{\max}$, of the subspaces to compute the affinity among $d_{\max} + 2$ data points. As a result, it is possible that points in different subspaces of small dimensions obtain a large affinity, i.e., be considered by the algorithm to be from the same subspace.

– Unlike other algorithms that obtain nearly zero clustering errors for independent subspaces, LSA obtains larger clustering errors for both independent and disjoint subspaces. This comes from the fact that LSA computes the affinity between pairs of points by first fitting a local subspace to each data point and its nearest neighbors. Since the neighborhood of a data point may contain points from different subspaces, the locally fitted subspace may not be close to the true underlying subspace at the

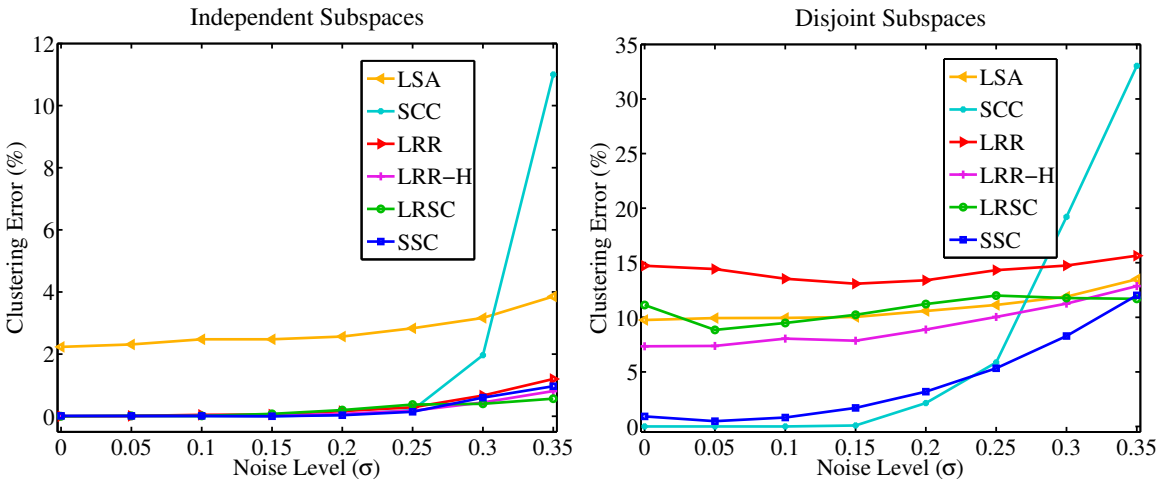


Figure 3.9: Average clustering errors of different subspace clustering algorithms as a function of the noise level, σ, for independent subspace model (left) and disjoint subspace model (right).

point, hence degrading the performance of the algorithm. Moreover, for disjoint subspaces, the probability of having points on other subspaces that are close to the given point increases, hence, increasing the clustering error of LSA.

Figure 3.9 shows the average clustering errors of different algorithms as a function of the noise level, σ, for independent and disjoint subspace models for $d = (3, 3, 3)$. As the results show, all algorithms except LSA obtain very low clustering errors for small and moderate amount of noise for independent subspaces (the performance of SCC degrades as the for the noise level above $\sigma = 0.25$). On the other hand, for disjoint subspaces, only SSC and SCC obtain low clustering errors for small and moderate amount of noise, while other algorithms obtain large errors for all levels of noise.

3.7.3 Dealing with incomplete data

We investigate now the performances of different subspace clustering algorithms in the case of missing entries in the data. We generate $n = 5$ subspaces of dimensions $d_i = i$ in $\mathbb{R}^D$, for $i \in \{1, 2, \ldots, 5\}$, and generate $10\, d_i$ random data points in each subspace $\mathcal{S}_i$. We randomly select ρ percentage of the D entries of the data points as the missing entries. We then apply subspace clustering algorithms to the data matrix after removing the rows corresponding to the missing entries, as described in Section 3.3.2, and compute the average clustering errors for each algorithm over 100 random trials.

Figure 3.10 shows the clustering results of different algorithms for independent and disjoint subspaces, for $D = 30$. Notice that for both models, SSC obtains very low clustering errors when at most 70% of the entries are missing. For independent subspace model, all methods except SCC obtain low clustering errors when at most 50% of the entries are missing. On the other hand, for disjoint subspaces, the clustering errors of all algorithms except SSC increase, as we also saw in the previous section for the case of noisy data.

3.7.4 Dealing with sparse outlying entires

Finally, we evaluate the performance of the SSC algorithm for dealing with sparse outlying entries in the data. We investigate the effect of the subspace model, per-

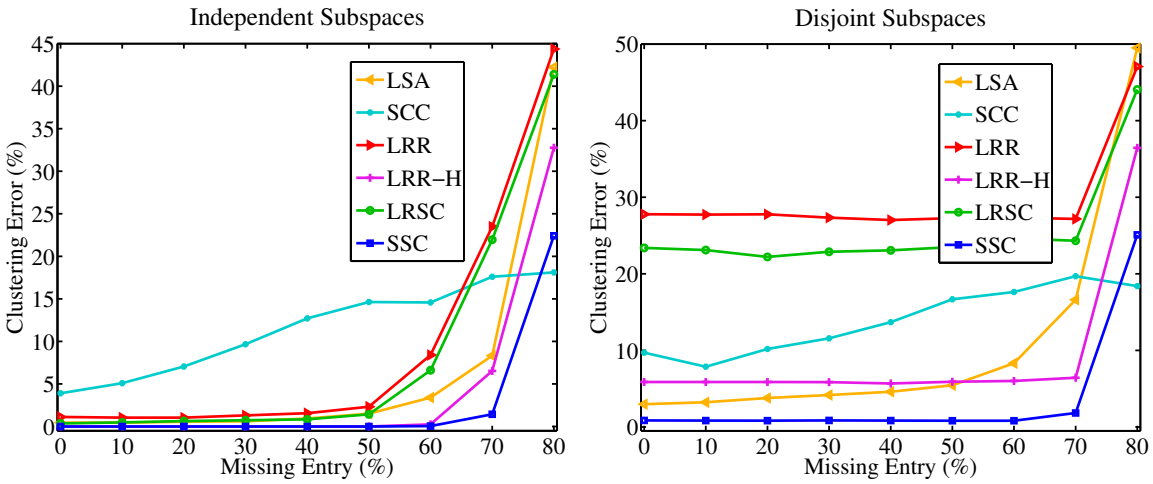


Figure 3.10: Average clustering errors of different subspace clustering algorithms as a function of the percentage of the missing entries in the data, for independent subspace model (left) and disjoint subspace model (right).

centage of the corrupted data points, percentage of the corrupted entries in the data points and the magnitude of the corruption terms. We generate $n = 3$ subspaces of dimensions $d_1 = 2$, $d_2 = 3$ and $d_3 = 5$ in $\mathbb{R}^{100}$ and randomly generate $10\,d_i$ data points in each subspace $\mathcal{S}_i$. We randomly select ρ_1 percentage of the $N = \sum_{i=1}^{3} 10 d_i$ data points as the candidate data points to be corrupted by sparse errors. For each candidate corrupted data point, we randomly select ρ_2 percentage of its entries and add a random Guassian error of variance σ to each selected entry. We change ρ_1 and ρ_2 in $\{10, 20, \ldots, 90\}$ and $\sigma \in \{0.1, 0.3\}$.

Figure 3.11 shows the average clustering errors of SSC for independent and disjoint subspace models over 100 random trials. Notice that, in all cases, SSC obtains very low clustering error when either the percentage of the corrupted data or the percentage of the corrupted entries are not large. When both the percentages of the corrupted

data and corrupted entries increase the clustering error increases, as expected. As the plots show, when the magnitude of the error, σ, increases, the clustering error increases. More specifically, the maximum clustering error increases from 2.5% to 17.7% for independent subspaces and from 4.6% to 23.3% for disjoint subspaces. Moreover, as expected, for a fixed σ, the clustering error for the case of disjoint subspace model is higher than the error for the case of independent subspace model.

3.8 Experiments with real data

In this section, we evaluate the performance of the SSC algorithm in dealing with two real-world problems: segmenting multiple motions in videos (Fig. 3.1) and clustering human face images (Fig. 3.2). We compare the performance of SSC with the best state-of-the-art subspace clustering algorithms: LSA [118], SCC [22], LRR [76], and LRSC [46].

3.8.1 Implementation details

We implement the SSC optimization algorithm in (3.13) using the Alternating Direction Method of Multipliers (ADMM) framework, described in Section 3.4. For the motion segmentation experiments, we use the noisy variation of the optimization program (3.13), i.e., without the term $\boldsymbol{E}$, with the affine constraint, and choose $\lambda_z = 800/\mu_z$ in all the experiments (μ_z is defined in (3.14)). For the face clustering

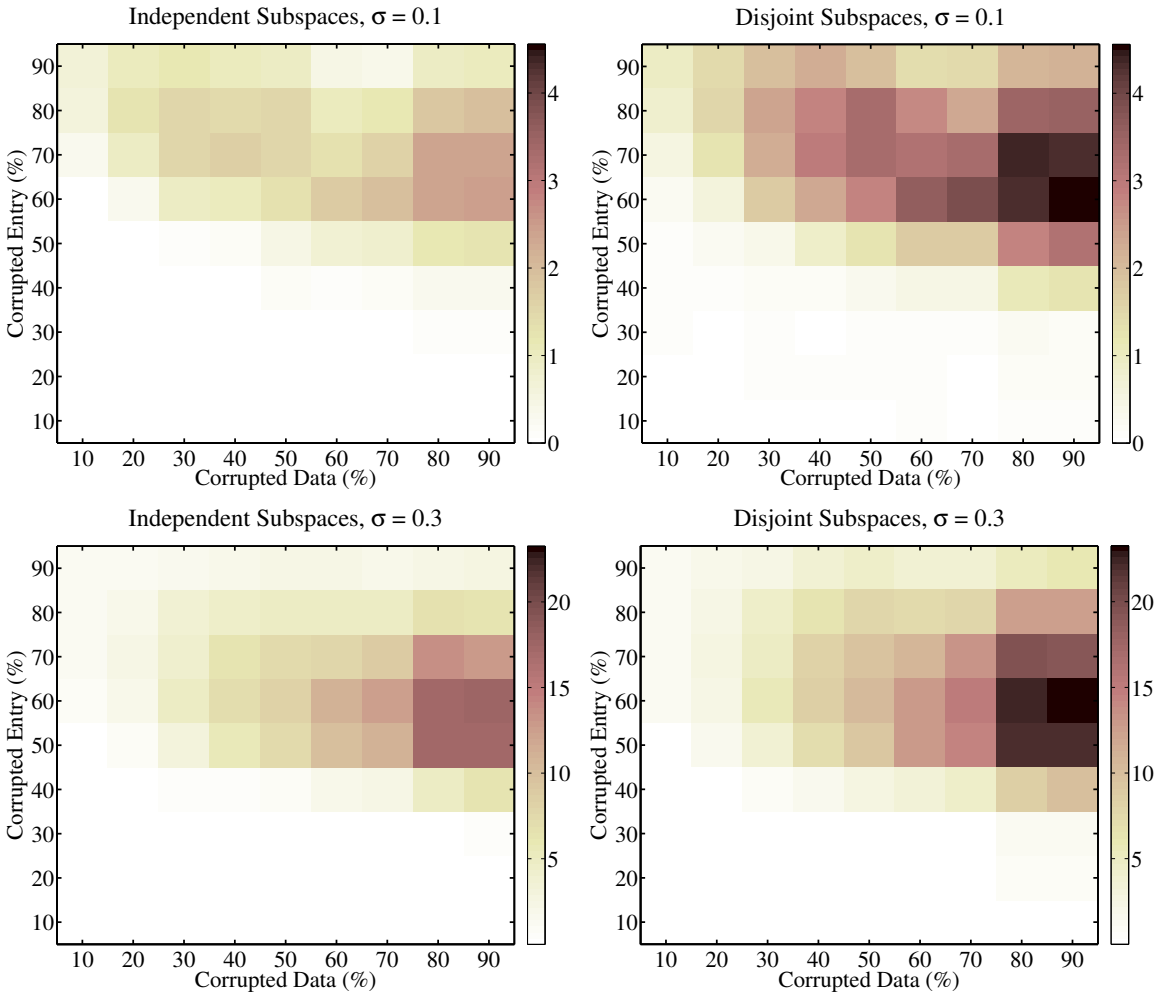


Figure 3.11: Average clustering errors of the SSC algorithm as a function of the percentage of the corrupted data points and the percentage of the corrupted entries for independent (left) and disjoint (right) subspace model for different magnitudes of corruption, $\sigma \in \{0.1, 0.3\}$.

experiments, we use the sparse outlying entries variation of the optimization program (3.13), i.e., without the term $\boldsymbol{Z}$, and choose $\lambda_e = 20/\mu_e$ in all the experiments (μ_e is defined in (3.14)). It is also worth mentioning that SSC performs better with the ADMM approach than general interior point solvers [68], which typically return many small nonzero coefficients, degrading the spectral clustering result.

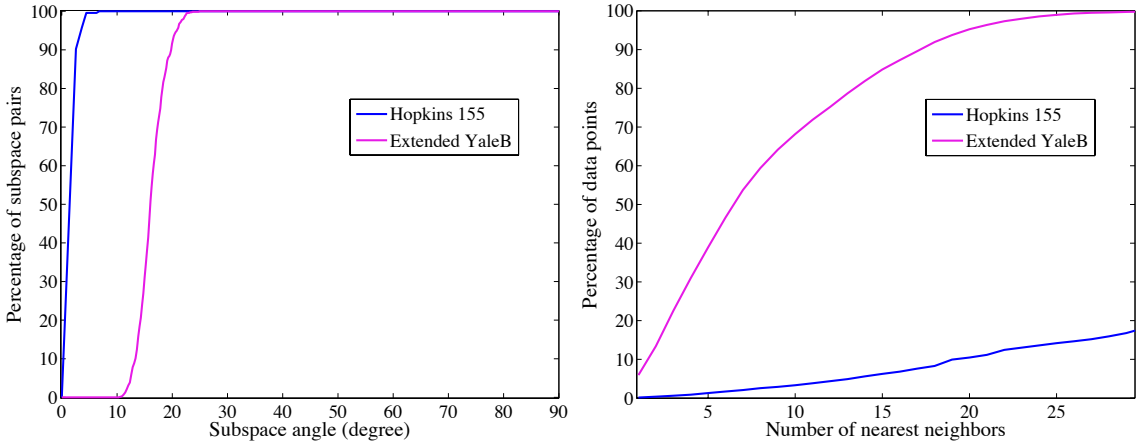


Figure 3.12: Left: percentage of pairs of subspaces whose smallest principal angle is smaller than a given value. Right: percentage of data points in pairs of subspaces whose K nearest neighbors contain points from the other subspace.

As LSA and SCC need to know the number of subspaces a priori and also estimating the number of subspaces from the eigenspectrum of the graph Laplacian in the noisy setting is often unreliable, in order to have a fair comparison, we provide the number of subspaces as an input to all the algorithms.

3.8.2 Datasets and some statistics

For the motion segmentation problem, we consider the Hopkins 155 dataset [103], which consists of 155 video sequences of 2 or 3 motions corresponding to 2 or 3 low-dimensional subspaces in each video [12, 102]. For the face clustering problem, we consider the Extended Yale B dataset [73], which consists of face images of 38 human subjects, where images of each subject lie in a low-dimensional subspace [6].

Before describing each problem in detail and presenting the experimental results,

we present some statistics on the two datasets that help to better understand the challenges of subspace clustering and the performance of different algorithms. First, we compute the smallest principal angle for each pair of subspaces, which in the motion segmentation problem corresponds to a pair of motions in a video and in the face clustering problem corresponds to a pair of subjects. Then, we compute the percentage of the subspace pairs whose smallest principal angle is below a certain value, which ranges from 0 to 90 degrees. Figure 3.12 (left) shows the corresponding graphs for the two datasets. As shown, subspaces in both datasets have relatively small principal angles. In the Hopkins-155 dataset, principal angles between subspaces are always smaller than 10 degrees, while in the Extended Yale B dataset, principal angles between subspaces are between 10 and 20 degrees. Second, we take each pair of subspaces in a dataset and compute the percentage of data points that among their K-nearest neighbors there are points from the other subspace. Figure 3.12 (right) shows the average percentages over all possible pairs of subspaces in each dataset. As shown, in the Hopkins-155 dataset for almost all data points, their few nearest neighbors belong to the same subspace. On the other hand, for the Extended Yale B dataset, there is a relatively large number of data points whose nearest neighbors come from the other subspace. This percentage rapidly increases as the number of nearest neighbors increases. As a result, from the two plots in Figure 3.12, we can conclude that in the Hopkins 155 dataset the challenge is that subspaces have small principal angles, while in the Extended Yale B dataset, beside the principal

angles between subspaces being small, the challenge is that data points in a subspace are very close to other subspaces.

3.8.3 Motion segmentation

Given a video sequence of multiple rigidly moving objects, the motion segmentation problem refers to the problem of segmenting the video into multiple spatiotemporal regions that correspond to different motions in the scene (Fig. 3.1). This problem is often solved by extracting and tracking a set of N feature points $\{\mathbf{x}_{fi} \in \mathbb{R}^2\}_{i=1}^{N}$ through the frames $f = 1, \ldots, F$ of the video. Each data point $\boldsymbol{y}_i$, which is also called a feature trajectory, corresponds to a $2F$-dimensional vector that consists of stacking the feature points $\mathbf{x}_{fi}$ in the video as

$$\boldsymbol{y}_i \triangleq \begin{bmatrix} \mathbf{x}_{1i}^\top & \mathbf{x}_{2i}^\top & \cdots & \mathbf{x}_{Fi}^\top \end{bmatrix}^\top \in \mathbb{R}^{2F}. \tag{3.61}$$

Motion segmentation refers to the problem of separating these feature trajectories according to their underlying motions. Under the affine projection model, all feature trajectories associated with a single rigid motion lie in an affine subspace of $\mathbb{R}^{2F}$ of dimension at most 3, or equivalently lie in a linear subspace of $\mathbb{R}^{2F}$ of dimension at most 4 [12, 102]. Therefore, feature trajectories of n rigid motions lie in a union of n low-dimensional subspaces of $\mathbb{R}^{2F}$. Hence, motion segmentation reduces to clustering of data points in a union of subspaces.

In this section, we evaluate the performance of the SSC algorithm as well as that

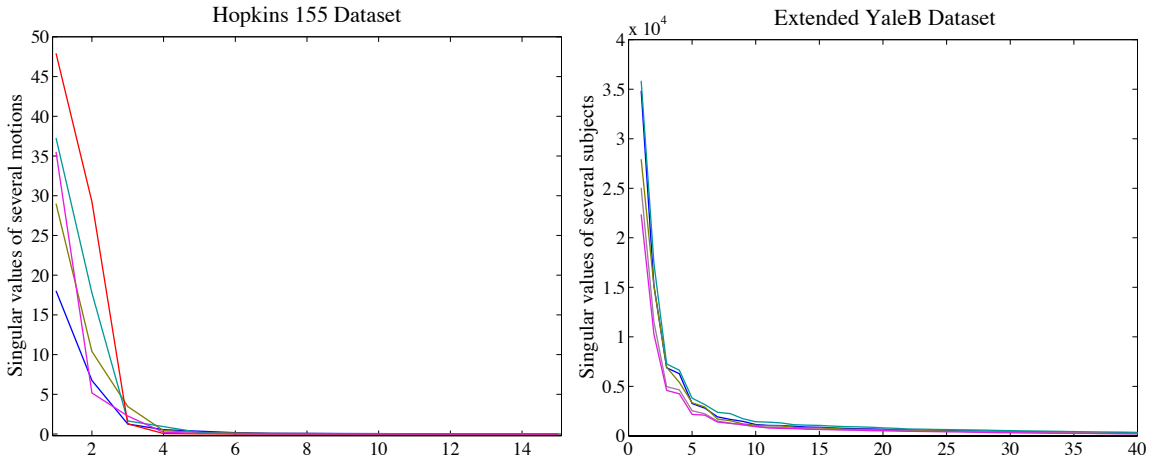


Figure 3.13: Left: singular values of several motions in the Hopkins 155 dataset. Each motion corresponds to a subspace of dimension at most 4. Right: singular values of several faces in the Extended Yale B dataset. Each subject corresponds to a subspace of dimension around 9.

of state-of-the-art subspace clustering methods for the problem of motion segmentation. To do so, we consider the Hopkins 155 dataset [103] that consists of 155 video sequences, where 120 of the videos have two motions and 35 of the videos have three motions. On average, in the dataset, each sequence of 2 motions has $N = 266$ feature trajectories and $F = 30$ frames, while each sequence of 3 motions has $N = 398$ feature trajectories and $F = 29$ frames. The left plot of Figure 3.13 shows the singular values of several motions in the dataset. Note that the first four singular values are nonzero and the rest of the singular values are very close to zero, corroborating the 4-dimensionality of the underlying linear subspace of each motion.[7] In addition, it shows that the feature trajectories of each video can be well modeled as data points that almost perfectly lie in a union of linear subspaces of dimension at most 4.

[7] If we subtract the mean of the data points in each motion from them, the singular values drop at 3, showing the 3-dimensionality of the affine subspaces.

Table 3.3: Clustering error (%) of different algorithms on the Hopkins 155 dataset with the $2F$-dimensional data points.

Algorithms	LSA	SCC	LRR	LRR-H	LRSC	SSC
2 Motions						
Mean	4.23	2.89	4.10	2.13	3.69	**1.52**
Median	0.56	**0.00**	0.22	**0.00**	0.29	**0.00**
3 Motions						
Mean	7.02	8.25	9.89	**4.03**	7.69	4.40
Median	1.45	**0.24**	6.22	1.43	3.80	0.56
All						
Mean	4.86	4.10	5.41	2.56	4.59	**2.18**
Median	0.89	**0.00**	0.53	**0.00**	0.60	**0.00**

Table 3.4: Clustering error (%) of different algorithms on the Hopkins 155 dataset with the $4n$-dimensional data points obtained by applying PCA.

Algorithms	LSA	SCC	LRR	LRR-H	LRSC	SSC
2 Motions						
Mean	3.61	3.04	4.83	3.41	3.87	**1.83**
Median	0.51	**0.00**	0.26	**0.00**	0.26	**0.00**
3 Motions						
Mean	7.65	7.91	9.89	4.86	7.72	**4.40**
Median	1.27	1.14	6.22	1.47	3.80	**0.56**
All						
Mean	4.52	4.14	5.98	3.74	4.74	**2.41**
Median	0.57	**0.00**	0.59	**0.00**	0.58	**0.00**

The results of applying subspace clustering algorithms to the dataset when we use the original $2F$-dimensional feature trajectories and when we project the data into a $4n$-dimensional subspace (n is the number of subspaces) using PCA are shown in Table 3.3 and Table 3.4, respectively. From the results, we make the following conclusions:

– In both cases, SSC obtains a small clustering error outperforming the other algorithms. This suggests that the separation of different motion subspaces in terms

of their principal angles and the distribution of the feature trajectories in each motion subspace are sufficient for the success of the sparse optimization program, hence clustering.

– Without post-processing of its coefficient matrix, LRR has higher errors than other algorithms. On the other hand, post-processing of the low-rank coefficient matrix significantly improves the clustering performance (LRR-H).

– LRSC tries to find a noise-free dictionary for data while finding their low-rank representation. This helps to improve over LRR. Also, note that the errors of LRSC are higher than the reported ones in [46]. This comes from the fact that [46] has used the erroneous compacc.m function from [32] to compute the errors.

– The clustering performances of each algorithm for the two cases of using the $2F$-dimensional feature trajectories and using the $4n$-dimensional PCA projections are close. This comes from the fact that the feature trajectories of n motions in a video almost perfectly lie in a $4n$-dimensional linear subspace of the $2F$-dimensional ambient space. Thus, projection using PCA onto a $4n$-dimensional subspace preserves the structure of the subspaces and the data, hence, for each algorithm, the clustering error in Table 3.3 is close to the error in Table 3.4.

To provide a compact summary of the statistics of the errors of different algorithms, we show the boxplots for the two cases of using $2F$- and $4n$-dimensional motion data in Figure 3.14. Each boxplot shows, for a given set of measures, the median value (the red line in the box), the 25% and 75% percentiles (the edges of the

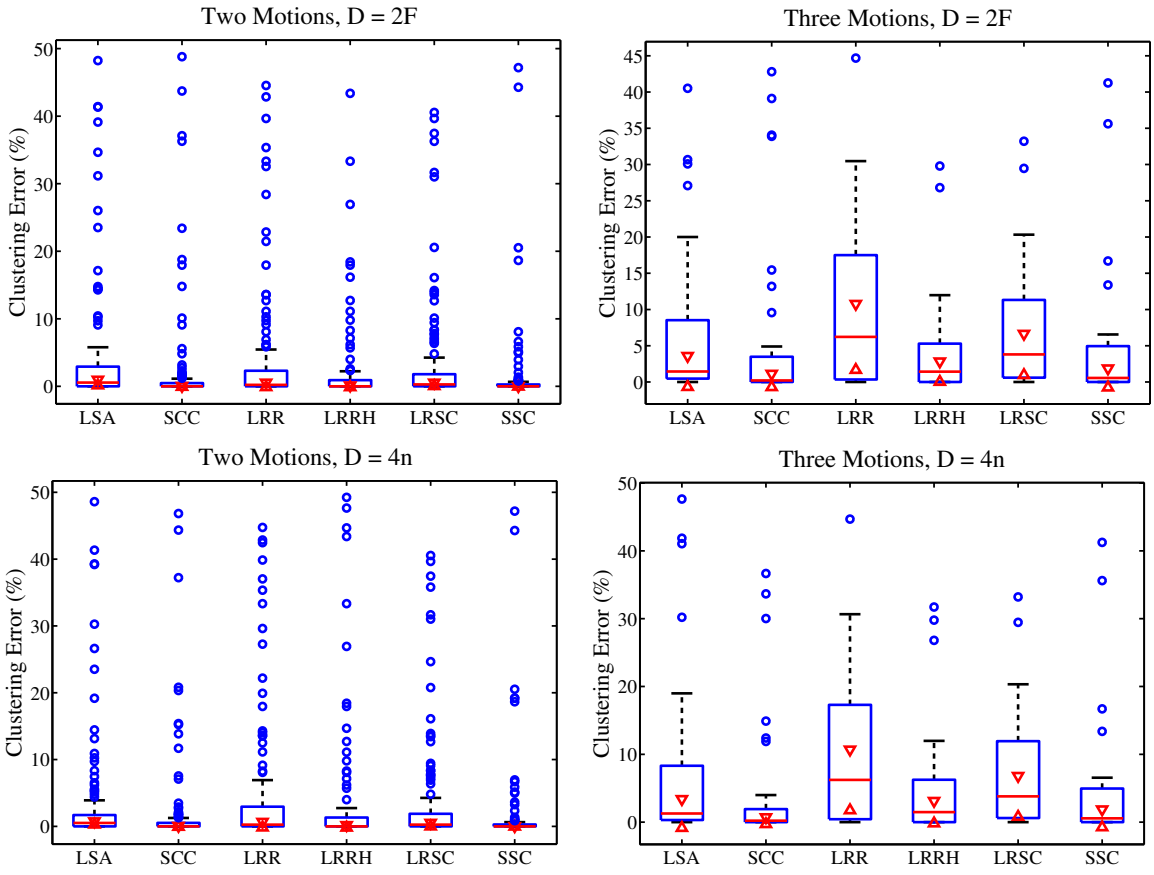


Figure 3.14: Boxplots of motions segmentation algorithms on the Hopkins 155 Dataset. Top: clustering errors (%) for two motions (left) and three motions (right) using $2F$-dimensional data. Bottom: clustering errors (%) for two motions (left) and three motions (right) using $4n$-dimensional data.

box) and the maximum and minimum values not considered outliers (the endpoints of the black whiskers that extend beyond the box). The boxplot also shows, in blue 'o', the outlier values that do not conform to the distribution satisfied by the majority of the values. Finally, the boxplot also marks in red circles, the 5% confidence intervals around the median for the distribution. Two medians are significantly different at the 5% significance level if their confidence intervals do not overlap. From Figure 3.14 one can see that SSC and SCC have a small boxplot length for two motions, showing

that most of the inliers are concentrated close to the median, which is 0. Increasing the number of motions to three makes the clustering problem harder and the lengths of the boxplots of all algorithms increase. However, note that SSC and SCC are least affected by the increase in the number of motions. In all cases, however, the 5% significance intervals of different methods have some overlap, which shows that the performances of the algorithms are statistically close. In the next experiments on face clustering, which has also larger number of subspaces, we will show that the SSC algorithm obtains statistically significant improvement over the state of the art.

3.8.3.1 Effect of the regularization parameter

Figure 3.15 shows the effect of the regularization parameter value in the clustering performance of SSC over the entire Hopkins 155 dataset, where we change α_z in the regularization parameter $\lambda_z = \alpha_z/\mu_z$. Note that the clustering errors of SSC as a function of α_z for the two cases of using the $2F$-dimensional data and the $4n$-dimensional data obtained by PCA follow a similar pattern. Moreover, for a large range of α_z, the clustering error is less than 2.5% for both cases.

3.8.3.2 Effect of the affine constraint

As mentioned before, for the motion segmentation experiments, we used the optimization program (3.13), with the affine constraint (and without the term $\boldsymbol{E}$ since the data do not have sparse outlying entries). The reason for using the affine constraint is

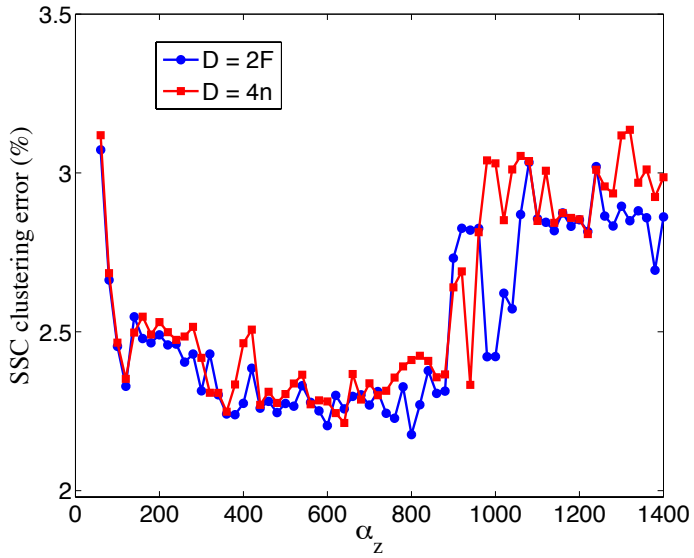


Figure 3.15: Clustering error (%) of SSC as a function of α_z in the regularization parameter $\lambda_z = \alpha_z/\mu_z$ for the two cases of clustering of $2F$-dimensional data and $4n$-dimensional data obtained by PCA.

that, under the affine projection model, the feature trajectories of each moving object lie in an affine subspace of dimension at most 3. However, most motion segmentation algorithms model the data in each moving object as lying in a linear subspace of dimension at most 4. Table 3.5 shows the results of the SSC algorithm for the two cases of using the $2F$-dimensional and $4n$-dimensional data with and without using the affine constraint in the optimization program. From the result that incorporating the affine model directly in to the optimization program helps to improve the segmentation results. Specifically, for the $2F$-dimension and $4n$-dimensional data, the affine constraint improves the average clustering error from 5.57% to 2.18% and from 5.71% to 2.41%, respectively, over the entire Hopkins 155 dataset.

Table 3.5: Clustering error (%) of the SSC algorithm on the Hopkins 155 dataset for $2F$-dimensional and $4n$-dimensional data points obtained by applying PCA, for the two cases of not using (Linear) and using (Affine) the affine constraint.

Algorithms	Linear (2F)	Affine (2F)	Linear (4n)	Affine (4n)
2 Motions				
Mean	4.04	**1.52**	4.08	**1.83**
Median	**0.00**	**0.00**	**0.00**	**0.00**
3 Motions				
Mean	10.82	**4.40**	11.30	**4.40**
Median	5.35	**0.56**	5.35	**0.56**
All				
Mean	5.57	**2.18**	5.71	**2.41**
Median	**0.00**	**0.00**	**0.00**	**0.00**

3.8.4 Motion segmentation with missing data and outlying entries

We now examine the robustness of SSC to missing data and outliers. We use twelve sequences from [112], with nine sequences of two motions and three sequences of three motions (see Figure 3.16). We compare the performance of SSC with the ALC algorithm [88], which can also deal with missing and outlying entires.

For incomplete trajectories, we apply SSC and ALC to video sequences with 4% to 35% missing entries. ALC first completes the data using the Power Factorization (PF) method, or using an ℓ_1-minimization. Once ALC obtains the complete data, it projects them into a lower dimensional space of dimension r. ALC uses $r = 5$ or $r = d_{\text{sp}}$, where d_{sp}, known as the sparsity preserving dimension, which is obtained by

$$d_{\text{sp}} = \operatorname{argmin} d \quad \text{s.t.} \quad d \geq 8 \log(2F/d). \tag{3.62}$$

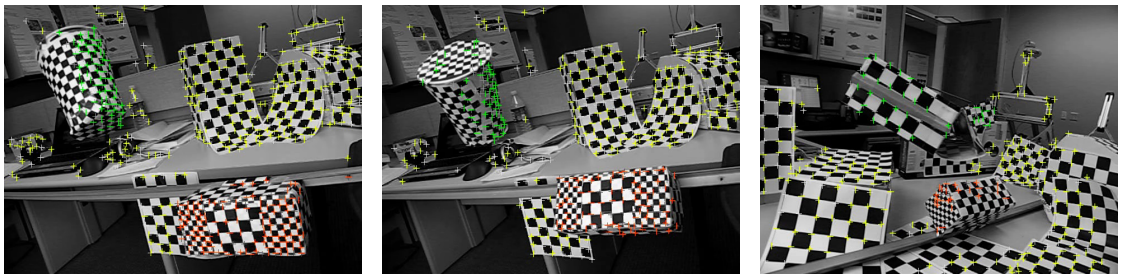


Figure 3.16: Example frames from three video sequences with incomplete or corrupted trajectories.

Table 3.6: Clustering error (%) of different algorithms for 12 real motion sequences with missing data.

Method	PF+ ALC_5	PF+ALC_{sp}	ℓ_1+ALC_5	ℓ_1+ALC_{sp}	SSC
Average	1.89	10.81	3.81	1.28	**0.13**
Median	0.39	7.85	0.17	1.07	0.00

For SSC, we use the data points in the original $2F$-dimensional ambient space without projecting them into lower dimensions. Table 3.6 shows the results of ALC and SSC on the 12 motion sequences with missing entries. Note that SSC achieves a misclassification error of 0.13% improving the performance of ALC.

For corrupted trajectories, we apply SSC and ALC to video sequences between 4% and 35% of whose entries are corrupted. Our results in Table 3.7 compared with the results of ℓ_1-based ALC indicate the robustness of SSC to outliers.

Note that in both cases of missing and outlying entries, in contrast to ALC, we do not need to use the ℓ_1-minimization as an initialization step to complete or correct the trajectories and then apply the segmentation algorithm. The sparse coefficients obtained by the ℓ_1-minimization are used directly to build the similarity graph to obtain the segmentation.

Table 3.7: Clustering error (%) of different algorithms for 12 real motion sequences with corrupted trajectories.

Method	$\ell_1 + \text{ALC}_5$	$\ell_1 + \text{ALC}_{\text{sp}}$	SSC
Average	4.15	3.02	**1.05**
Median	0.21	0.89	0.43

3.8.5 Face clustering

Given face images of multiple subjects, acquired with a fixed pose and varying illumination, we consider the problem of clustering images according to their subjects (Fig. 3.2). It has been shown that, under the Lambertian assumption, images of a subject with a fixed pose and varying illumination lie close to a linear subspace of dimension 9 [6]. Thus, the collection of face images of multiple subjects lie close to a union of 9-dimensional subspaces.

In this section, we evaluate the clustering performance of SSC as well as the state-of-the-art methods on the Extended Yale B dataset [73]. The dataset consists of 192×168 pixel cropped face images of $n = 38$ individuals, where there are $N_i = 64$ frontal face images for each subject acquired under various lighting conditions. To reduce the computational cost and the memory requirements of all algorithms we downsample the images to 48×42 pixels and treat each $2,016$-dimensional vectorized image as a data point, hence, $D = 2,016$. The right plot in Figure 3.13 shows the singular values of data points of several subjects in the dataset. Note that the singular value curve has a knee around 9, corroborating the approximate 9-dimensionality of the face data in each subject. In addition, the singular values gradually decay to zero,

showing that the data are corrupted by errors. Thus, the face images of n subjects can be modeled as corrupted data points lying close to a union of 9-dimensional subspaces.

To study the effect of the number of subjects in the clustering performance of different algorithms, we devise the following experimental setting: we divide the 38 subjects into 4 groups, where the first three groups correspond to subjects 1 to 10, 11 to 20, 21 to 30, and the fourth group corresponds to subjects 31 to 38. For each of the first three groups we consider all choices of $n \in \{2, 3, 5, 8, 10\}$ subjects and for the last group we consider all choices of $n \in \{2, 3, 5, 8\}$.[8] Finally, we apply clustering algorithms for each trial, i.e., each set of n subjects.

3.8.5.1 Applying RPCA separately on each subject

As we showed in the svd plot of the face data in Figure 3.13 (right), the face images do not perfectly lie in a linear subspace as they are corrupted by errors. In fact, the errors correspond to the cast shadows and specularities in the face images and can be modeled as sparse outlying entries. As a result, it is important for a subspace clustering algorithm to effectively deal with data with sparse corruptions.

In order to validate the fact that corruption of faces is due to sparse outlying errors and show the importance of dealing with corruptions while clustering, we start

[8]Note that choosing n out of 38 leads to extremely large number of trials. Thus, we have devised the above setting in order to have a repeatable experiment with a reasonably large number of trials for each n.

Table 3.8: Clustering error (%) of different algorithms on the Extended Yale B dataset after applying RPCA separately to the data points in each subject.

Algorithm	LSA	SCC	LRR	LRR-H	LRSC	SSC
2 Subjects						
Mean	6.15	1.29	0.09	0.05	**0.00**	0.06
Median	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
3 Subjects						
Mean	11.67	19.33	0.12	0.10	**0.00**	0.08
Median	2.60	8.59	**0.00**	**0.00**	**0.00**	**0.00**
5 Subjects						
Mean	21.08	47.53	0.16	0.15	**0.00**	0.07
Median	19.21	47.19	**0.00**	**0.00**	**0.00**	**0.00**
8 Subjects						
Mean	30.04	64.20	4.50	11.57	**0.00**	0.06
Median	29.00	63.77	0.20	15.43	**0.00**	**0.00**
10 Subjects						
Mean	35.31	63.80	0.15	13.02	**0.00**	0.89
Median	30.16	64.84	**0.00**	13.13	**0.00**	0.31

by the following experiment. We apply the Robust Principal Component Analysis (RPCA) algorithm [18] to remove the sparse outlying entries of the face data in each subject. Note that *in practice*, we do not know the clustering of the data beforehand, hence cannot apply the RPCA to the faces of each subject. However, as we will show, this experiment illustrates some of the challenges of the face clustering and validates several conclusions about the performances of different algorithms.

Table 3.8 shows the clustering error of different algorithms after applying RPCA to the data points in each subject and removing the sparse outlying entries, i.e., after bringing the data points back to their low-dimensional subspaces. From the results, we make the following conclusions:

– The clustering error of SSC is very close to zero for different number of subjects

suggesting that SSC can deal well with face clustering if the face images are corruption free. In other words, while the data in different subspaces are very close to each other, as shown in Figure 3.12 (right), the performance of the SSC is more dependent on the principal angles between subspaces which, while small, are large enough for the success of SSC.

– The LRR and LRSC algorithms have also low clustering errors (LRSC obtains zero errors) showing the effectiveness of removing sparse outliers in the clustering performance. On the other hand, while LRR-H has a low clustering error for 2, 3, and 5 subjects, it has a relatively large error for 8 and 10 subjects, showing that the post processing step on the obtained low-rank coefficient matrix not always improves the result of LRR.

– For LSA and SCC, the clustering error is relatively large and the error increases as the number of subjects increases. This comes from the fact that, as shown in Figure 3.12 (right), for face images, the neighborhood of each data point contains points that belong to other subjects and, in addition, the number of neighbors from other subjects increases as we increase the number of subjects.

3.8.5.2 Applying RPCA simultaneously on all subjects

In practice, we cannot apply RPCA separately to the data in each subject because the clustering is unknown. In this section, we deal with sparse outlying entries in the data by applying the RPCA algorithm to the collection of all data points for each

Table 3.9: Clustering error (%) of different algorithms on the Extended Yale B dataset after applying RPCA simultaneously to all the data in each trial.

Algorithm	LSA	SCC	LRR	LRR-H	LRSC	SSC
2 Subjects						
Mean	32.53	9.29	7.27	5.72	5.67	**2.09**
Median	47.66	7.03	6.25	3.91	4.69	**0.78**
3 Subjects						
Mean	53.02	32.00	12.29	10.01	8.72	**3.77**
Median	51.04	37.50	11.98	9.38	8.33	**2.60**
5 Subjects						
Mean	58.76	53.05	19.92	15.33	10.99	**6.79**
Median	56.87	51.25	19.38	15.94	10.94	**5.31**
8 Subjects						
Mean	62.32	66.27	31.39	28.67	16.14	**10.28**
Median	62.50	64.84	33.30	31.05	14.65	**9.57**
10 Subjects						
Mean	62.40	63.07	35.89	32.55	21.82	**11.46**
Median	62.50	60.31	34.06	30.00	25.00	**11.09**

trial prior to clustering. The results are shown in Table 3.9 from which we make the following conclusions:

– The clustering error for SSC is low for all different number of subjects. Specifically, SSC obtains 2.09% and 11.46% for clustering of data points in 2 and 10 subjects, respectively.

– Applying RPCA to all data points simultaneously may not be as effective as applying RPCA to data points in each subject separately. This comes from the fact that RPCA tends to bring the data points into a common low-rank subspace which can result in decreasing the principal angles between subspaces and decreasing the distances between data points in different subjects. This can explain the increase in the clustering error of all clustering algorithms with respect to the results in Table

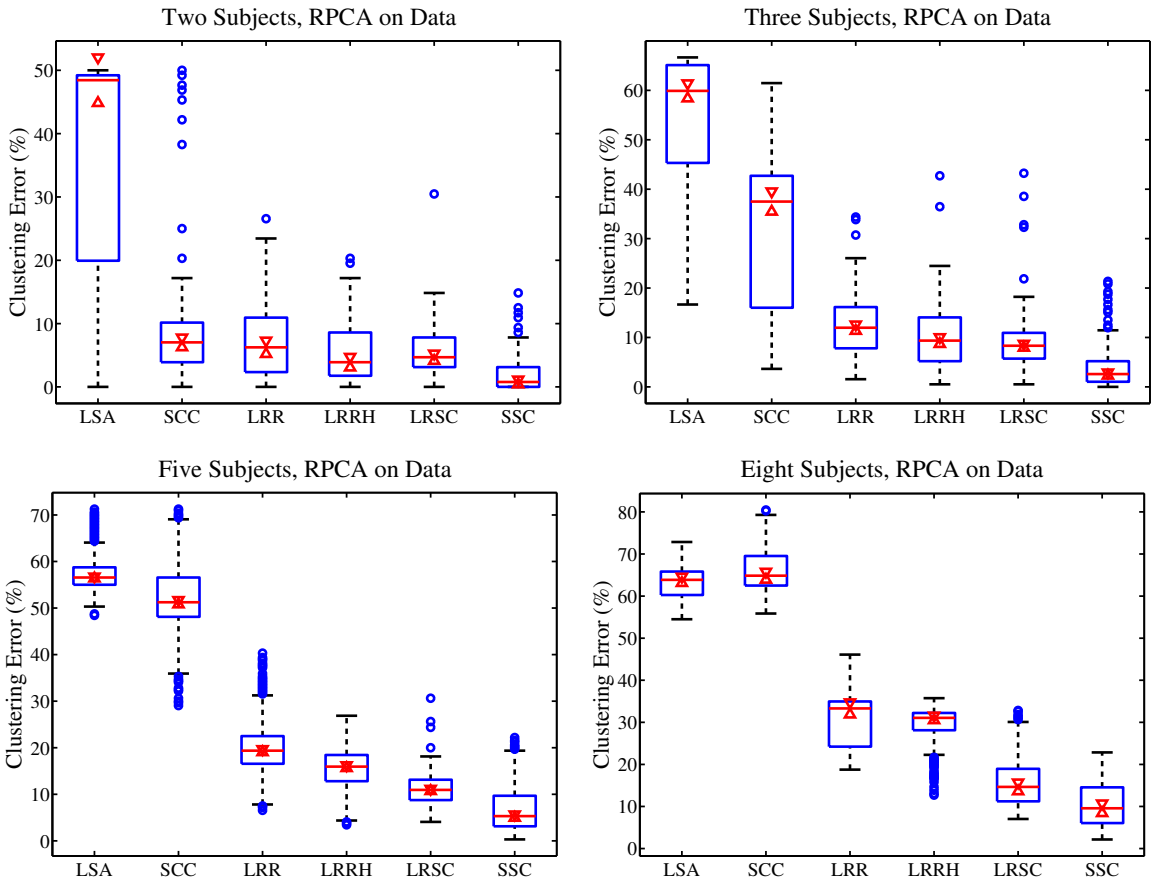


Figure 3.17: Boxplots for face clustering results on the Extended Yale B Dataset using face image data in $D = 2,016$-dimensional space after applying RPCA to each trial. Top: clustering errors (%) for two subjects (left) and three subjects (right). Bottom: clustering errors (%) for five subjects (left) and eight subjects (right).

3.8.

Figure 3.17 shows the boxplots of the clustering errors of all the algorithms for $\{2, 3, 5, 8\}$ subjects (there are three trials for 10 subjects, thus we do not shows the boxplot for it). From the result, one can see that, for all cases, the SSC algorithm obtains statistically significant improvement over the state of the art, since the 5% confidence intervals of SSC are clearly separated from and lower then those of the other algorithms.

Table 3.10: Clustering error (%) of different algorithms on the Extended Yale B dataset without pre-processing the data.

Algorithm	LSA	SCC	LRR	LRR-H	LRSC	SSC
2 Subjects						
Mean	32.80	16.62	9.52	2.54	5.32	**1.86**
Median	47.66	7.82	5.47	0.78	4.69	**0.00**
3 Subjects						
Mean	52.29	38.16	19.52	4.21	8.47	**3.10**
Median	50.00	39.06	14.58	2.60	7.81	**1.04**
5 Subjects						
Mean	58.02	58.90	34.16	6.90	12.24	**4.31**
Median	56.87	59.38	35.00	5.63	11.25	**2.50**
8 Subjects						
Mean	59.19	66.11	41.19	14.34	23.72	**5.85**
Median	58.59	64.65	43.75	10.06	28.03	**4.49**
10 Subjects						
Mean	60.42	73.02	38.85	22.92	30.36	**10.94**
Median	57.50	75.78	41.09	23.59	28.75	**5.63**

3.8.5.3 Using original data points

Finally, we apply the clustering algorithms to the original data points without pre-processing the data. The results are shown in Table 3.10 from which we make the following conclusions:

– The SSC algorithm obtains a low clustering error for all numbers of subjects, obtaining 1.86% and 10.94% clustering error for 2 and 10 subjects, respectively. In fact, the error is smaller than applying RPCA to all data points. This is due to the fact that SSC directly incorporates the corruption model of the data by sparse outlying entries into the sparse optimization program, giving it the ability to perform clustering on the corrupted data.

– While LRR also has a regularization term to deal with the corrupted data, the

clustering error is relatively large especially as the number of subjects increases. This can be due to the fact that there is not a clear relationship between corruption of each data point and the LRR regularization term in general [76]. On the other hand, the post processing step of LRR-H on the low-rank coefficient matrix helps to significantly reduce the clustering error, although it is larger than the SSC error.

– As LRSC tries to recover error-free data points while finding their low-rank representation, it obtains smaller errors than LRR.

– LSA and SCC do not have an explicit way to deal with corrupted data. This together with the fact that the face images of each subject have relatively a large number of neighbors in other subjects, as shown in Figure 3.12 (right), result in low performances of these algorithms.

Figure 3.18 shows the boxplots for the clustering errors of all the algorithm for $\{2, 3, 5, 8\}$ subjects. From the result, one can see that, in all cases, the 5% confidence intervals of SSC are clearly separated from and lower than those of the other algorithms. Thus, the SSC algorithm obtains a statistically significant improvement over the state of the art.

3.8.5.4 Computational time comparison

The average computational time of each algorithm as a function of the number of subjects (or equivalently the number of data points) is shown in Figure 3.19. Note that the computational time of SCC is drastically higher than other algorithms.

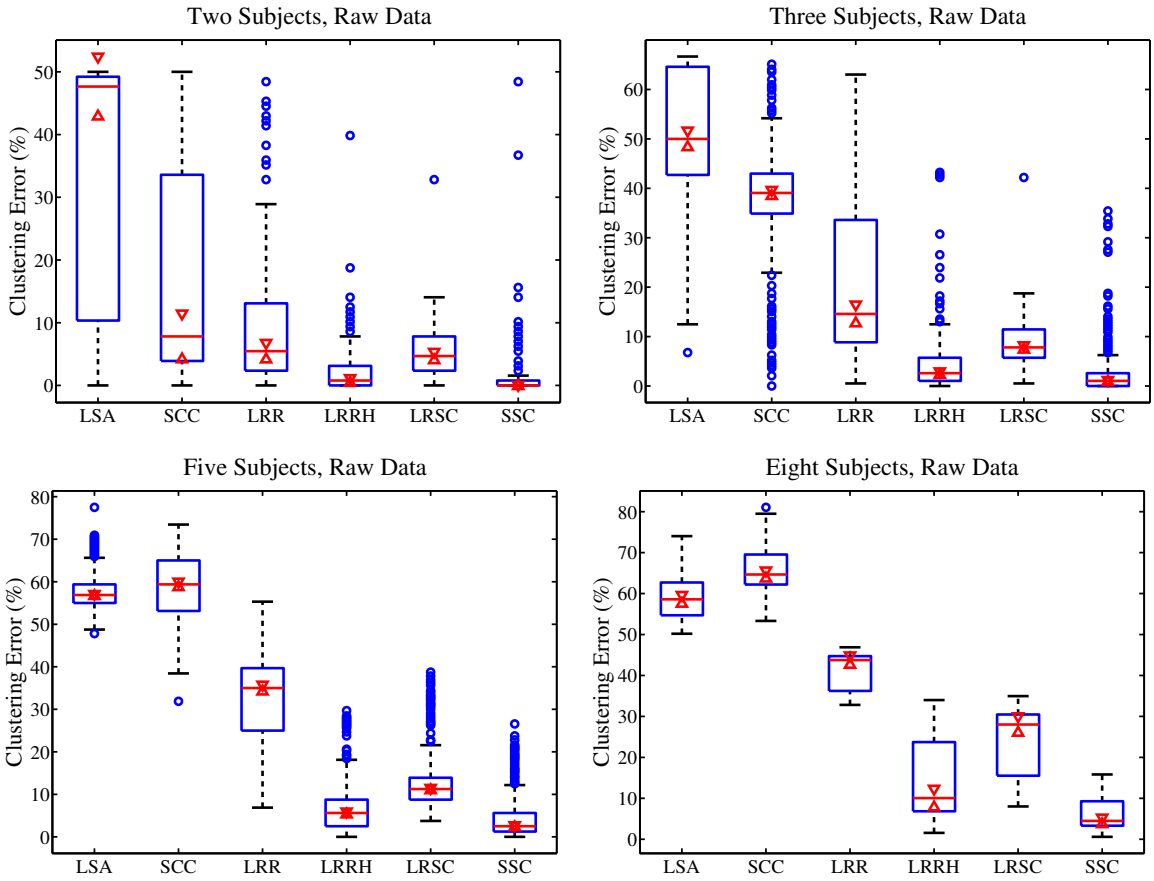


Figure 3.18: Boxplots for face clustering results on the Extended Yale B Dataset using raw face image data in $D = 2,016$-dimensional space. Top: clustering errors (%) for two subjects (left) and three subjects (right). Bottom: clustering errors (%) for five subjects (left) and eight subjects (right).

This comes from the fact that the complexity of SCC increases exponentially in the dimension of the subspaces, which in this case is $d = 9$. On the other hand, SSC, LRR and LRSC use fast and efficient convex optimization techniques which keeps their computational time lower than other algorithms.

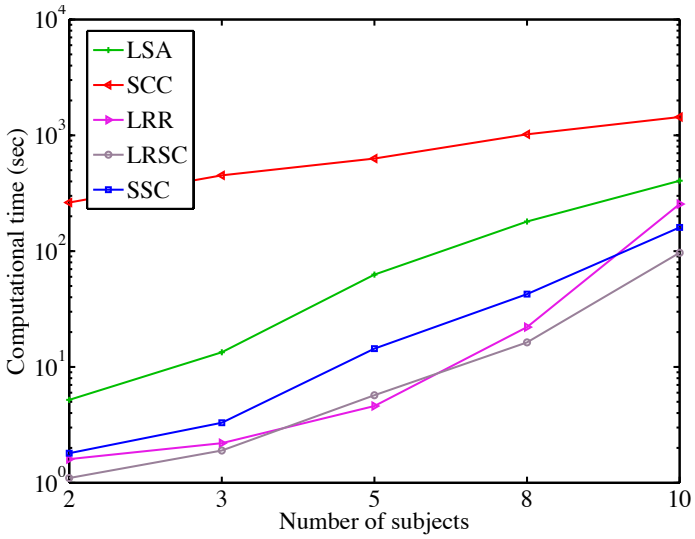


Figure 3.19: Average computational time (sec.) of the algorithms on the Extended Yale B dataset as a function of the number of subjects.

3.9 Conclusions

We studied the problem of clustering a collection of data points that lie in or close to a union of low-dimensional subspaces. We proposed a subspace clustering algorithm based on sparse representation techniques, called SSC, that finds a sparse representation of each point in the dictionary of the other points, builds a similarity graph using the sparse coefficients, and obtains the segmentation of the data using spectral clustering. We showed that, under appropriate conditions on the arrangement of the subspaces and the distribution of the data, the algorithm succeeds in recovering the desired sparse representation of the data points. A key advantage of the algorithm is its ability to directly deal with data nuisances, such as noise, sparse outlying entries, and missing entries as well as the more general class of affine subspaces by incorporating the corresponding models into the sparse optimization

program. Experiments on real data such as face images and motions in videos show the effectiveness of our algorithm and its superiority over the state of the art.

3.10 Appendix

In this appendix, we provide the proof of In this section, Proposition 1. We use the following Lemma whose proof can be found in [48].

Lemma 1 *Consider the optimization program*

$$\boldsymbol{c}^* = \text{argmin}_{\boldsymbol{c}} \ \ \|\boldsymbol{c}\|_1 + \frac{\lambda}{2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{c}\|_2^2. \tag{3.63}$$

For $\lambda < \|\boldsymbol{A}^\top \boldsymbol{y}\|_\infty$*, we have* $\boldsymbol{c}^* = \boldsymbol{0}$*.*

3.10.1 Proof of Proposition 1

Note that solving the optimization program (3.13) is equivalent to solving N optimization programs as

$$\begin{aligned} \min \ \ & \|\boldsymbol{c}_i\|_1 + \lambda_e\|\boldsymbol{e}_i\|_1 + \frac{\lambda_z}{2}\|\boldsymbol{z}_i\|_2^2 \\ \text{s.t.} \ \ & \boldsymbol{y}_i = \boldsymbol{Y}\boldsymbol{c}_i + \boldsymbol{e}_i + \boldsymbol{z}_i, \ \ c_{ii} = 0, \end{aligned} \tag{3.64}$$

where $\boldsymbol{c}_i$, $\boldsymbol{e}_i$, and $\boldsymbol{z}_i$ are the i-th columns of $\boldsymbol{C}$, $\boldsymbol{E}$, and $\boldsymbol{Z}$, repsectively.

(a) Consider the optimization program (3.64) without the term $\boldsymbol{z}_i$ and denote the objective function value by

$$\text{cost}(\boldsymbol{c}_i, \boldsymbol{e}_i) \triangleq \|\boldsymbol{c}_i\|_1 + \lambda_e\|\boldsymbol{e}_i\|_1. \tag{3.65}$$

Note that a feasible solution of (3.64) is given by $(\mathbf{0}, \boldsymbol{e}_i)$ for which the value of the objective function is equal to

$$\text{cost}(\mathbf{0}, \boldsymbol{e}_i) = \lambda_e \|\boldsymbol{y}_i\|_1. \tag{3.66}$$

On the other hand, using matrix norm properties, for any feasible solution $(\boldsymbol{c}_i, \boldsymbol{e}_i)$ of (3.64) we have

$$\|\boldsymbol{y}_i\|_1 = \|\boldsymbol{Y}\boldsymbol{c}_i + \boldsymbol{e}_i\|_1 \leq (\max_{j \neq i} \|\boldsymbol{y}_j\|_1) \, \|\boldsymbol{c}_i\|_1 + \|\boldsymbol{e}_i\|_1, \tag{3.67}$$

where we used the fact that $c_{ii} = 0$. Multiplying both sides of the above inequality by λ_e we obtain

$$\text{cost}(\mathbf{0}, \boldsymbol{y}_i) = \lambda_e \|\boldsymbol{Y}\|_1 \leq (\lambda_e \max_{j \neq i} \|\boldsymbol{y}_j\|_1) \, \|\boldsymbol{c}_i\|_1 + \lambda_e \|\boldsymbol{e}_i\|_1, \tag{3.68}$$

Note that if $\lambda_e < \frac{1}{\max_{j \neq i} \|\boldsymbol{y}_j\|_1}$, then from the above equation we have

$$\text{cost}(\mathbf{0}, \boldsymbol{y}_i) \leq \text{cost}(\boldsymbol{c}_i, \boldsymbol{e}_i). \tag{3.69}$$

In other words, $(\boldsymbol{c}_i = \mathbf{0}, \boldsymbol{e}_i = \boldsymbol{y}_i)$ achieve the minimum cost among all feasible solutions of (3.64). Hence, if $\lambda_e < \max_i \frac{1}{\max_{j \neq i} \|\boldsymbol{y}_j\|_1}$, then there exists $\ell \in \{1, \cdots, N\}$ such that in the solution of the optimization program (3.13) we have $(\boldsymbol{c}_\ell, \boldsymbol{e}_\ell) = (\mathbf{0}, \boldsymbol{y}_\ell)$.

(b) Consider the optimization program (3.64) without the term $\boldsymbol{e}_i$, which, using $\boldsymbol{z}_i = \boldsymbol{y}_i - \boldsymbol{Y}\boldsymbol{c}_i$, can be rewritten as

$$\min \;\; \|\boldsymbol{c}_i\|_1 + \frac{\lambda_z}{2} \|\boldsymbol{y}_i - \boldsymbol{Y}\boldsymbol{c}_i\|_2^2 \;\; \text{s.t.} \;\; c_{ii} = 0. \tag{3.70}$$

From Lemma 1 we have that, for $\lambda_z < \frac{1}{\max_{j \neq i} |\boldsymbol{y}_j{}^\top \boldsymbol{y}_i|}$, the solution of (3.70) is equal to $\boldsymbol{c}_i = 0$, or equivalently, the solution of (3.64) is given by $(\boldsymbol{c}_i, \boldsymbol{z}_i) = (\boldsymbol{0}, \boldsymbol{y}_i)$. As a result, if $\lambda_z < \max_i \frac{1}{\max_{j \neq i} |\boldsymbol{y}_j^\top \boldsymbol{y}_i|}$, then there exists $\ell \in \{1, \cdots, N\}$ such that in the solution of the optimization program (3.13) we have $(\boldsymbol{c}_\ell, \boldsymbol{z}_\ell) = (\boldsymbol{0}, \boldsymbol{y}_\ell)$.

Chapter 4

Sparse Manifold Clustering & Embedding

In Chapter 3, we considered the problem of clustering data that lie in a union of subspaces. While flat manifolds (subspaces) model well the distribution of data in several real-world problems, in general, real data lie in nonlinear manifolds. Since there is not a global linear relationship among the data in the same nonlinear manifold, subspace-based methods can not be used in general for clustering of data lying in a union of nonlinear manifolds. Figure 4.1 illustrates this point, by showing the similarity matrices of SSC, LRR, and LSA, studied in the previous chapter, for two 1-dimensional nonlinear manifolds. Notice that LSA and LRR obtain large similarities between points in different manifolds, while SSC obtains nonzero weights between a point in a manifold and some of the points in the other manifold. Hence, these

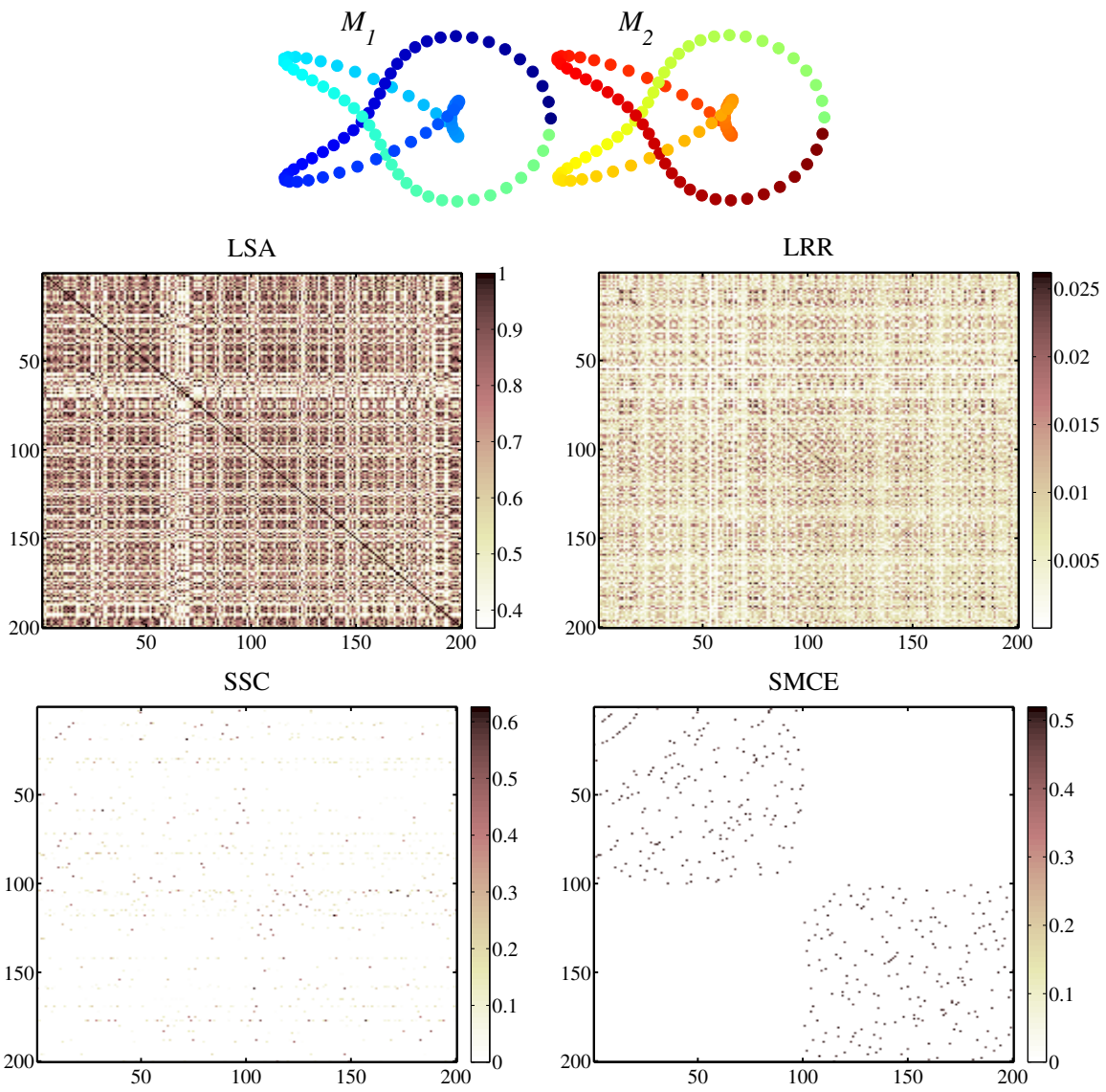


Figure 4.1: Subspace clustering, in general, cannot deal with nonlinear manifolds. Top: 200 data points in two 1-dimensional nonlinear manifolds embedded in $\mathbb{R}^{100}$. Data points are ordered such that the first 100 points are in the first manifolds and the next 100 points are in the second manifold. Middle: similarity matrices obtained by LSA and LRR. Bottom: similarity matrix obtained by SSC is shown in the left plot. Similarity matrix obtained by SMCE, proposed in this chapter, is shown in the right plot. Note that the three subspace clustering methods obtain nonzero weights between points in different manifolds, while SMCE obtains zero weights between points in different manifolds.

algorithms cannot separate the data into the underlying manifolds. As a result, there is a need for algorithms that can effectively cluster data lying in a union of nonlinear manifolds. On the other hand, when it comes to the problem of dimensionality reduction in nonlinear manifolds, it is often the case that linear dimensionality reduction algorithms cannot effectively unravel the low-dimensional representation of the data. Thus, there is a need for having methods that, by exploiting the nonlinear structure of the manifolds, can effectively reduce the dimension of the data.

In this Chapter, we consider the problem of clustering and dimensionality reduction of data lying in a union of nonlinear manifolds. We propose an algorithm called Sparse Manifold Clustering and Embedding (SMCE) for simultaneous clustering and dimensionality reduction of data lying in multiple nonlinear manifolds [43]. Unlike the SSC algorithm, studied in the previous chapter, that can select any point from the same subspace, the SMCE algorithm promotes selecting a few data points that are sufficiently close to the given point and span an affine subspace close to that point.

More specifically, similar to most dimensionality reduction methods, SMCE finds a small neighborhood around each data point and connects each point to its neighbors with appropriate weights. The key difference is that SMCE finds both the neighbors and the weights automatically and at the same time. This is done by solving a sparse optimization problem, which encourages selecting nearby points that lie in the same manifold and approximately span a low-dimensional affine subspace. The optimal solution encodes information that can be used for clustering and dimensionality re-

duction using spectral clustering and embedding. Moreover, the size of the optimal neighborhood of a data point, which can be different for different points, provides an estimate of the dimension of the manifold to which the point belongs. Through experiments we demonstrate that the proposed method can effectively handle multiple manifolds that are very close to each other, manifolds with non-uniform sampling and holes, as well as estimate the intrinsic dimensions of the manifolds.

Before presenting the proposed algorithm, we review existing approaches for dimensionality reduction and clustering of data lying in nonlinear manifolds.

4.1 A review of manifold clustering and embedding algorithms

4.1.1 Manifold embedding

In real-world problems, we are often confronted with high-dimensional data that lie in or close to a manifold of intrinsically low-dimension. As discussed in Chapter 1, it is important to perform dimensionality reduction, i.e., to find a compact representation of the data that unravels their few degrees of freedom.

The first step of most dimensionality reduction methods is to build a neighborhood graph by connecting each data point to a fixed number of nearest neighbors or to all points within a certain radius of the given point (see Chapter 2 for a review).

Local methods, such as Locally Linear Embedding (LLE) [91], Hessian LLE (HLLE) [28] and Laplacian eigenmaps (LEM) [7], try to preserve local relationships among points by learning a set of weights between each point and its neighbors. The main difference between these methods is in the procedures for learning these weights. Global methods, such as ISOMAP [98], Local Tangent Space Alignment (LTSA) [125], Maximum Variance Unfolding (MVU), [114], Minimum Volume Embedding (MVE) [92] and Structure Preserving Embedding (SPE) [93], try to preserve local and global relationships among all data points. For example, ISOMAP tries to preserve the geodesic distance between any pair of points, after approximating it by the length of the shortest path between points in the neighborhood graph. Both categories of methods find the low-dimensional representation of the data from a few eigenvectors of a matrix related to the learned weights between pairs of points.

For both local and global methods, a proper choice of the neighborhood size used to build the neighborhood graph is critical. Specifically, a small neighborhood size may not capture sufficient information about the manifold geometry, especially when it is smaller than the intrinsic dimension of the manifold. On the other hand, a large neighborhood size could violate the principles used to capture information about the manifold. Moreover, the curvature of the manifold and the density of the data points may be different in different regions of the manifold, hence using a fix neighborhood size may be inappropriate.

4.1.2 Manifold clustering

In many real-world problems, data lie in multiple manifolds of possibly different dimensions. Thus, to find a low-dimensional embedding of the data, one needs to first cluster the data according to the underlying manifolds and then find a low-dimensional representation for the data in each cluster. Since the manifolds can be very close to each other and they can have arbitrary dimensions, curvature and sampling, the manifold clustering and embedding problem is very challenging.

As discussed in Chapter 3, the particular case of clustering data lying in multiple flat manifolds (subspaces) is well studied and numerous algorithms have been proposed (see e.g., the tutorial [109]). However, such algorithms take advantage of the global linear relationships among data points in the same subspace, hence they cannot handle nonlinear manifolds. Other methods assume that the manifolds have different intrinsic dimensions and cluster the data according to the dimensions rather than the manifolds themselves [5,54,60,74,82]. However, in many real-world problems this assumption is violated. Moreover, estimating the dimension of a manifold from a point cloud is a very difficult problem on its own.

The work of [95] develops an EM-like extension of ISOMAP for clustering multiple nonlinear manifolds. However, this method is sensitive to a good initialization and is not a principled EM method since it uses heuristics in the E-step to assign points to manifolds.

When manifolds are densely sampled and sufficiently separated, existing dimen-

sionality reduction algorithms such as LLE and LEM can be extended to perform clustering before the dimensionality reduction step [2, 8, 56, 87] (see Chapter 2 for a review). More precisely, if the size of the neighborhood used to build the similarity graph is chosen to be small enough not to include points from other manifolds and large enough to capture the local geometry of the manifold, then the similarity graph will have multiple connected components, one per manifold. Therefore, spectral clustering methods can be employed to separate the data according to the connected components. However, as we will see later, finding the right neighborhood size is in general difficult, especially when manifolds are close to each other. Moreover, in some cases one cannot find a neighborhood that contains only points from the same manifold.

4.2 Sparse manifold clustering and embedding algorithm

In this section, we propose an algorithm based on sparse representation techniques for clustering and embedding of data in multiple nonlinear manifolds. Unlike conventional methods that first build a neighborhood graph and then extract information from it, our method simultaneously builds the neighborhood graph and obtains its weights. This leads to successful results even in challenging situations where the manifolds are spatially close to each other.

Assume we are given a collection of N data points $\{\boldsymbol{y}_i \in \mathbb{R}^D\}_{i=1}^N$ lying in n different manifolds $\{\mathcal{M}_\ell\}_{\ell=1}^n$ of intrinsic dimensions $\{d_\ell\}_{\ell=1}^n$. In this section, we consider the problem of simultaneously clustering the data according to the underlying manifolds and obtaining a low-dimensional representation of the data points in each cluster.

We approach this problem using a spectral clustering and embedding algorithm. Specifically, we build a similarity graph whose nodes represent the data points and whose edges represent the similarity between the data points. The fundamental challenge is to decide which nodes should be connected and how. To do clustering, we wish to connect each point to other points from the same manifold. To do dimensionality reduction, we wish to connect each point to neighboring points with appropriate weights that reflect the neighborhood information. To simultaneously pursue both goals, we wish to select neighboring points from the same manifold.

We address this problem by formulating an optimization algorithm based on sparse representation. The underlying assumption behind the proposed method is that each data point has a small neighborhood in which the minimum number of points whose affine span passes near the given point corresponds to the data points from the same manifold.

Assumption 1 *For each data point $\boldsymbol{y}_i \in \mathcal{M}_\ell$ consider the smallest ball $\mathcal{B}_i \subset \mathbb{R}^D$ that contains the $d_\ell + 1$ nearest neighbors of $\boldsymbol{y}_i$ from $\mathcal{M}_\ell$. Let the neighborhood $\mathcal{N}_i$ be the set of all data points in $\mathcal{B}_i$ excluding $\boldsymbol{y}_i$. In general, this neighborhood contains points from $\mathcal{M}_\ell$ as well as other manifolds. We assume that for all i there exists $\epsilon \geq 0$ such*

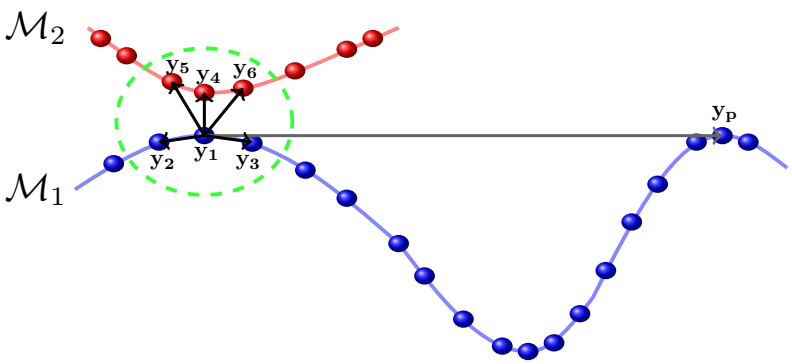


Figure 4.2: For $\boldsymbol{y}_1 \in \mathcal{M}_1$, the smallest neighborhood containing points from $\mathcal{M}_1$ also contains points from $\mathcal{M}_2$. However, the minimum number of points in this neighborhood whose affine span passes close to $\boldsymbol{y}_1$ corresponds to the two data points from $\mathcal{M}_1$.

that the nonzero entries of the sparsest solution of

$$\left\| \sum_{j \in \mathcal{N}_i} c_{ij} \frac{\boldsymbol{y}_j - \boldsymbol{y}_i}{\|\boldsymbol{y}_j - \boldsymbol{y}_i\|_2} \right\|_2 \leq \epsilon \quad \textit{and} \quad \sum_{j \in \mathcal{N}_i} c_{ij} = 1 \tag{4.1}$$

correspond to the $d_\ell + 1$ neighbors of $\boldsymbol{y}_i$ from $\mathcal{M}_\ell$. In other words, among all affine subspaces spanned by subsets of the vectors $\{\frac{\boldsymbol{y}_j - \boldsymbol{y}_i}{\|\boldsymbol{y}_j - \boldsymbol{y}_i\|_2}\}_{j \in \mathcal{N}_i}$ and passing near $\boldsymbol{y}_i$ up to ϵ error, the one that has the lowest dimension corresponds to the $d_\ell + 1$ neighbors of $\boldsymbol{y}_i$ from $\mathcal{M}_\ell$ and has dimension d_ℓ.

In the limiting case of densely sampled data, this affine subspace coincides with the d_ℓ-dimensional tangent space of $\mathcal{M}_\ell$ at $\boldsymbol{y}_i$. To illustrate this, consider the two manifolds shown in Figure 4.2 and assume that points $\boldsymbol{y}_4$, $\boldsymbol{y}_5$ and $\boldsymbol{y}_6$ are closer to $\boldsymbol{y}_1$ than $\boldsymbol{y}_2$ or $\boldsymbol{y}_3$. Then, any small ball centered at $\boldsymbol{y}_1 \in \mathcal{M}_1$ that contains $\boldsymbol{y}_2$ and $\boldsymbol{y}_3$ will also contain $\boldsymbol{y}_4$, $\boldsymbol{y}_5$ and $\boldsymbol{y}_6$. In this case, among affine spans of all possible choices of $d_1 + 1 = 2$ vectors $\{(\boldsymbol{y}_j - \boldsymbol{y}_i)/\|\boldsymbol{y}_j - \boldsymbol{y}_i\|_2\}_{j=2}^{6}$ in this neighborhood, the one corresponding to $j = 2$ and $j = 3$ is the closest one to $\boldsymbol{y}_1$, and is also close to the

tangent space of $\mathcal{M}_1$ at $\boldsymbol{y}_1$. On the other hand, while the affine span of any 3 or more data points in the neighborhood may pass through $\boldsymbol{y}_1$, it requires a combination of more than 2 data points. Hence, the minimum number of points in the neighborhood of $\boldsymbol{y}_1$ whose affine span passes close to $\boldsymbol{y}_1$ corresponds to the two data points from $\mathcal{M}_1$.

4.2.1 Optimization algorithm

Our goal is to propose a method that selects, for each data point $\boldsymbol{y}_i$, a few neighbors that lie in the same manifold. If the neighborhood $\mathcal{N}_i$ were known and of relatively small size, one could search for the minimum number of points that satisfy (4.1). However, $\mathcal{N}_i$ is not known a priori and searching for a few data points in $\mathcal{N}_i$ that satisfy (4.1) becomes more computationally complex as the size of the neighborhood increases. To tackle this problem, we let the size of the neighborhood be arbitrary. However, by using a sparse optimization program, we bias the method to select a few data points that are close to $\boldsymbol{y}_i$ and span a low-dimensional affine subspace passing near $\boldsymbol{y}_i$.

Consider a point $\boldsymbol{y}_i$ in the d_ℓ-dimensional manifold $\mathcal{M}_\ell$ and let

$$\boldsymbol{Y}_i \triangleq \begin{bmatrix} \frac{\boldsymbol{y}_1-\boldsymbol{y}_i}{\|\boldsymbol{y}_1-\boldsymbol{y}_i\|_2} & \cdots & \frac{\boldsymbol{y}_N-\boldsymbol{y}_i}{\|\boldsymbol{y}_N-\boldsymbol{y}_i\|_2} \end{bmatrix} \in \mathbb{R}^{D\times N-1}. \tag{4.2}$$

It follows from Assumption 1 that, among the columns of $\boldsymbol{Y}_i$, the ones that correspond to the neighbors of $\boldsymbol{y}_i$ in $\mathcal{M}_\ell$ span a d_ℓ-dimensional affine subspace of $\mathbb{R}^D$ that passes

near $\boldsymbol{y}_i$. In other words,

$$\|\boldsymbol{Y}_{-i}\boldsymbol{c}_i\|_2 \leq \epsilon \;\text{ and }\; \mathbf{1}^\top \boldsymbol{c}_i = 1 \tag{4.3}$$

has a solution $\boldsymbol{c}_i$ whose $d_\ell + 1$ nonzero entries correspond to $d_\ell + 1$ neighbors of $\boldsymbol{y}_i$ in $\mathcal{M}_\ell$.

Notice that after relaxing the size of the neighborhood, the solution $\boldsymbol{c}_i$ that uses the minimum number of vectors, i.e., the solution $\boldsymbol{c}_i$ with the smallest number of nonzero entries, may no longer be unique. In the example of Figure 4.2, for instance, a solution of (4.3) with two nonzero entries can correspond to an affine combination of $(\boldsymbol{y}_2 - \boldsymbol{y}_1)/\|\boldsymbol{y}_2 - \boldsymbol{y}_1\|_2$ and $(\boldsymbol{y}_3 - \boldsymbol{y}_1)/\|\boldsymbol{y}_3 - \boldsymbol{y}_1\|_2$ or an affine combination of $(\boldsymbol{y}_2 - \boldsymbol{y}_1)/\|\boldsymbol{y}_2 - \boldsymbol{y}_1\|_2$ and $(\boldsymbol{y}_p - \boldsymbol{y}_1)/\|\boldsymbol{y}_p - \boldsymbol{y}_1\|_2$. To bias the solution of (4.3) to the one that corresponds to the closest neighbors of $\boldsymbol{y}_i$ in $\mathcal{M}_l$, we set up an optimization program whose objective function penalizes points based on their proximity to $\boldsymbol{y}_i$. That is, a point $\boldsymbol{y}_j$ that is closer to $\boldsymbol{y}_i$ gets a lower penalty, characterized by a weight $q_{ij} > 0$, than points that are farther away. Let $\boldsymbol{Q}_i$ be a diagonal matrix whose j-th diagonal entry is q_{ij}. We consider the following weighted ℓ_1-optimization program

$$\min \|\boldsymbol{Q}_i\boldsymbol{c}_i\|_1 \quad \text{subject to} \quad \|\boldsymbol{Y}_{-i}\boldsymbol{c}_i\|_2 \leq \epsilon, \quad \mathbf{1}^\top \boldsymbol{c}_i = 1, \tag{4.4}$$

where the ℓ_1-norm promotes sparsity of the solution [29] and the *proximity inducing matrix* $\boldsymbol{Q}_i$ favors selecting points that are close to $\boldsymbol{y}_i$.

One choice of the proximity inducing matrix is to select the q_{ij} to be

$$q_{ij} = \frac{\|\boldsymbol{y}_j - \boldsymbol{y}_i\|_2^\alpha}{\sum_{t \neq i} \|\boldsymbol{y}_t - \boldsymbol{y}_i\|_2^\alpha} \in (0, 1], \tag{4.5}$$

for $\alpha > 0$. One can also use other types of weights, such as exponential weights, given by

$$q_{ij} = \frac{\exp(\|\boldsymbol{y}_j - \boldsymbol{y}_i\|_2/\sigma)}{\sum_{t \neq i} \exp(\|\boldsymbol{y}_t - \boldsymbol{y}_i\|_2/\sigma)} \in (0, 1], \tag{4.6}$$

where $\sigma > 0$. The role of α in (4.5) and σ in (4.6) is to better deal with different manifold curvatures and adjust the size of the effective neighborhood from which we select the neighbors. More specifically, for a manifold that has a large curvature, the selected neighbors should be closer to the given point, hence requiring to have larger α or σ. On the other hand, for a manifold that has a small curvature, e.g., is close to a subspace, one can select the neighbors that are not necessarily the closest to the given point, yet are sufficiently close to it and lie on the same manifold. This is acheved by a smaller value of α or σ.

Another optimization program, which is related to (4.4) by the method of Lagrange multipliers, is

$$\min \lambda \|\boldsymbol{Q}_i \boldsymbol{c}_i\|_1 + \frac{1}{2}\|\boldsymbol{Y}_i \boldsymbol{c}_i\|_2^2 \quad \text{subject to} \quad \mathbf{1}^\top \boldsymbol{c}_i = 1, \tag{4.7}$$

where the parameter λ sets the trade-off between the sparsity of the solution and the affine reconstruction error [29, 99]. As we will show, for a wide range of values of λ, the optimization program in (4.7) successfully finds a sparse solution of each point whose nonzero elements correspond to the neighbors in the same manifold. Notice that, by the definition of $\boldsymbol{Q}_i$ and $\boldsymbol{Y}_i$, the solutions of the optimization programs (4.4) and (4.7) are invariant with respect to a global rotation, translation, and scaling of the data points.

Remark 7 *Although, for each data point, we consider all $N-1$ other points, from which we select a few, the points that are far from the given data point will not be selected by the weighted sparse optimization program. Thus, we can consider only the $L < N-1$ nearest neighbors of each data point in the proposed optimization program, which in turn improves the computational time of the algorithm.*

The proposed optimization programs can also be viewed as a modification of the SSC algorithm, studied in the previous chapter for clustering of data in multiple subspaces, to deal with nonlinear manifolds with the goal of obtaining the clustering and embedding of the data at the same time.

Notice that, in sharp contrast to the nearest neighbors-based methods, which first fix the number of neighbors or the neighborhood radius and then compute the weights between points in each neighborhood, we do the two steps at the same time. In other words, the optimization programs (4.4) and (4.7) *automatically* choose a few neighbors of the given data point, which approximately span a low-dimensional affine subspace at that point. As we will show in the experiments, this helps to effectively deal with manifolds that are spatially close to each other as well as manifolds of different intrinsic dimensions and with non-uniform sampling.

4.2.2 Clustering and dimensionality reduction

By solving the proposed optimization programs for each data point, we obtain the necessary information for clustering and dimensionality reduction. This is because

the solution $\boldsymbol{c}_i^\top \triangleq \begin{bmatrix} c_{i1} & \cdots & c_{iN} \end{bmatrix}$ of the proposed optimization programs satisfies

$$\sum_{j \neq i} \frac{c_{ij}}{\|\boldsymbol{y}_j - \boldsymbol{y}_i\|_2} (\boldsymbol{y}_j - \boldsymbol{y}_i) \approx \boldsymbol{0}. \tag{4.8}$$

Hence, we can rewrite $\boldsymbol{y}_i \approx \begin{bmatrix} \boldsymbol{y}_1 & \boldsymbol{y}_2 & \cdots & \boldsymbol{y}_N \end{bmatrix} \boldsymbol{w}_i$, where the weight vector $\boldsymbol{w}_i^\top \triangleq \begin{bmatrix} w_{i1} & \cdots & w_{iN} \end{bmatrix} \in \mathbb{R}^N$ associated to the i-th data point is defined as

$$w_{ii} \triangleq 0, \quad w_{ij} \triangleq \frac{c_{ij}/\|\boldsymbol{y}_j - \boldsymbol{y}_i\|_2}{\sum_{t \neq i} c_{it}/\|\boldsymbol{y}_t - \boldsymbol{y}_i\|_2}, \; j \neq i. \tag{4.9}$$

The indices of the few nonzero elements of $\boldsymbol{w}_i$, ideally, correspond to the neighbors of $\boldsymbol{y}_i$ in the same manifold and their values are proportional to their inverse distances to $\boldsymbol{y}_i$.

Next, we use the weights $\boldsymbol{w}_i$ to perform clustering and dimensionality reduction. We do so by building a similarity graph $\mathcal{G} = (V, E)$ whose nodes represent the data points. We connect each node i, corresponding to $\boldsymbol{y}_i$, to the node j, corresponding to $\boldsymbol{y}_j$, with an edge whose weight is equal to $|w_{ij}| + |w_{ji}|$. While, potentially, every node can get connected to all other nodes, because of the sparsity of $\boldsymbol{w}_i$, each node i connects itself to only a few other nodes that correspond to the neighbors of $\boldsymbol{y}_i$ in the same manifold. We call such neighbors as *sparse neighbors*. In addition, the distances of the sparse neighbors to $\boldsymbol{y}_i$ are reflected in the weights $|w_{ij}| + |w_{ji}|$.

The similarity graph built in this way has ideally several connected components, where points in the same manifold are connected to each other and there is no connection between two points in different manifolds. In other words, the similarity matrix

of the graph is ideally of the form

$$\boldsymbol{W} \triangleq \Big[|\boldsymbol{w}_1| \cdots |\boldsymbol{w}_N| \Big] + \Big[|\boldsymbol{w}_1| \cdots |\boldsymbol{w}_N| \Big]^\top = \begin{bmatrix} \boldsymbol{W}[1] & \cdots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{0} & \cdots & \boldsymbol{W}[n] \end{bmatrix} \boldsymbol{\Gamma}, \tag{4.10}$$

where $\boldsymbol{W}[\ell]$ is the similarity matrix of the data points in $\mathcal{M}_\ell$ and $\boldsymbol{\Gamma} \in \mathbb{R}^{N \times N}$ is an unknown permutation matrix. Clustering of the data follows by applying the spectral clustering algorithm of [84] to $\boldsymbol{W}$, similar to what we discussed in Section 3.2.2. One can also determine the number of connected components, i.e., the number of manifolds, by analyzing the eigenspectrum of the Laplacian matrix [113].

Any of the existing dimensionality reduction techniques can be applied to the data in each cluster to obtain a low-dimensional representation of the data in the corresponding manifold. However, this would require new computation of the neighborhoods and the weights. On the other hand, the similarity graph built by our method has a locality preserving property by the definition of the weights. Thus, we can use the adjacency matrix, $\boldsymbol{W}[i]$, of the i-th cluster as a similarity between points in the corresponding manifold and obtain a low-dimensional embedding of the data by taking the last few eigenvectors of the normalized Laplacian matrix associated to $\boldsymbol{W}[i]$ [7]. More precisely, we form the normalized Laplacian matrix associated to $\boldsymbol{W}[i] \in \mathbb{R}^{N_i \times N_i}$ as

$$\boldsymbol{L}[i] = \boldsymbol{I} - \boldsymbol{D}[i]^{-1} \boldsymbol{W}[i], \tag{4.11}$$

where $\boldsymbol{D}[i] = \text{diag}(\boldsymbol{W}[i]\boldsymbol{1})$. Collecting the d_i eigenvectors of $\boldsymbol{L}[i]$ corresponding to

its second to the $(d_i + 1)$-th smallest eigenvalues in a matrix $\boldsymbol{V}[i] \in \mathbb{R}^{N_i \times d_i}$, the d_i-dimensional representation of the N_i data points are then given by the rows of $\boldsymbol{V}[i]$.

4.2.3 Advantages of SMCE

An advantage of the proposed sparse optimization algorithm is that it can deal with manifolds of different intrinsic dimensions without requiring to know the dimensions of the manifolds a priori. Moreover, it can provide estimates of the intrinsic dimensions of the manifolds. These come from the fact that a data point $\boldsymbol{y}_i \in \mathcal{M}_\ell$ and its neighbors in $\mathcal{M}_\ell$, under appropriate sampling conditions, lie approximately in the d_ℓ-dimensional tangent space of $\mathcal{M}_\ell$ at $\boldsymbol{y}_i$. Since $d_\ell + 1$ vectors in this tangent space are linearly dependent, the solution $\boldsymbol{c}_i$ of the proposed optimization program in (4.7), with an appropriate choice of the regularization parameter, λ, is expected to have $d_\ell + 1$ nonzero elements. In practice, there is a range of λ for which the number of neighbors recovered by (4.7) reflects the dimensionality of the underlying manifold, as we will also show in the experiments.

To obtain an estimate of the dimension of a manifold, let Ω_ℓ denote the set of the indices of data points that belong to the ℓ-th cluster. For each data point in Ω_ℓ, sort the elements of $|\boldsymbol{c}_i|$ from the largest to the smallest values and denote the new vector as $\tilde{\boldsymbol{c}}_i$. We define the *median sparse coefficient* vector associated to the ℓ-th cluster as

$$\boldsymbol{msc}^{(\ell)} = \text{median}\{\tilde{\boldsymbol{c}}_i\}_{i \in \Omega_\ell}, \tag{4.12}$$

whose j-th element is computed as the median of the j-th elements of the vectors $\{\tilde{\boldsymbol{c}}_i\}_{i\in\Omega_\ell}$. Thus, the number of nonzero elements of $\boldsymbol{msc}^{(\ell)}$ or, more practically, the number of elements with relatively large magnitudes, gives an estimate of the intrinsic dimension of the ℓ-th manifold plus one.[1]

An advantage of our method is that it does not require to know the manifold dimensions a priori and allows to have a different neighborhood size for each data point, depending on the local dimension of the underlying manifold at that point. For example, in the case of two manifolds of dimensions $d_1 = 2$ and $d_2 = 30$, for data points in the ℓ-th manifold, we automatically obtain solutions with approximately $d_\ell + 1$ nonzero elements. On the other hand, conventional manifold clustering and dimensionality reduction methods that fix the number of neighbors fall into trouble, as we will also show in the experiments, because the number of neighbors would be too small for one manifold or too large for the other manifold.

4.3 Experiments

In this section, we evaluate the performance of SMCE on a number of synthetic and real experiments. For all the experiments, we use the optimization program (4.7).

As discussed in Remark 7, since the weighted ℓ_1-optimization does not select points that are very far from the given point, one consider only $L < N-1$ nearest neighbors

[1] One can also use the mean of the sorted coefficients in each cluster to compute the dimension of each manifold. However, we prefer to use the median for robustness reasons.

of each data point in the optimization program. In our experiments, we observed that we often obtain the same results using all data points and using $L = N/10$ nearest neighbors. In our experiments, we use the weights defined in (4.5), and in all of the experiments, except the motion segmentation experiment in Section 4.3.2.3, we set $\alpha = 1$. As in the case of nearest neighbors-based methods, there is no guarantee that the points in the same manifold form a single connected component of the similarity graph built by SMCE. However, this has always been the case in our experiments, as we will show next.

4.3.1 Experiments with synthetic data

Manifold embedding. We first evaluate SMCE for the dimensionality reduction task only. We sample $N = 1,000$ data points from a 2-sphere, where a neighborhood of its north pole is excluded. We embed the data in $\mathbb{R}^{100}$, add small Gaussian white noise to it and apply SMCE for $\lambda \in \{10, 50, 100, 200\}$. Figure 4.3 (top row) shows the embedding results of SMCE in a 2-dimensional Euclidean space. The three nonzero elements of the $\boldsymbol{msc}$ vector for $\lambda \in \{50, 100, 200\}$ correctly reflect the fact that the intrinsic dimension of the sphere is equal to two. However, note that for very large values of λ the embedding quality degrades since we put more emphasis on the sparsity of the solution.

The middle and bottom rows of Figure 4.3 show the embeddings obtained by several state-of-the-art algorithms for $K = 5$ and $K = 20$ nearest neighbors, respec-

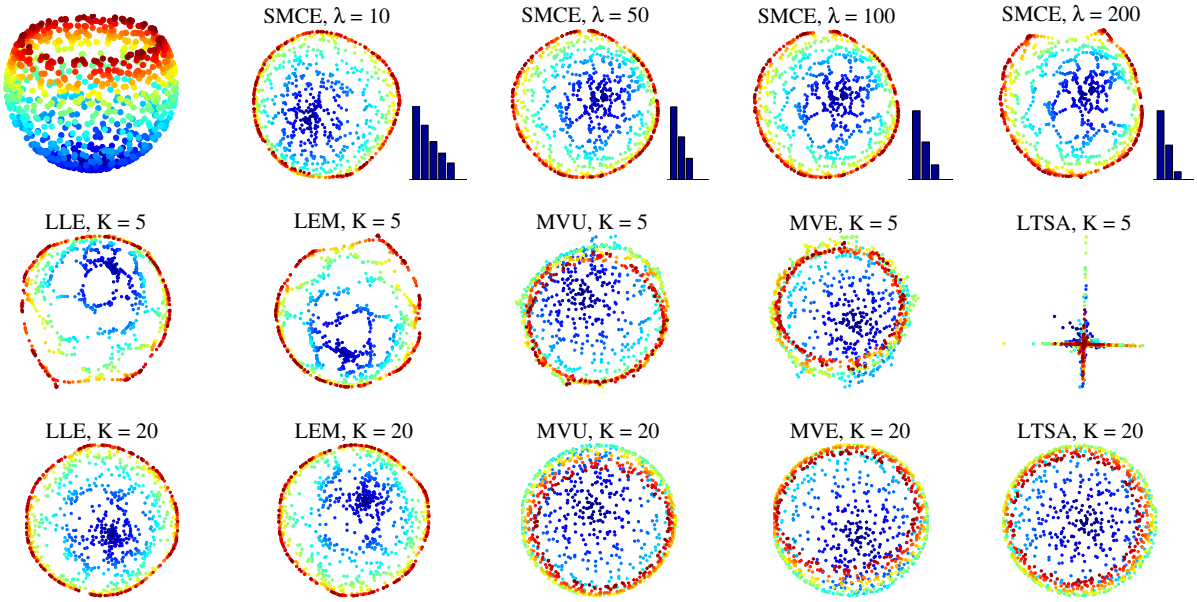


Figure 4.3: Top row: A punctured sphere embedded in $\mathbb{R}^{100}$, the 2-D embeddings and the $\boldsymbol{msc}$ vectors obtained by SMCE for different values of λ. Middle and bottom rows: embeddings obtained by the state-of-the-art nearest neighbor-based algorithms for $K = 5$ and $K = 20$, respectively.

tively. Notice that for $K = 5$, all methods obtain poor embedding results. On the other hand, for $K = 20$, LLE and LEM obtain embeddings similar to those of SMCE, while MVU, MVE and LTSA still do not obtain a good embedding (the red area near the north pole of the sphere gets embedded inside the disc instead of outside it). This suggests that the principle employed by SMCE to select the a few neighbors whose affine span passes near the given point is very effective, i.e., SMCE chooses a few neighbors that are very informative for dimensionality reduction.

Manifold clustering and embedding: non-uniform sampling. Next, we consider a 1-dimensional trefoil-knot, which is isometric to a circle in $\mathbb{R}^2$, and a two-dimensional flat manifold, which is isometric to a plane in $\mathbb{R}^2$, embedded jointly in $\mathbb{R}^{100}$

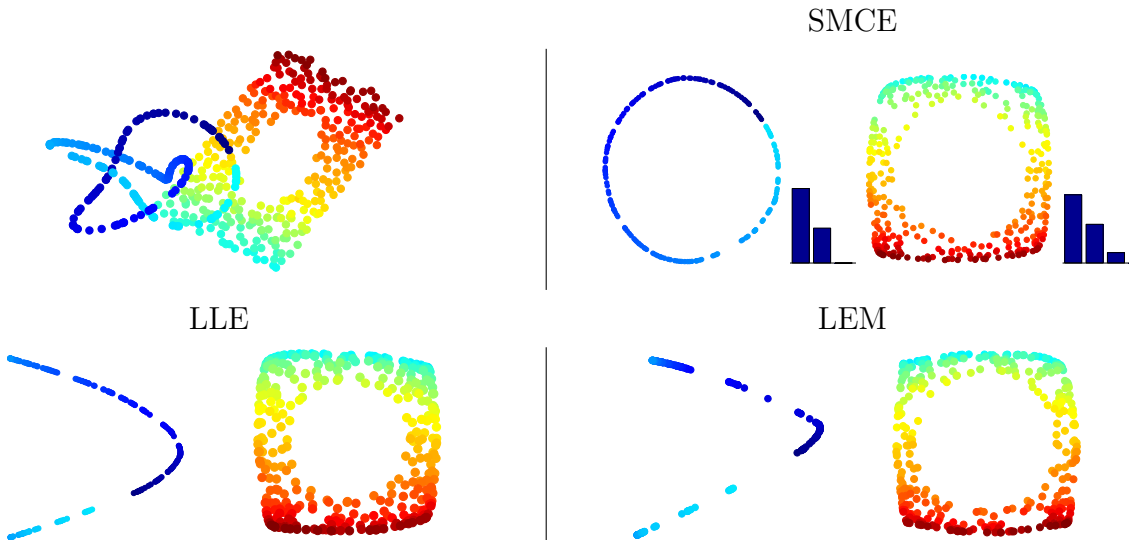


Figure 4.4: Clustering and embedding for a trefoil-knot with non-uniform sampling and a plane with a hole. Left: original manifolds. Middle: embedding and ***msc*** vectors obtained using SMCE. Right: clustering and embedding using LLE.

and corrupted with small Gaussian white noise. Both manifolds are randomly sampled and the data corresponding to a disk in the middle of the plane are then removed (see Figure 4.4). The non-convexity and non-uniform sampling of the parameter space causes some dimensionality reduction methods such as Isomap to warp the embedding [28]. In addition, it makes the choice of the number of nearest neighbors or the neighborhood radius challenging for the conventional dimensionality reduction algorithms. More precisely, for each manifold we need an appropriate neighborhood size that captures the local geometry of the manifolds and does not include points from other manifolds. In this example, K-nearest neighbor-based methods succeed in clustering only for $K \leq 7$ because there are points whose 8-th nearest neighbor comes from the other manifold. Figure 4.4 shows the results of SMCE with $\lambda = 10$ as well as the results of LLE and LEM with $K = 7$. While LLE and LEM with 7 nearest

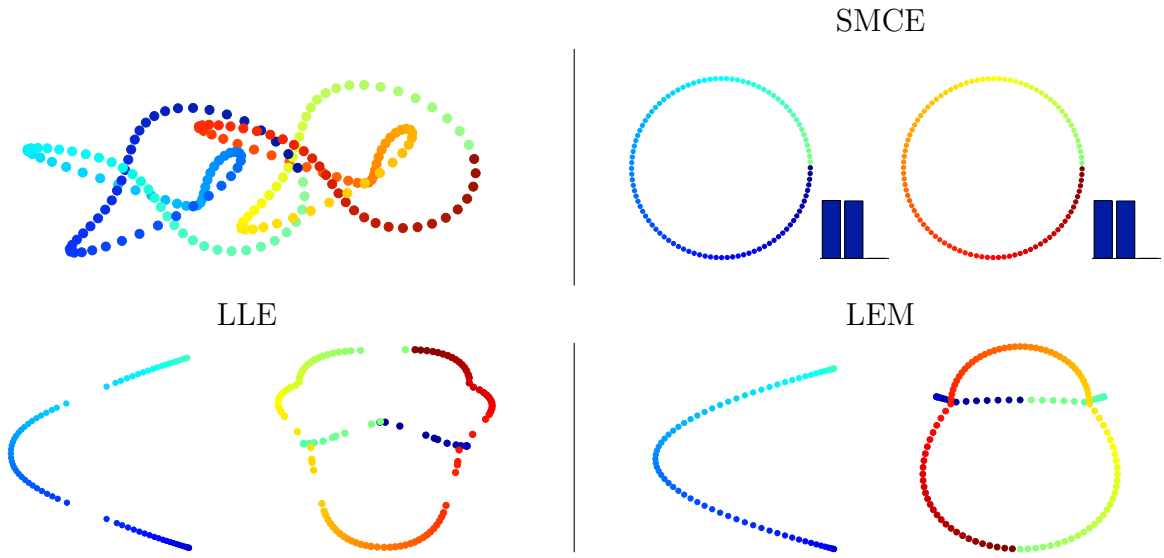


Figure 4.5: Top row: two trefoil-knots embedded in $\mathbb{R}^{100}$ and the clustering, embeddings and $\boldsymbol{msc}$ vectors obtained by SMCE. Bottom row: clustering and embeddings obtained by LLE and LEM.

Table 4.1: Clustering errors (%) of LLE and LEM as a function of K and of SMCE as a function of λ, for the example of the two trefoil-knots shown in Figure 4.5.

K	2	3	4	5	6	8	10	20
LLE Error	15.5	9.5	16.5	13.5	16.5	37.5	38.5	14.5
LEM Error	15.5	13.5	17.5	14.5	24.5	27.5	13.5	12.5
λ	0.2	2	20	50	80	100	200	400
SMCE Error	15.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

neighbors does not find a faithful embedding of the trefoil-knot, SMCE with 2 or 3 neighbors succeeds in clustering, embedding, and estimating the intrinsic dimensions of the manifolds, which suggests that SMCE chooses more informative neighbors for both clustering and embedding.

Manifold clustering and embedding: spatial proximity. Next, we consider the challenging case of manifolds being close to each other. We consider two trefoil-knots, shown in Figure 4.5, which are embedded in $\mathbb{R}^{100}$ and are corrupted with small

Gaussian white noise. The data points are sampled such that among the 2 nearest neighbors of 1% of the data points there are points from the other manifold. Also, among the 3 and 5 nearest neighbors of 9% and 18% of the data points, respectively, there are points from the other manifold. The nearest neighbor-based methods will connect such points to their neighbors in the other manifold and assign large weights to the connections. As a result, these methods cannot obtain a successful clustering or a proper embedding. Table 4.1 shows the clustering errors of LLE and LEM for different number of nearest neighbors, K, as well as the clustering errors of SMCE for different values of λ. Notice that, while there is no K such that LLE and LEM can successfully cluster the data, SMCE obtains a perfect clustering for a wide range of the values of λ. Figure 4.5 shows the results of SMCE for $\lambda = 10$ as well as LLE and LEM for $K = 3$. As the results show, enforcing that the sparse neighbors of a point span a low-dimensional affine subspace passing near the point helps to select neighbors from the correct manifold. This results in successful clustering and embedding of the data as well as unraveling the dimensions of the manifolds (SMCE obtains, in general, two sparse neighbors for each data point indicating the 1-dimensionality of the manifolds). On the other hand, the fact that LLE and LEM choose the wrong neighbors with strong weights, results in low-quality embeddings.

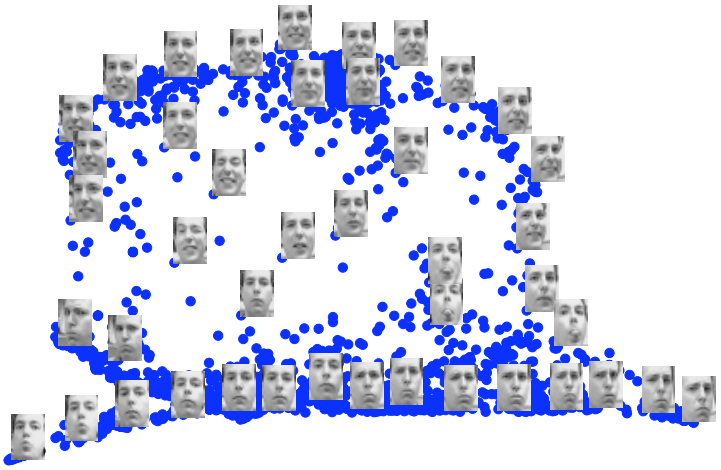


Figure 4.6: 2-D embedding of the Frey face dataset obtained by SMCE.

4.3.2 Experiments with real data

In this section, we evaluate the performance of SMCE on real datasets. We show that challenges such as manifold proximity and non-uniform sampling are also common in real datasets, and that our algorithm is able to handle these issues effectively.

4.3.2.1 Clustering and embedding of faces

Frey face dataset. We consider the dimensionality reduction of the images in the Frey face dataset, which consists of $1,965$ face images captured under varying pose and expression. Each image is vectorized as a 560 element vector of pixel intensities. Figure 4.6 shows the 2-dimensional embedding obtained by SMCE. Notice that the low-dimensional representation captures well the left to right pose variations in the horizontal axis and the expression changes in the vertical axis.

Extended YaleB face dataset. We consider now the problem of clustering and

Table 4.2: Percentage of face images in the two subjects of the Extended YaleB face database whose K nearest neighbors contain points from the other subject.

K	1	2	3	4	7	10
	3.9%	10.2%	23.4%	35.2%	57.0%	64.8%

embedding of face images in the Extended Yale B database [73].

First, we consider face images of two different subjects in the dataset. Each subject has 64 images of 192×168 pixels, captured under a fixed pose and expression and with varying illuminations, leading to almost $33,000$-dimensional vectorized faces. Notice that the space of the face images under varying illumination is not densely sampled and in addition the two manifolds are very close to each other. Table 4.2 shows the percentage of the points in the two manifolds (subjects) whose K nearest neighbors contain points from the other manifold (subject).

Figure 4.7 shows the clustering errors of LLE and LEM as a function of the number of nearest neighbors, K, and the clustering errors of SMCE as a function of the regularization parameter, λ. Notice that, for both LLE and LEM, there is only one value of K such that the clustering error is minimum (around 11%), and as we change K, the clustering errors increase. On the other hand, for SMCE, there is a large range of λ such that the clustering error is small (less than 4%).

Figure 4.8 (top row) shows the embeddings obtained by LLE, LEM and SMCE for all the data prior to clustering. Notice that only SMCE reasonably separates the low-dimensional representations of the face images according to their subjects. As the table shows, there are several points whose nearest neighbor comes from the other

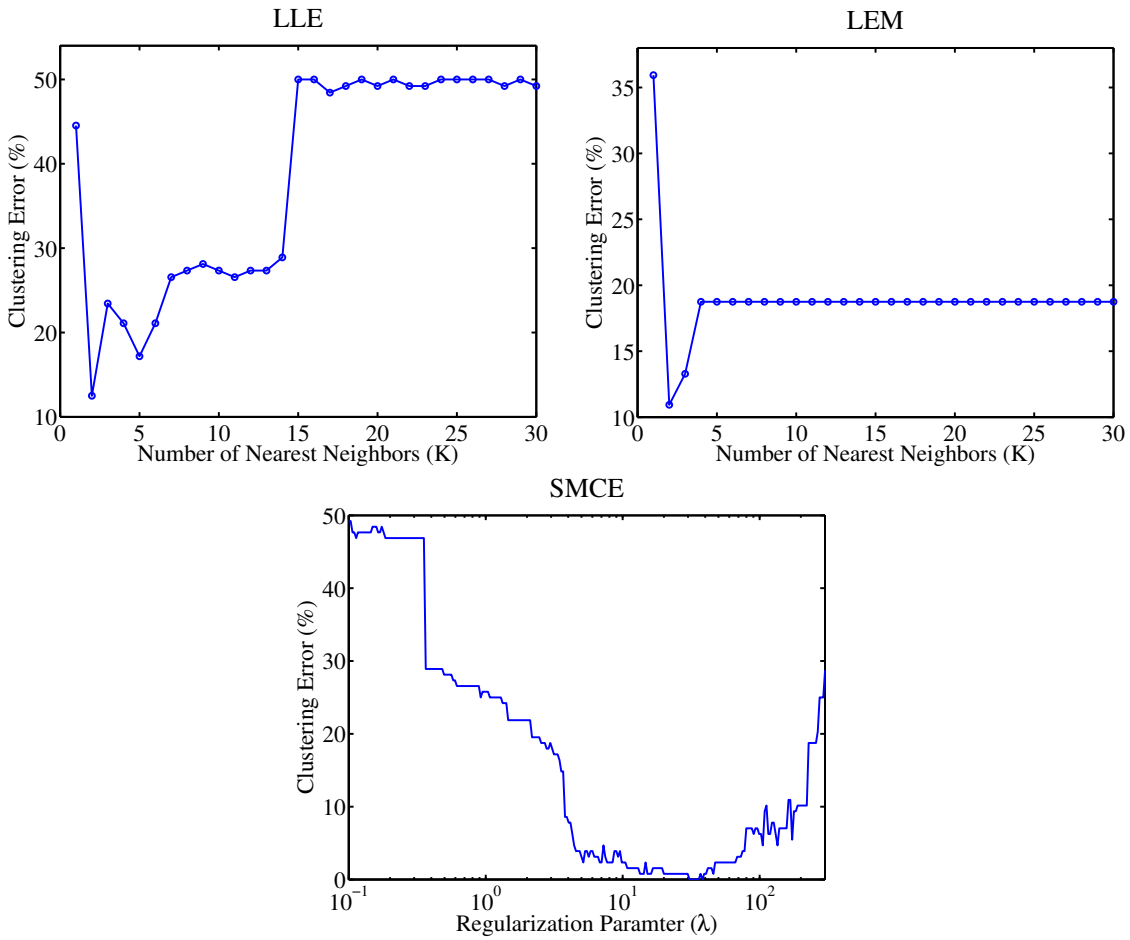


Figure 4.7: Top: clustering errors (%) of LLE and LEM for two subjects in the Extended YaleB dataset as a function of the number of nearest neighbors (K). Bottom: clustering error (%) of SMCE for two subjects in the Extended YaleB dataset as a function of the regularization parameter λ.

manifold. Below the embedding of each method in Figure 4.8 (bottom row), we have shown the weight vector associated to a data point in $\mathcal{M}_1$ whose nearest neighbor comes from $\mathcal{M}_2$. Notice that, while the nearest neighbor-based methods select wrong neighbors (neighbors from $\mathcal{M}_2$) with strong weights, SMCE selects a few neighbors from the correct manifold. The plots in Figure 4.9 show the embeddings obtained by SMCE for each cluster. Notice that, as we move along the horizontal axis, the

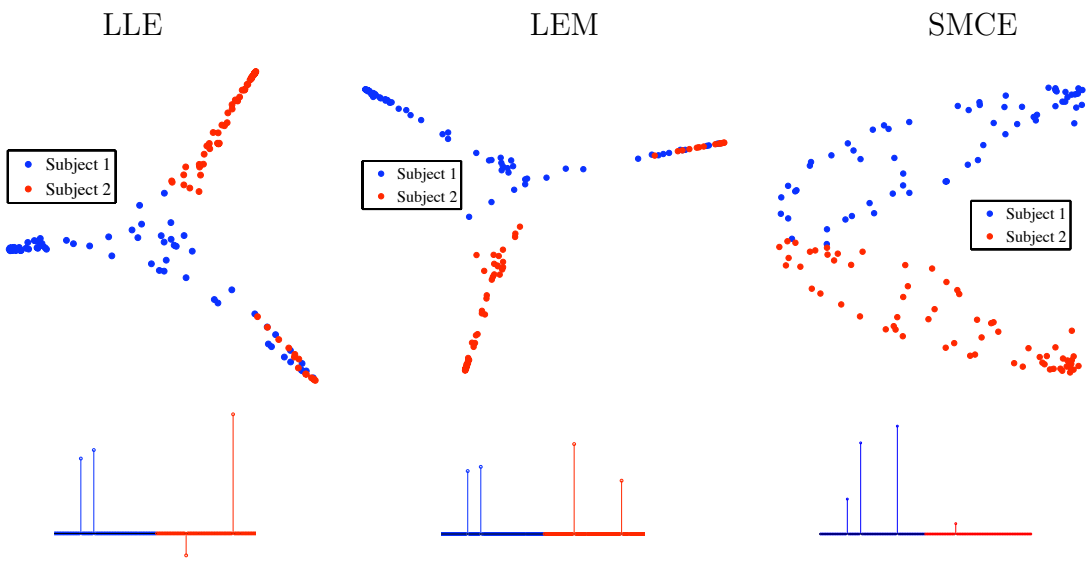


Figure 4.8: Top: 2-D embeddings obtained by LLE, LEM and SMCE for the face data of two subjects in the Extended YaleB dataset. Bottom: the weights associated to a data point from subject 1 obtained by each algorithm.

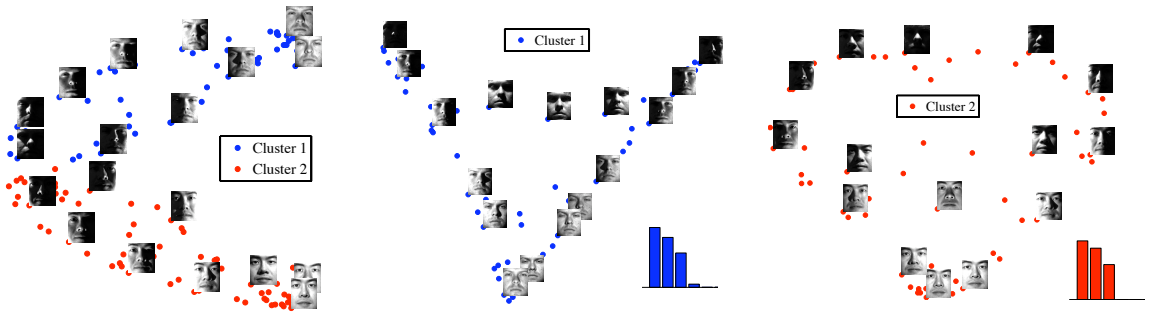


Figure 4.9: Clustering, embedding, and $\boldsymbol{msc}$ vectors obtained by SMCE for the face data of two subjects in the Extended YaleB dataset.

direction of the light source, for the images in each cluster, changes from left to right, while as we move along the vertical axis, the overall darkness of the images changes from light to dark. Also, the $\boldsymbol{msc}$ vectors suggest a 2-dimensionality of the face manifolds, correctly reflecting the number of the degrees of freedom of the light source on the illumination rig (a sphere in $\mathbb{R}^3$).

Table 4.3: Clustering errors (%) of LLE, LEM and SMCE on the Extended YaleB dataset as a function of the number of subjects (clusters).

Algorithm	LLE	LEM	SMCE
2 Subjects			
Mean	25.39	26.83	**13.78**
Median	21.88	24.22	**4.69**
3 Subjects			
Mean	38.39	41.05	**18.82**
Median	39.58	42.19	**12.76**
5 Subjects			
Mean	48.84	50.01	**21.52**
Median	50.00	49.38	**22.81**
8 Subjects			
Mean	55.57	56.56	**25.46**
Median	55.27	56.84	**26.86**
10 Subjects			
Mean	57.58	59.58	**27.14**
Median	57.58	58.59	**25.62**

Next, we consider clustering of face images on the entire Extended YaleB dataset, where we use the same settings as the one described in Section 3.8.5. Table 4.3 shows the clustering errors of LLE, LEM and SMCE on the dataset as a function of the number of subjects (clusters). The results of LLE and LEM correspond to using $K = 3$ and the results of SMCE correspond to using $\lambda = 50$, which give the lowest clustering errors. Notice from the results that SMCE significantly outperforms LLE and LEM. More specifically, SMCE obtains half of the clustering errors of LLE and LEM for all different number of subjects.

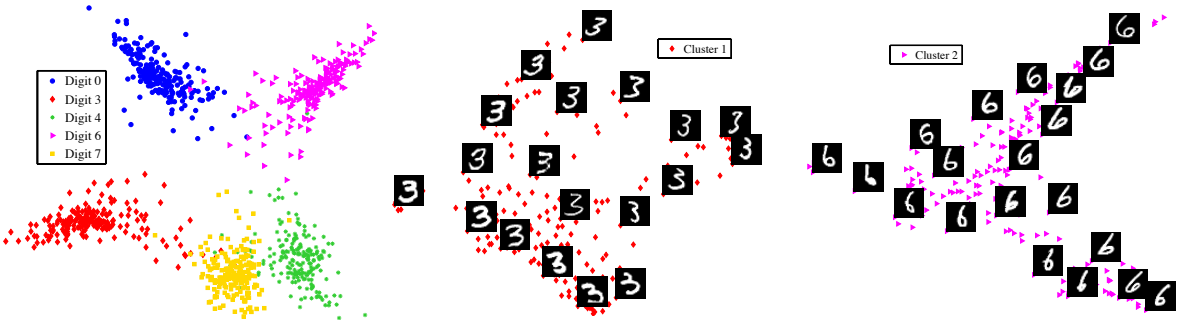


Figure 4.10: Clustering and embedding of five digits from the MNIST dataset using SMCE. Left: 2-D embedding of all the data points from digits $\{0, 3, 4, 6, 7\}$. Middle: 2-D embedding of the data points in the first cluster that corresponds to the digit 3. Right: 2-D embedding of the data points in the second cluster that corresponds to the digit 6.

4.3.2.2 Clustering and embedding of digits

We also consider the clustering and dimensionality reduction of the digits from the MNIST test dataset [72]. We use the images from five digits $\{0, 3, 4, 6, 7\}$ in the dataset where we randomly select 200 data points from each digit. The left plot in Figure 4.10 shows the joint embedding of all the data points obtained by SMCE. One can see that the data are well separated according to their classes.

The middle and the right plots in Figure 4.10, show the 2-dimensional embeddings obtained by SMCE for two clusters, which correspond to the digits 3 and 6. Notice that, for the embedding of the cluster corresponding to the digit 3, as we move on the horizontal axis from left to right, the overall thickness of the digits changes from thick to thin. On the other hand, as we move on the vertical axis from bottom to top, the digits change from being slanted to the left to being slanted to the right. Similarly, for the embedding of the cluster corresponding to the digit 6, as we move

on the horizontal axis from left to right, the digits change from being slanted to the left to being slanted to the right. On the other hand, as we move on the vertical axis from bottom to top, the size of the end loop of the digit changes from small to large.

4.3.2.3 Motion segmentation experiments

To evaluate the clustering performance of the SMCE algorithm on a large dataset, we consider the problem of motion segmentation on the Hopkins 155 dataset, which we also considered in the previous chapter for the subspace clustering problem.

First, we run the LLE and LEM algorithms for different numbers of the nearest neighbors, K. More specifically, for each value of K, we build the K-nearest neighbor graph and obtain the weights using LLE and LEM. Then, we use the obtained weights in a spectral clustering framework, as described in Chapter 2, to cluster the similarity graph into $n \in \{2, 3\}$ groups. Here, n denotes the number of motions in each video, which we assume is given to the algorithms. The top plots of Figure 4.11 show the average clustering errors for sequences of two and three motions in the dataset. From the results we make the following conclusions:

– The clustering errors of both algorithms are minimum for $K \in [5, 9]$, which is close to the dimension of each manifold, i.e., 3. This comes from the fact that typically for a d-dimensional manifold $K \approx 2d$ neighbors well capture the geometry of the manifold.

– The clustering errors of both algorithms are large for all values of K, compared

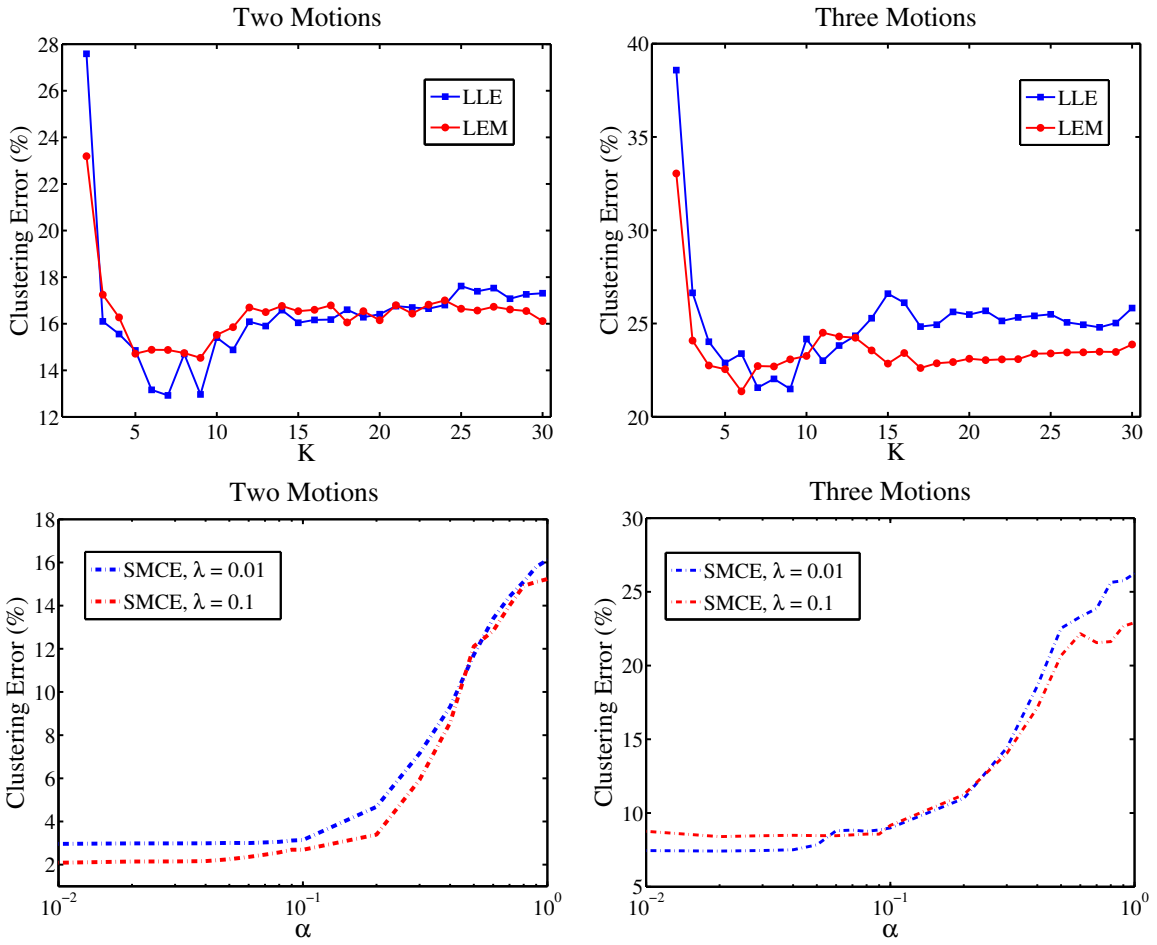


Figure 4.11: Top: clustering errors (%) of LLE and LEM algorithms on the Hopkins 155 dataset as a function of the number of the nearest neighbors, K, for two motions (left) and three motions (right). Bottom: clustering errors (%) of SMCE algorithm on the Hopkins 155 dataset as a function of the exponent, α, of the weights in (4.5), for two different values of λ, for two motions (left) and three motions (right).

to the state-of-the-art results that we studied in Chapter 3. This comes from the fact that, for small values of K, the neighborhood graph does not capture well the structure of the manifold, e.g., the points from the same manifold may form multiple components of the similarity graph. On the other hand, for large values of K, while the points from the same manifold get connected to each other, they also get connected

to points from other manifolds, hence spectral clustering fails to correctly separate the data points into their underlying manifolds.

Next, we run the SMCE algorithm for different values of the exponent, α, of the weights in (4.5) and for different values of the regularization parameter $\lambda \in \{0.01, 0.1\}$. The bottom plots of Figure 4.11 show the average clustering errors for sequences of two and three motions. From the results, we make the following conclusions:

– For a wide range of the weight exponent value, α, SMCE obtains small clustering errors. As α increases, the clustering error also increases. This comes from the fact that for large values of α, the algorithm becomes short-sighted, selecting few points that are very close to the given point. On the other hand, for smaller values of α, the algorithm allows for the selection of the points in the same manifold that are farther, due to manifold gap and non-uniform sampling, yet lie on a low-dimensional affine space passing close to the given point.

– The clustering errors for the two values of λ are relatively close to each other, corroborating the robustness of the algorithm to the choice of the regularization parameter.

Figure 4.12 shows the boxplots of the errors obtained by LLE, LEM and SMCE for two and three motions. Notice from the plots that SMCE obtains statistically significant improvement over LLE and LEM for the segmentation of motions. Specifically, for sequences with two motions, the length of the boxplot for SMCE is short, showing that the algorithm obtain small errors for the majority of the sequences. Table

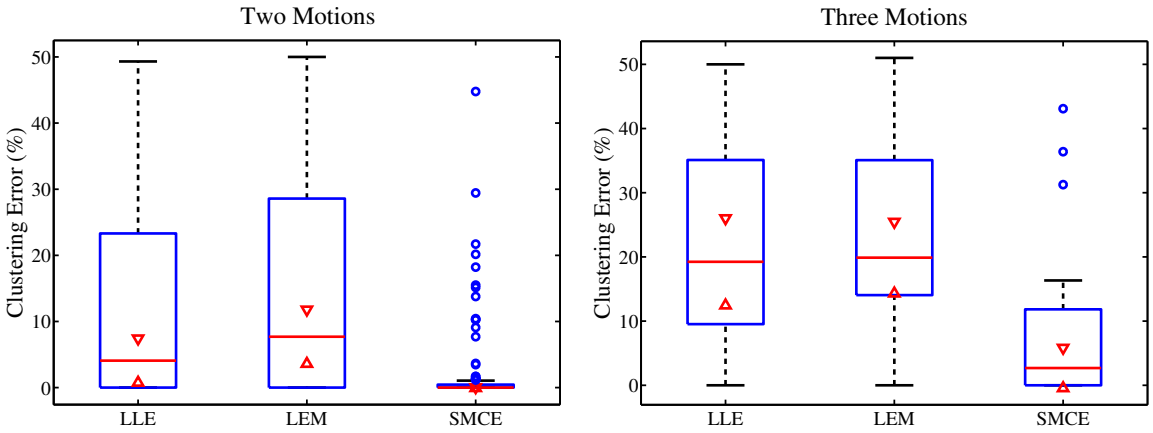


Figure 4.12: Boxplots for the motion segmentation errors of nonlinear manifold clustering algorithms on the Hopkins 155 dataset using $2F$-dimensional data points. Left: clustering errors (%) for two motions. Right: clustering errors (%) for three motions.

Table 4.4: Clustering errors (%) of linear and nonlinear manifold clustering algorithms on the Hopkins 155 dataset with the $2F$-dimensional data points.

Algorithms	LSA	SSC	LLE	LEM	SMCE
2 Motions					
Mean	4.23	**1.52**	12.92	14.75	2.15
Median	0.56	**0.00**	4.08	7.69	**0.00**
3 Motions					
Mean	7.02	**4.40**	21.56	23.56	7.03
Median	1.45	**0.56**	19.24	19.89	2.69
All					
Mean	4.86	**2.18**	14.87	16.74	3.25
Median	0.89	**0.00**	8.71	13.97	**0.00**

4.4 also shows the average and the median clustering errors of different algorithms. Note that SMCE obtains larger clustering error than SSC, studied in the previous chapter, while it achieves much smaller errors than other manifold clustering as well as subspace clustering methods, studied in Chapter 3. This comes from the fact that SMCE at the same time of clustering pursues a dimensionality reduction objective, i.e., it looks for points that are sufficiently close to the given point.

4.4 Conclusions

We proposed an algorithm based on sparse representation for simultaneous clustering and dimensionality reduction of data lying in multiple manifolds. We used the solution of a sparse optimization program to build a similarity graph from which we obtained clustering and low-dimensional embedding of the data. The sparse representation of each data point ideally encodes information that can be used for inferring the dimensionality of the underlying manifold around that point. We demonstrated the effectiveness of the proposed algorithm for dealing with spatially close manifolds and manifolds with non-uniform sampling and holes on synthetic and real data.

Chapter 5

Classification of Multi-Manifold Data via Block-Sparse Recovery

In Chapters 3 and 4, we studied the problem of clustering data that lie in a union of subspaces and nonlinear manifolds, respectively. A closely related and important problem in the case of multi-manifold data, is the classification problem. The main difference is that, in the clustering problem, the separation of the given data into their underlying manifolds is unknown and needs to be recovered, while in the classification problem, the data (training samples) are separated according to their underlying manifolds, and the goal is to find the manifold of a new data point (test sample).

In this chapter, we consider the problem of classification of multi-manifold data, where each class corresponds to a different manifold, and we want to classify a given query and determine the manifold, i.e., the class, it belongs to. Motivated by practical

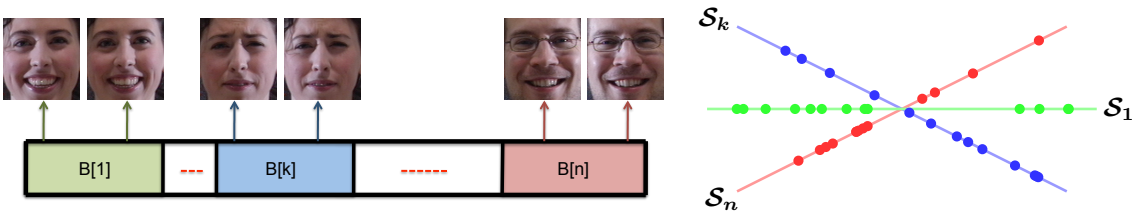


Figure 5.1: In the face recognition problem, the dictionary has a block structure where the training images of each subject form a few blocks of the dictionary and lie in a union of subspaces.

problems such as face recognition [6,73,117], we model the manifold of each class with a union of low-dimensional subspaces (see Figure 5.1) and assume that a test sample lies in the direct sum of a few subspaces from the same class. We exploit the fact that the dictionary of all training samples has a block structure where training data in each class form a few blocks of the dictionary, hence, a test sample can be represented by a combination of the training data from a few blocks of the dictionary [42].

We cast the classification as a block-sparse recovery problem where our goal is to find a representation of a test example that uses the minimum number of blocks from the dictionary. We consider two different classes of non-convex optimization programs, consider their convex relaxations, and study conditions under which the relaxations are equivalent to the original problems [42, 44]. In addition, we show that the optimization programs can be modified properly to also deal with corrupted data. We evaluate these algorithms on synthetic and real data, where we consider the problem of automatic face recognition. We show that casting the face recognition problem as a block-sparse recovery problem can improve the results of the state-of-the-art face recognition algorithms, especially when there are relatively a small

number of training data in each class.

5.1 Problem settings

We assume that we are given N training data that lie in a union of L nonlinear manifolds $\{\mathcal{M}_\ell\}_{\ell=1}^L$ corresponding to L classes. Motivated by practical problems such as face recognition [6, 73, 117], we model the manifold of each class with a union of subspaces $\{\mathcal{S}_i\}_{i=1}^n$ of dimensions $\{d_i\}_{i=1}^n$, where for all i we have $d_i \ll D$ (see Figure 5.1). In other words, there exists a partition $\{\Gamma_\ell\}_{\ell=1}^L$ of the set $\{1, 2, \cdots, n\}$ such that the elements in Γ_ℓ denote the indices of subspaces associated with $\mathcal{M}_\ell$. We denote by $\{\boldsymbol{b}_{ij} \in \mathbb{R}^D\}_{j=1}^{m_i}$ the training data in the i-th subspace an let

$$\boldsymbol{B}[i] \triangleq \begin{bmatrix} \boldsymbol{b}_{i1} & \boldsymbol{b}_{i2} & \cdots & \boldsymbol{b}_{im_i} \end{bmatrix} \in \mathbb{R}^{D \times m_i}. \tag{5.1}$$

We also denote by $\boldsymbol{B}$ the collection of all training data across all classes, i.e.,

$$\boldsymbol{B} \triangleq \begin{bmatrix} \boldsymbol{B}[1] & \boldsymbol{B}[2] & \cdots & \boldsymbol{B}[n] \end{bmatrix} \in \mathbb{R}^{D \times N}. \tag{5.2}$$

Given a test example $\boldsymbol{y} \in \mathbb{R}^D$, which belongs to one of the L classes, our goal is to find the class to which the test example belongs.

The classification algorithms, which we study in this chapter, can be thought of as generalizations of the representation-based classification (SRC) algorithm [117], where we assume a multi-manifold structure for the training data. In the next section, we review the SRC algorithm.

5.2 A review of sparse representation-based classification

The sparse representation-based classification (SRC) algorithm [117] is based on the assumption that a test sample has a sparse representation in the dictionary of all the training data across different classes, where it can be written as a linear combination of a few training samples from the correct class. Hence, one is interested in solving the following optimization problem

$$P_{\ell_0}: \quad \min \ \|\boldsymbol{c}\|_0 \quad \text{s.t.} \quad \boldsymbol{y} = \boldsymbol{B}\boldsymbol{c}, \tag{5.3}$$

where $\|\cdot\|_0$ denotes the ℓ_0 norm and indicates the number of nonzero elements of the given vector. Since the P_{ℓ_0} optimization program is in general NP-hard, a convex relaxation of it is considered by replacing the ℓ_0 with the ℓ_1 norm and solving the following convex program

$$P_{\ell_1}: \quad \min \ \|\boldsymbol{c}\|_1 \quad \text{s.t.} \quad \boldsymbol{y} = \boldsymbol{B}\boldsymbol{c}. \tag{5.4}$$

The SRC algorithm finds the class of a given test example as the class that best represents the test example using its training data. More precisely, for a given test example $\boldsymbol{y}$, if we denote by $\boldsymbol{c}^{*\top} = \begin{bmatrix} \boldsymbol{c}^{*\top}[1] & \dots & \boldsymbol{c}^{*\top}[n] \end{bmatrix}$ the optimal solution of P_{ℓ_1}, the class of $\boldsymbol{y}$ is obtained by

$$\text{class}(\boldsymbol{y}) = \text{argmin}_i \left\| \boldsymbol{y} - \sum_{j \in \Gamma_i} \boldsymbol{B}[j]\boldsymbol{c}^*[j] \right\|_2, \tag{5.5}$$

where, as mentioned before, Γ_i denotes the set of indices of the blocks corresponding to class i. In other words, it is assumed that the best reconstruction of the test sample is achieved using a few training samples from the same class.

An important advantage of the SRC algorithm is that it can deal with corrupted data within the same framework. To see this, let $\boldsymbol{y}_0$ be a test example corrupted with an error $\boldsymbol{e}$ that has a few nonzero entries, i.e., $\boldsymbol{y} = \boldsymbol{y}_0 + \boldsymbol{e}$, where $\|\boldsymbol{e}\|_0 \ll D$. Note that $\boldsymbol{y}_0$ has a sparse representation in the dictionary of the training data $\boldsymbol{B}$ and the error has a sparse representation in the standard basis $\boldsymbol{I}$ (the identity matrix in $\mathbb{R}^{D\times D}$). Thus, in a new dictionary formed by concatenating the training data and the standard basis, $\boldsymbol{y}$ has a sparse representation that can be recovered from

$$\bar{P}_{\ell_0}: \ \min \left\| \begin{bmatrix} \boldsymbol{c} \\ \\ \boldsymbol{e} \end{bmatrix} \right\|_0 \quad \text{s.t.} \quad \boldsymbol{y} = \begin{bmatrix} \boldsymbol{B} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{c} \\ \\ \boldsymbol{e} \end{bmatrix}. \tag{5.6}$$

To solve this problem, one uses an ℓ_1 relaxation and instead solve the following convex program

$$\bar{P}_{\ell_1}: \ \min \left\| \begin{bmatrix} \boldsymbol{c} \\ \\ \boldsymbol{e} \end{bmatrix} \right\|_1 \quad \text{s.t.} \quad \boldsymbol{y} = \begin{bmatrix} \boldsymbol{B} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{c} \\ \\ \boldsymbol{e} \end{bmatrix}. \tag{5.7}$$

Finally, the class of the corrupted test sample is given by

$$\text{class}(\boldsymbol{y}) = \text{argmin}_i \left\| \boldsymbol{y} - \boldsymbol{e}^* - \sum_{j\in\Gamma_i} \boldsymbol{B}[j]\boldsymbol{c}^*[j] \right\|_2. \tag{5.8}$$

5.3 Challenges of multi-manifold data classification

While SRC algorithm has been shown to be effective for classification, there still remain questions about classification in the multi-manifold setting using sparse representation which have not been sufficiently explored or have not been answered yet.

C1– The SRC method looks for the sparsest representation of a test example with the hope that such a representation selects few training data from the correct class. However, as shown in Figure 5.1, the dictionary of the training data has a structure in which the manifold of each class is modeled by a union of low-dimensional subspaces. *Is there a way to direct the SRC method to take into account the dictionary structure, e.g., by finding a representation of a test example that involves only a few blocks of the dictionary corresponding to the training data from a single class.* If so, *what would be the behavior of the new algorithms in dealing with corrupted data?*

C2– When it comes to the problem of classification in multiple subspaces, there is a fundamental gap between the theory of sparse recovery and the practice of machine learning.

C2a– When the number of training data in each class is large, we can better capture the underlying distribution of the data and the classification performance increases. Nonetheless, existing sparse recovery algorithms do not have theoretical

guarantees when it comes to highly redundant dictionaries and the conditions for their success almost never hold. *Can we fill the gap between the current sparse representation theory and the classification practice?*

C2b– When the number of training data in each class is small, sparse recovery methods have good theoretical guarantees. However, classification algorithms do not perform well. *Can we have alternative methods based on sparse representation that can lead to better classification results when the number of training data in each class is small?*

5.4 Classification via block-sparse representation

In this thesis, we assume that the data in each class lie in a manifold, which can be modeled as a union of low-dimensional subspace. We argue that looking for the sparsest representation of a test example is not the best criterion for classification.

In order to see this, we consider the example in Figure 5.2 (left) where we have 3 classes whose training samples lie in three subspaces; $\mathcal{S}_1$ being a 2-dimensional subspace, $\mathcal{S}_2$ and $\mathcal{S}_3$ being 1-dimensional subspaces. The test sample $\boldsymbol{y}$, which belongs to class 1, can be written as a linear combination of any two data points from class 1, while it can also be written as a linear combination of one data point from class 2 and one from class 3. Thus, from the sparsest representation perspective, there is no

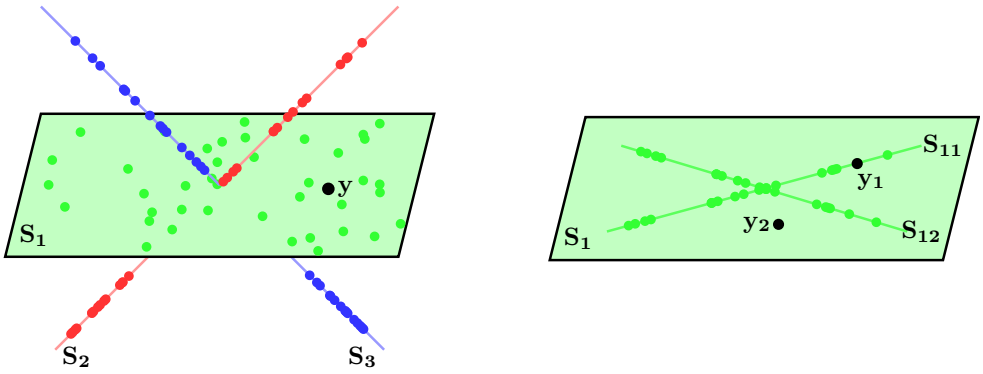


Figure 5.2: Left: sparsest representation of a test example does not necessarily come from the correct class. $\boldsymbol{y}$ can be written as a linear combination of one data point from $\mathcal{S}_2$ and one from $\mathcal{S}_3$ as well as a linear combination of two data points from $\mathcal{S}_1$. Right: training data in a class might be separated into several blocks. Thus, a test example can be written as a linear combination of a few blocks in each class.

difference between the two representations as they both have two nonzero elements, while obviously from a classification perspective, the first solution is the desired one. Now, if instead of looking for the sparsest representation we look for a representation that uses the minimum number of subspaces/blocks, we obtain the desired solution for perfect classification.

In a general classification task, the dictionary of the training samples has a block structure with several blocks for each class corresponding to the training samples from that class lying in a union of low-dimensional subspaces. We assume that a test sample can be represented as a linear combination of the training samples from a few blocks of the dictionary corresponding to its underlying class. For example, in Figure 5.2 (right), the test example $\boldsymbol{y}_1$ can be written as a linear combination of 1 block while $\boldsymbol{y}_2$ can be written as a linear combination of two blocks of the underlying class. As another example, in the face recognition problem, each class consists of images of a

single subject that can be separated into multiple blocks based on different expressions as shown in Figure 5.1.

5.4.1 Block-sparse representation via P_{ℓ_q/ℓ_0}

Based on what we have discussed so far, a better objective for classification is to solve

$$P_{\ell_q/\ell_0}: \ \min \sum_{i=1}^{n} \mathrm{I}(\|\boldsymbol{c}[i]\|_q > 0) \quad \text{s.t.} \quad \boldsymbol{y} = \boldsymbol{B}\boldsymbol{c}, \tag{5.9}$$

where $\mathrm{I}(\cdot)$ is the indicator function and $q > 0$. This optimization problem seeks the minimum number of nonzero coefficient blocks that reconstruct the test example. Note that the optimization program P_{ℓ_q/ℓ_0} is, in general, NP-hard since it requires searching exhaustively over all possible few blocks of $\boldsymbol{B}$ and checking whether they span the given $\boldsymbol{y}$. An ℓ_1 relaxation of this program is given by

$$P_{\ell_q/\ell_1}: \ \min \sum_{i=1}^{n} \|\boldsymbol{c}[i]\|_q \quad \text{s.t.} \quad \boldsymbol{y} = \boldsymbol{B}\boldsymbol{c}, \tag{5.10}$$

which is a convex program when $q \geq 1$.

Remark 8 *For $q = 1$, while the non-convex programs P_{ℓ_0/ℓ_1} and P_{ℓ_0} are different, their convex relaxations P_{ℓ_1/ℓ_1} and P_{ℓ_1} are the same. Thus, P_{ℓ_1} can also be thought of as a block-sparse recovery method that under appropriate conditions, as will be discussed in this chapter, finds a representation of the test example with the minimum number of nonzero blocks.*

5.4.2 Block-sparse representation via P'_{ℓ_q/ℓ_0}

We will also consider an alternative optimization program for the classification problem, which can be formulated as

$$P'_{\ell_q/\ell_0}: \ \min \sum_{i=1}^{n} \mathrm{I}(\|\boldsymbol{B}[i]\boldsymbol{c}[i]\|_q > 0) \quad \text{s.t.} \quad \boldsymbol{y} = \boldsymbol{B}\boldsymbol{c}, \tag{5.11}$$

where $q > 0$. The ℓ_1 relaxation of this optimization program is given by

$$P'_{\ell_q/\ell_1}: \ \min \sum_{i=1}^{n} \|\boldsymbol{B}[i]\boldsymbol{c}[i]\|_q \quad \text{s.t.} \quad \boldsymbol{y} = \boldsymbol{B}\boldsymbol{c}, \tag{5.12}$$

which is a convex program for $q \geq 1$. Unlike P_{ℓ_q/ℓ_0} that minimizes the number of nonzero coefficient blocks $\boldsymbol{c}[i]$, the optimization program P'_{ℓ_q/ℓ_0} minimizes the number of nonzero reconstructed vectors $\boldsymbol{B}[i]\boldsymbol{c}[i] \in \mathcal{S}_i$, i.e., minimizes the number of active subspaces. When the blocks consist of linearly independent data, the solution of P'_{ℓ_q/ℓ_0} has also the minimum number of nonzero coefficient blocks, because $\|\boldsymbol{B}[i]\boldsymbol{c}[i]\|_q > 0$ if and only if $\|\boldsymbol{c}[i]\|_q > 0$. On the other hand, when the blocks consist of linearly dependent vectors, we may have $\|\boldsymbol{c}[i]\|_q > 0$ while $\|\boldsymbol{B}[i]\boldsymbol{c}[i]\|_q = 0$. As a result, while P'_{ℓ_q/ℓ_0} still finds the minimum number of active subspaces, it does not necessarily finds the minimum number of nonzero coefficient blocks. In practice, to prevent such overfittings in the case of redundant blocks, we need to add a small regularization term on the values of the coefficients, such as $\|\boldsymbol{c}\|_2^2$, to the optimization program.

5.5 Theoretical analysis

In the previous sections, we showed that when data in multiple classes live in multiple low-dimensional subspaces, the classification problem can be cast as a structured sparse recovery problem where we are interested in solving the non-convex optimization programs P_{ℓ_q/ℓ_0} and P'_{ℓ_q/ℓ_0}.

In this section, we study conditions under which the optimization programs P_{ℓ_q/ℓ_1} and P'_{ℓ_q/ℓ_1} can find the minimum number of nonzero coefficient blocks, $\boldsymbol{c}[i]$, and the minimum number of reconstructed vectors, $\boldsymbol{B}[i]\boldsymbol{c}[i] \in \mathcal{S}_i$, respectively. Unlike the state-of-the-art structured sparse recovery literature [36,37,50] that only consider the case where $q = 2$ and the data in each block are linearly independent, our theoretical analysis allows for arbitrary $q > 0$. Also, motivated by practical problems such as classification, we allow for arbitrary number of data in each block.

Before presenting theoretical guarantees for the two classes of optimization programs, we define the setting for the problem and present definitions that characterize the relationships between subspaces and among data points in each subspace.

5.5.1 Problem settings and definitions

We consider the problem of block-sparse recovery in a union of subspaces. We assume that the dictionary $\boldsymbol{B} \in \mathbb{R}^{D\times N}$ consists of n blocks and the vectors in each block $\boldsymbol{B}[i] \in \mathbb{R}^{D\times m_i}$ live in a linear subspace $\mathcal{S}_i$ of dimension d_i. Unlike the state-of-

the-art block-sparse recovery literature, we do not restrict the blocks to have linearly independent columns. Instead, we allow for both non-redundant ($m_i = d_i$) and redundant ($m_i > d_i$) blocks. For reasons that will become clear in the subsequent sections, throughout this section, we assume that the subspaces $\{\mathcal{S}_i\}_{i=1}^n$ spanned by the columns of the blocks $\{\boldsymbol{B}[i]\}_{i=1}^n$ are disjoint.

In order to characterize a dictionary $\boldsymbol{B}$, we introduce two notions that characterize the relationship between the blocks and among the atoms of each block of the dictionary. We start by introducing notions that capture the inter-block relationships of a dictionary. To do so, we make use of the subspaces associated with the blocks.

Definition 4 *The* subspace coherence *between two disjoint subspaces $\mathcal{S}_i$ and $\mathcal{S}_j$ is defined as*

$$\mu(\mathcal{S}_i, \mathcal{S}_j) = \max_{\mathbf{x} \in \mathcal{S}_i, \boldsymbol{z} \in \mathcal{S}_j} \frac{|\mathbf{x}^\top \boldsymbol{z}|}{\|\mathbf{x}\|_2 \|\boldsymbol{z}\|_2} \in [0, 1). \tag{5.13}$$

The mutual subspace coherence*, μ_S, is defined as the largest subspace coherence among all pairs of subspaces,*

$$\mu_S \triangleq \max_{i \neq j} \mu(\mathcal{S}_i, \mathcal{S}_j). \tag{5.14}$$

Notice from Definition 2 that two disjoint subspaces intersect only at the origin. Therefore, their subspace coherence is always smaller than one.[1] The following result shows how to compute the subspace coherence efficiently from the singular values of a matrix obtained from the subspace bases [57].

[1]Note that the smallest principal angle [57] between $\mathcal{S}_i$ and $\mathcal{S}_j$, $\theta(\mathcal{S}_i, \mathcal{S}_j)$, is related to the subspace coherence by $\mu(\mathcal{S}_i, \mathcal{S}_j) = \cos(\theta(\mathcal{S}_i, \mathcal{S}_j))$. Thus, μ_S is the cosine of the smallest principal angle among all pairs of different subspaces.

Proposition 2 *Let $\mathcal{S}_i$ and $\mathcal{S}_j$ be two disjoint subspaces with* orthonormal bases $\boldsymbol{A}_i$ *and $\boldsymbol{A}_j$, respectively. The subspace coherence $\mu(\mathcal{S}_i, \mathcal{S}_j)$ is given by*

$$\mu(\mathcal{S}_i, \mathcal{S}_j) = \sigma_1(\boldsymbol{A}_i^\top \boldsymbol{A}_j). \tag{5.15}$$

It follows from Definition 4 that the mutual subspace coherence can be computed as

$$\mu_S = \max_{i \neq j} \sigma_1(\boldsymbol{A}_i^\top \boldsymbol{A}_j). \tag{5.16}$$

Comparing this with the notion of block-coherence defined in Chapter 2, the main difference is that block-coherence, μ_B, uses directly block matrices which are assumed to be non-redundant. However, mutual subspace coherence, μ_S, uses orthonormal bases of the blocks that can be either non-redundant or redundant. The two notions coincide with each other when the blocks are non-redundant and consist of orthonormal vectors.

While the mutual subspace coherence can be easily computed, it has the shortcoming of not characterizing very well the collection of subspaces because it only reflects the most extreme correlations between subspaces. Thus, we define a notion that better characterizes the relationship between the blocks of a dictionary.

Definition 5 *Let Λ_k denote a subset of k different elements from $\{1, \ldots, n\}$. The* k-cumulative subspace coherence *is defined as*

$$\zeta_k \triangleq \max_{\Lambda_k} \max_{i \notin \Lambda_k} \sum_{j \in \Lambda_k} \mu(\mathcal{S}_i, \mathcal{S}_j). \tag{5.17}$$

Roughly speaking, the k-cumulative subspace coherence measures the maximum total subspace coherence between a fixed subspace and a collection of k other subspaces. Note that for $k = 1$, we have $\zeta_1 = \mu_S$.

Mutual/cumulative subspace coherence can be thought of as natural extensions of mutual/cumulative coherence, defined in defined in chapter 2. In fact, they are equivalent to each other for the case of one-dimensional subspaces, where each block of the dictionary consists of a single atom. The following Lemma shows the relationship between mutual and cumulative subspace coherence of a dictionary.

Lemma 2 *Consider a dictionary $\boldsymbol{B}$, which consists of n blocks. For every $k \leq n$, we have*

$$\zeta_k \leq k\mu_S. \tag{5.18}$$

The proof of Lemma 2 is straightforward and is provided in the appendix at the end of this chapter. While computing ζ_k is, in general, more costly than computing μ_S, it follows from Lemma 2 that conditions for block-sparse recovery based on ζ_k are weaker than those based on μ_S, as we will show in the next sections. In fact, for a dictionary, ζ_k can be much smaller than $k\mu_S$, which results in weaker block-sparse recovery conditions based on ζ_k.

To see this, consider the four one-dimensional subspaces shown in Figure 5.3, where $\mathcal{S}_1$ and $\mathcal{S}_2$ are orthogonal to $\mathcal{S}_3$ and $\mathcal{S}_4$, respectively. Also, the principal angles between $\mathcal{S}_1$ and $\mathcal{S}_2$ as well as $\mathcal{S}_3$ and $\mathcal{S}_4$ are equal to $\theta < \pi/4$. Hence, the ordered subspace coherences are $0 \leq 0 \leq \sin(\theta) \leq \sin(\theta) \leq \cos(\theta) \leq \cos(\theta)$. One can verify

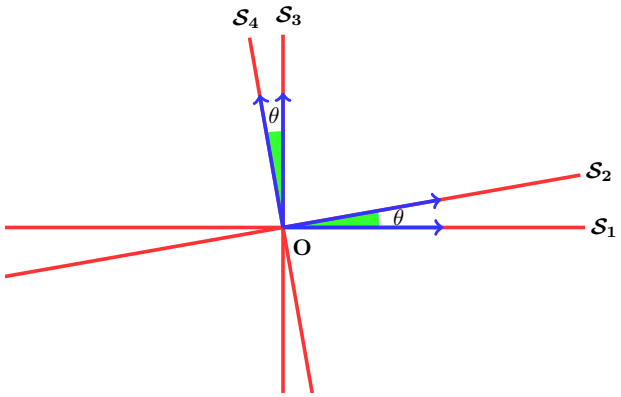


Figure 5.3: Four one-dimensional subspaces in a two-dimensional space. $\mathcal{S}_1$ and $\mathcal{S}_2$ are orthogonal to $\mathcal{S}_3$ and $\mathcal{S}_4$, respectively.

that

$$\zeta_3 = \cos(\theta) + \sin(\theta) < 3\mu_S = 3\cos(\theta).$$

In fact, for small values of θ, ζ_3 is much smaller than $3\mu_S$.[2]

Next, we introduce notions that capture the intra-block characteristics of a dictionary.

Definition 6 *Let $q > 0$. For a dictionary $\boldsymbol{B}$, define the* intra-block q-restricted isometry constant*, ϵ_q, as the smallest constant such that for every i there exists a full column-rank submatrix $\bar{\boldsymbol{B}}[i] \in \mathbb{R}^{D\times d_i}$ of $\boldsymbol{B}[i] \in \mathbb{R}^{D\times m_i}$ such that for every $\bar{\boldsymbol{c}}[i]$ we have*

$$(1-\epsilon_q)\|\bar{\boldsymbol{c}}[i]\|_q^2 \leq \|\bar{\boldsymbol{B}}[i]\bar{\boldsymbol{c}}[i]\|_2^2 \leq (1+\epsilon_q)\|\bar{\boldsymbol{c}}[i]\|_q^2. \tag{5.19}$$

Roughly speaking, ϵ_q characterizes the best q-restricted isometry property among all

[2]Another notion, which can be computed efficiently, is the sum of the k largest subspace coherences, $u_k \triangleq \mu_1 + \cdots + \mu_k$, where the sorted subspace coherences among all pairs of different subspaces are denoted by $\mu_S = \mu_1 \geq \mu_2 \geq \mu_3 \geq \cdots$. We can show that $\zeta_k \leq u_k \leq k\mu_S$. In the example of Figure 5.3, $u_3 = 2\cos(\theta) + \sin(\theta)$, which is between ζ_3 and $3\mu_S$.

submatrices of $\boldsymbol{B}[i]$ that span subspace $\mathcal{S}_i$. When $q = 2$, for a dictionary with non-redundant blocks, where $\bar{\boldsymbol{B}}[i] = \boldsymbol{B}[i]$, ϵ_2 coincides with the 1-block restricted isometry constant of $\boldsymbol{B}$ defined in chapter 2, i.e., $\epsilon_2 = \delta_{B,1}$. Thus, ϵ_q can be thought of as a generalization of the 1-block restricted isometry constant, $\delta_{B,1}$, to generic dictionaries with both non-redundant and redundant blocks and arbitrary $q \geq 1$.

Definition 7 *Let $q > 0$. For a dictionary $\boldsymbol{B}$, define the* upper intra-block q-restricted isometry constant*, σ_q, as the smallest constant such that for every i and $\boldsymbol{c}[i]$ we have*

$$\|\boldsymbol{B}[i]\boldsymbol{c}[i]\|_2^2 \leq (1 + \sigma_q)\|\boldsymbol{c}[i]\|_q^2. \tag{5.20}$$

While in general $\epsilon_q \leq \sigma_q$, for the special case of non-redundant blocks, where $\bar{\boldsymbol{B}}[i] = \boldsymbol{B}[i]$, we have $\epsilon_q = \sigma_q$.

Remark 9 *It is important to note that the theory developed in this chapter holds for any $q > 0$. However, since $q \geq 1$ leads to convex programs that can be solved more efficiently, we consider this case in our experiments.*

5.5.2 Uniqueness of block-sparse representations

Consider a dictionary $\boldsymbol{B}$ with n blocks $\boldsymbol{B}[i] \in \mathbb{R}^{D \times m_i}$ generated by disjoint subspaces $\mathcal{S}_i$ of dimensions d_i. Let $\boldsymbol{y}$ be a signal that has a block-sparse representation in $\boldsymbol{B}$ using k blocks indexed by $\{i_1, \ldots, i_k\}$. We can write

$$\boldsymbol{y} = \sum_{l=1}^{k} \boldsymbol{B}[i_l]\boldsymbol{c}[i_l] = \sum_{l=1}^{k} \boldsymbol{s}_{i_l}, \tag{5.21}$$

where $\boldsymbol{s}_{i_l} \triangleq \boldsymbol{B}[i_l]\boldsymbol{c}[i_l]$ is a vector in the subspace $\mathcal{S}_{i_l}$. In this section, we investigate conditions under which we can uniquely recover the indices $\{i_l\}$ of the blocks/subspaces as well as the vectors $\{\boldsymbol{s}_{i_l} \in \mathcal{S}_{i_l}\}$ that generate a block-sparse representation of a given $\boldsymbol{y}$. We will investigate the efficient recovery of such a block-sparse representation using the optimization programs P_{ℓ_q/ℓ_1} and P'_{ℓ_q/ℓ_1} in the subsequent sections.

In general, uniqueness of $\{\boldsymbol{s}_{i_l}\}$ is a weaker notion than the uniqueness of $\{\boldsymbol{c}[i_l]\}$ since a unique set of coefficient blocks $\{\boldsymbol{c}[i_l]\}$ uniquely determines the vectors $\{\boldsymbol{s}_{i_l}\}$, but the converse is not necessarily true. More precisely, given $\boldsymbol{s}_{i_l}$, the equation $\boldsymbol{s}_{i_l} = \boldsymbol{B}[i_l]\boldsymbol{c}[i_l]$ does not have a unique solution $\boldsymbol{c}[i_l]$ when $\boldsymbol{B}[i_l]$ is redundant. The solution is unique, only when the block is non-redundant. Therefore, the uniqueness conditions we present next, are more general than the state-of-the-art results. While [37] and [36] provide conditions for the uniqueness of the blocks $\{i_l\}$ and the coefficient blocks $\{\boldsymbol{c}[i_l]\}$, which only hold for non-redundant blocks, we provide conditions for the uniqueness of the blocks $\{i_l\}$ and the vectors $\{\boldsymbol{s}_{i_l}\}$ for generic dictionaries with non-redundant or redundant blocks. We show the following result whose proof is provided in the appendix at the end of this chapter.

Proposition 3 *Let $\bar{\boldsymbol{B}}[i] \in \mathbb{R}^{D\times d_i}$ be an arbitrary full column-rank submatrix of $\boldsymbol{B}[i] \in \mathbb{R}^{D\times m_i}$ and define*

$$\bar{\boldsymbol{B}} \triangleq \begin{bmatrix} \bar{\boldsymbol{B}}[1] & \cdots & \bar{\boldsymbol{B}}[n] \end{bmatrix}. \tag{5.22}$$

The blocks $\{i_l\}$ and the vectors $\{\boldsymbol{s}_{i_l}\}$ that generate a k-block-sparse representation of a signal can be determined uniquely if and only if $\bar{\boldsymbol{B}}\,\bar{\boldsymbol{c}} \neq 0$ for every $2k$-block-sparse

vector $\bar{\boldsymbol{c}} \neq 0$.

Remark 10 *Note that the disjointness of subspaces is a necessary condition for uniquely recovering the blocks that take part in a block-sparse representation of a signal. This comes from the fact that for* $k = 1$*, the uniqueness condition of Proposition 3 requires that any two subspaces intersect only at the origin.*

Next, we state another uniqueness result that we will use in our theoretical analysis in the next sections. For a fixed $\tau \in [0, 1)$ and for each $i \in \{1, \ldots, n\}$ define

$$\mathbb{W}_{\tau,i} \triangleq \{\boldsymbol{s}_i \in \mathcal{S}_i,\, 1 - \tau \leq \|\boldsymbol{s}_i\|_2^2 \leq 1 + \tau\}, \tag{5.23}$$

which is the set of all vectors in $\mathcal{S}_i$ whose norm is bounded by $1+\tau$ from above and by $1 - \tau$ from below. Let $\Lambda_k = \{i_1, \ldots, i_k\}$ be a set of k indices from $\{1, \ldots, n\}$. Define

$$\mathbb{B}_\tau(\Lambda_k) \triangleq \{\boldsymbol{B}_{\Lambda_k} = \begin{bmatrix} \boldsymbol{s}_{i_1} \cdots \boldsymbol{s}_{i_k} \end{bmatrix}, \boldsymbol{s}_{i_l} \in \mathbb{W}_{\tau,i_l}, 1 \leq l \leq k\}, \tag{5.24}$$

which is the set of matrices $\boldsymbol{B}_{\Lambda_k} \in \mathbb{R}^{D \times k}$ whose columns are drawn from subspaces indexed by Λ_k and their norms are bounded according to (5.23). With abuse of notation, we use $\boldsymbol{B}_k$ to indicate $\boldsymbol{B}_{\Lambda_k} \in \mathbb{B}_\tau(\Lambda_k)$ whenever Λ_k is clear from the context. For example, $\boldsymbol{B}_n \in \mathbb{R}^{D \times n}$ indicates a matrix whose columns are drawn from all n subspaces. We have the following result whose proof is provided in the appendix at the end of this chapter.

Corollary 1 *Let $\tau \in [0, 1)$. The blocks $\{i_l\}$ and the vectors $\{\boldsymbol{s}_{i_l}\}$ that constitute a k-block-sparse representation of a signal can be determined uniquely if and only if* $\text{rank}(\boldsymbol{B}_n) \geq 2k$ *for every* $\boldsymbol{B}_n \in \mathbb{B}_\tau(\Lambda_n)$.

Note that the result of Corollary 1 still holds if we let the columns of $\boldsymbol{B}_n$ have arbitrary nonzero norms, because the rank of a matrix does not change by scaling its columns with nonzero constants. However, as we will show in the next section, the bounds on the norms as in (5.23) appear when we analyze block-sparse recovery using the optimization programs P_{ℓ_q/ℓ_1} and P'_{ℓ_q/ℓ_1}. While checking the condition of Corollary 1 is not possible, as it requires computing every possible $\boldsymbol{s}_i$ in $\mathbb{W}_{\tau,i}$, we use the result of Corollary 1 in our theoretical analysis in the next sections.

In the remainder of this Chapter, we assume that a given signal $\boldsymbol{y}$ has a *unique* k-block-sparse representation in $\boldsymbol{B}$.

Definition 8 *By uniqueness of a block-sparse representation, we mean that the blocks Λ_k and the vectors $\{\boldsymbol{s}_i \in \mathcal{S}_i\}_{i\in\Lambda_k}$ for which $\boldsymbol{y} = \sum_{i\in\Lambda_k} \boldsymbol{s}_i$ can be determined uniquely.*

Under the uniqueness assumption, in the subsequent sections, we investigate conditions under which P_{ℓ_q/ℓ_1} and P'_{ℓ_q/ℓ_1} recover the unique set of nonzero blocks Λ_k and the unique vectors $\{\boldsymbol{s}_i \in \mathcal{S}_i\}_{i\in\Lambda_k}$.

5.5.3 Block-sparse recovery via P_{ℓ_q/ℓ_1}

In this section, we study conditions under which P_{ℓ_q/ℓ_1} recovers the unique block-sparse representation of a given signal for both the case of non-redundant and redundant blocks.

To that end, let Λ_k be a set of k indices from $\{1, \cdots, n\}$ and $\Lambda_{\widehat{k}}$ be the set of the remaining $n-k$ indices. Let $\mathbf{x}$ be a nonzero vector in the intersection of $\oplus_{i\in\Lambda_k}\mathcal{S}_i$ and $\oplus_{i\in\Lambda_{\widehat{k}}}\mathcal{S}_i$, where $\oplus$ denotes the direct sum operator. Let the minimum ℓ_q/ℓ_1-norm coefficient vector when we choose only the k blocks of $\boldsymbol{B}$ indexed by Λ_k be

$$\breve{\boldsymbol{c}}^* = \operatorname{argmin} \sum_{i\in\Lambda_k} \|\boldsymbol{c}[i]\|_q \quad \text{s.t.} \quad \mathbf{x} = \sum_{i\in\Lambda_k} \boldsymbol{B}[i]\boldsymbol{c}[i], \tag{5.25}$$

and let the minimum ℓ_q/ℓ_1-norm coefficient vector when we choose the blocks indexed by $\Lambda_{\widehat{k}}$ be

$$\widehat{\boldsymbol{c}}^* = \operatorname{argmin} \sum_{i\in\Lambda_{\widehat{k}}} \|\boldsymbol{c}[i]\|_q \quad \text{s.t.} \quad \mathbf{x} = \sum_{i\in\Lambda_{\widehat{k}}} \boldsymbol{B}[i]\boldsymbol{c}[i]. \tag{5.26}$$

The following theorem gives conditions under which the convex program P_{ℓ_q/ℓ_1} is guaranteed to successfully recover a k-block-sparse representation of a given signal.

Theorem 4 *For any signal that has a unique k-block-sparse representation in $\boldsymbol{B}$, the optimization program P_{ℓ_q/ℓ_1} recovers the unique block-sparse representation if and only if*

$$\forall \Lambda_k,\, \forall \mathbf{x} \in (\oplus_{i\in\Lambda_k}\mathcal{S}_i) \,\cap\, (\oplus_{i\in\Lambda_{\widehat{k}}}\mathcal{S}_i),\, \mathbf{x}\neq\mathbf{0} \implies \sum_{i\in\Lambda_k} \|\breve{\boldsymbol{c}}^*[i]\|_q < \sum_{i\in\Lambda_{\widehat{k}}} \|\widehat{\boldsymbol{c}}^*[i]\|_q. \tag{5.27}$$

Proof. ($\Longleftarrow$) Fix Λ_k and $\boldsymbol{y}$ in $\oplus_{i\in\Lambda_k}\mathcal{S}_i$ and let $\boldsymbol{c}^*$ be the solution of P_{ℓ_q/ℓ_1}. If $\boldsymbol{c}^*$ has at most k nonzero blocks, then by the uniqueness assumption the nonzero blocks are indexed by Λ_k. For the sake of contradiction, assume that $\boldsymbol{c}^*$ has more than k nonzero blocks, so $\boldsymbol{c}^*$ is nonzero for some blocks in $\Lambda_{\widehat{k}}$. Define

$$\mathbf{x} \triangleq \boldsymbol{y} - \sum_{i\in\Lambda_k} \boldsymbol{B}[i]\boldsymbol{c}^*[i] = \sum_{i\in\Lambda_{\widehat{k}}} \boldsymbol{B}[i]\boldsymbol{c}^*[i]. \tag{5.28}$$

From (5.28) we have that $\mathbf{x}$ lives in the intersection of $\oplus_{i\in\Lambda_k}\mathcal{S}_i$ and $\oplus_{i\in\Lambda_{\widehat{k}}}\mathcal{S}_i$. Let $\breve{\boldsymbol{c}}^*$ and $\widehat{\boldsymbol{c}}^*$ be respectively the solutions of the optimization problems in (5.25) and (5.26), for $\mathbf{x}$. We can write

$$\mathbf{x} = \sum_{i\in\Lambda_k} \boldsymbol{B}[i]\breve{\boldsymbol{c}}^*[i] = \sum_{i\in\Lambda_{\widehat{k}}} \boldsymbol{B}[i]\widehat{\boldsymbol{c}}^*[i]. \tag{5.29}$$

We also have the following inequalities

$$\sum_{i\in\Lambda_k} \|\breve{\boldsymbol{c}}^*[i]\|_q < \sum_{i\in\Lambda_{\widehat{k}}} \|\widehat{\boldsymbol{c}}^*[i]\|_q \leq \sum_{i\in\Lambda_{\widehat{k}}} \|\boldsymbol{c}^*[i]\|_q, \tag{5.30}$$

where the first inequality follows from the sufficient condition in (5.27). The second inequality follows from the second inequalities in (5.28) and (5.29) and the fact that $\widehat{\boldsymbol{c}}^*$ is the optimal solution of (5.26) for $\mathbf{x}$. Using the first equalities in (5.28) and (5.29), we can rewrite $\boldsymbol{y}$ as

$$\boldsymbol{y} = \sum_{i\in\Lambda_k} \boldsymbol{B}[i](\boldsymbol{c}^*[i] + \breve{\boldsymbol{c}}^*[i]), \tag{5.31}$$

which implies that $\boldsymbol{c}^* + \breve{\boldsymbol{c}}^*$ is a solution of $\boldsymbol{y} = \boldsymbol{B}\boldsymbol{c}$. Finally, using (5.30) and the

triangle inequality, we obtain

$$\sum_{i\in\Lambda_k}\|\boldsymbol{c}^*[i]+\breve{\boldsymbol{c}}^*[i]\|_q \leq \sum_{i\in\Lambda_k}\|\boldsymbol{c}^*[i]\|_q + \sum_{i\in\Lambda_k}\|\breve{\boldsymbol{c}}^*[i]\|_q$$
$$< \sum_{i\in\Lambda_k}\|\boldsymbol{c}^*[i]\|_q + \sum_{i\in\Lambda_{\widehat{k}}}\|\widehat{\boldsymbol{c}}^*[i]\|_q \leq \sum_{i=1}^{n}\|\boldsymbol{c}^*[i]\|_q. \quad (5.32)$$

This contradicts the optimality of $\boldsymbol{c}^*$, since it means that $\boldsymbol{c}^* + \breve{\boldsymbol{c}}^*$, which is also a solution of $\boldsymbol{y} = \boldsymbol{B}\boldsymbol{c}$, has a strictly smaller ℓ_q/ℓ_1-norm than $\boldsymbol{c}^*$.

($\Longrightarrow$) We prove this using contradiction. Assume there exist Λ_k and $\mathbf{x}$ in the intersection of $\oplus_{i\in\Lambda_k}\mathcal{S}_i$ and $\oplus_{i\in\Lambda_{\widehat{k}}}\mathcal{S}_i$ for which the condition in (5.27) does not hold, i.e.,

$$\sum_{i\in\Lambda_{\widehat{k}}}\|\widehat{\boldsymbol{c}}^*[i]\|_q \leq \sum_{i\in\Lambda_k}\|\breve{\boldsymbol{c}}^*[i]\|_q. \quad (5.33)$$

Thus, a solution of $\mathbf{x} = \boldsymbol{B}\boldsymbol{c}$ is given by $\widehat{\boldsymbol{c}}^*$ that is not k-block-sparse and has a ℓ_q/ℓ_1-norm that is smaller than or equal to any k-block-sparse solution, contradicting the assumption that P_{ℓ_q/ℓ_1} recovers the uniques k-block-sparse representation of any given signal. ∎

The condition of Theorem 4 (and Theorem 5 in the next section) is closely related to the nullspace property in [27, 70, 96, 108]. However, the key difference is that we do not require the condition of Theorem 4 to hold for all feasible vectors of (5.25) and (5.26), denoted by $\breve{\boldsymbol{c}}$ and $\widehat{\boldsymbol{c}}$, respectively. Instead, we only require the condition of Theorem 4 to hold for the optimal solutions of (5.25) and (5.26). Thus, while the nullspace property might be violated by some feasible vectors $\breve{\boldsymbol{c}}$ and $\widehat{\boldsymbol{c}}$, our condition can still hold for $\breve{\boldsymbol{c}}^*$ and $\widehat{\boldsymbol{c}}^*$, guaranteeing the equivalence of the two optimization

programs.

Notice that it is not possible to check the condition in (5.27) for every Λ_k and for every $\mathbf{x}$ in the intersection of $\oplus_{i\in\Lambda_k}\mathcal{S}_i$ and $\oplus_{i\in\Lambda_{\widehat{k}}}\mathcal{S}_i$. In addition, the condition in (5.27) does not explicitly incorporate the inter-block and intra-block parameters of the dictionary. In what follows, we propose sufficient conditions that incorporate the inter-block and intra-block parameters of the dictionary and can be efficiently checked. We use the following Lemma, which is a generalization of Theorem 3.5 in [104], whose proof is provided in the Appendix.

Lemma 3 *Let $\boldsymbol{E}_k \in \mathbb{R}^{D\times k}$ be a matrix whose columns are chosen from subspaces indexed by Λ_k and $\boldsymbol{E}_k \in \mathbb{B}_\alpha(\Lambda_k)$ for a fixed $\alpha \in [0,1)$. Let $\boldsymbol{E}_{\widehat{k}} \in \mathbb{R}^{D\times n-k}$ be a matrix whose columns are chosen from subspaces indexed by $\Lambda_{\widehat{k}}$ where the Euclidean norm of each column is less than or equal to $\sqrt{1+\beta}$. We have*

$$\|(\boldsymbol{E}_k^\top \boldsymbol{E}_k)^{-1}\boldsymbol{E}_k^\top \boldsymbol{E}_{\widehat{k}}\|_{1,1} \leq \frac{\sqrt{(1+\alpha)(1+\beta)}\,\zeta_k}{1-[\,\alpha+(1+\alpha)\zeta_{k-1}\,]}. \tag{5.34}$$

Proposition 4 *For any signal that has a unique k-block-sparse representation in $\boldsymbol{B}$, the optimization program P_{ℓ_q/ℓ_1} recovers the unique block-sparse representation if*

$$\sqrt{\frac{1+\sigma_q}{1+\epsilon_q}}\,\zeta_k + \zeta_{k-1} < \frac{1-\epsilon_q}{1+\epsilon_q}. \tag{5.35}$$

Proof. Fix a set $\Lambda_k = \{i_1,\ldots,i_k\}$ of k indices from $\{1,\ldots,n\}$ and let $\Lambda_{\widehat{k}} = \{i_{k+1},\ldots,i_n\}$ denote the set of the remaining indices. Consider a signal $\mathbf{x}$ in the intersection of $\oplus_{i\in\Lambda_k}\mathcal{S}_i$ and $\oplus_{i\in\Lambda_{\widehat{k}}}\mathcal{S}_i$. The structure of the proof is as follows. We

show that $\mathbf{x}$ can be written as $\mathbf{x} = \boldsymbol{B}_k \boldsymbol{a}_k$, where for the solution of (5.25), we have $\sum_{i \in \Lambda_k} \|\breve{\boldsymbol{c}}^*[i]\|_q \leq \|\boldsymbol{a}_k\|_1$. Also, we show that for the solution of (5.26), one can write $\mathbf{x} = \boldsymbol{B}_{\widehat{k}} \boldsymbol{a}_{\widehat{k}}$, where $\|\boldsymbol{a}_{\widehat{k}}\|_1 = \sum_{i \in \Lambda_{\widehat{k}}} \|\widehat{\boldsymbol{c}}^*[i]\|_q$. Under the sufficient condition of the Proposition, we show that $\|\boldsymbol{a}_k\|_1 < \|\boldsymbol{a}_{\widehat{k}}\|_1$, implying that the condition of Theorem 4 is satisfied.

To start, let $\widehat{\boldsymbol{c}}^*$ be the solution of the optimization program in (5.26). For every $i \in \Lambda_{\widehat{k}}$, define the vectors $\boldsymbol{s}_i$ and the scalars a_i as follows. If $\widehat{\boldsymbol{c}}^*[i] \neq 0$ and $\boldsymbol{B}[i]\widehat{\boldsymbol{c}}^*[i] \neq 0$, let

$$\boldsymbol{s}_i \triangleq \frac{\boldsymbol{B}[i]\widehat{\boldsymbol{c}}^*[i]}{\|\widehat{\boldsymbol{c}}^*[i]\|_q}, \quad a_i \triangleq \|\widehat{\boldsymbol{c}}^*[i]\|_q. \tag{5.36}$$

Otherwise, let $\boldsymbol{s}_i$ be an arbitrary vector in $\mathcal{S}_i$ of unit Euclidean norm and $a_i = 0$. We can write

$$\mathbf{x} = \sum_{i \in \Lambda_{\widehat{k}}} \boldsymbol{B}[i]\widehat{\boldsymbol{c}}^*[i] = \boldsymbol{B}_{\widehat{k}} \boldsymbol{a}_{\widehat{k}}, \tag{5.37}$$

where $\boldsymbol{B}_{\widehat{k}} \triangleq \begin{bmatrix} \boldsymbol{s}_{i_{k+1}} & \cdots & \boldsymbol{s}_{i_n} \end{bmatrix}$ and $\boldsymbol{a}_{\widehat{k}} \triangleq \begin{bmatrix} a_{i_{k+1}} & \cdots & a_{i_n} \end{bmatrix}^\top$. Note that from Definition 7, we have $\|\boldsymbol{s}_i\|_2 \leq \sqrt{1+\sigma_q}$ for every $i \in \Lambda_{\widehat{k}}$.

Let $\bar{\boldsymbol{B}}[i] \in \mathbb{R}^{D \times d_i}$ be the submatrix of $\boldsymbol{B}[i]$ associated with ϵ_q according to Definition 6. Since $\bar{\boldsymbol{B}}[i]$ spans the subspace $\mathcal{S}_i$, there exists $\bar{\boldsymbol{c}}[i]$ such that

$$\mathbf{x} = \sum_{i \in \Lambda_k} \bar{\boldsymbol{B}}[i]\bar{\boldsymbol{c}}[i] \triangleq \boldsymbol{B}_k \boldsymbol{a}_k, \tag{5.38}$$

where $\boldsymbol{B}_k \triangleq \begin{bmatrix} \boldsymbol{s}_{i_1} & \cdots & \boldsymbol{s}_{i_k} \end{bmatrix}$ and $\boldsymbol{a}_k \triangleq \begin{bmatrix} a_{i_1} & \cdots & a_{i_k} \end{bmatrix}^\top$. For every $i \in \Lambda_k$ the vectors $\boldsymbol{s}_i$

and the scalars a_i are defined as

$$\boldsymbol{s}_i \triangleq \frac{\bar{\boldsymbol{B}}[i]\bar{\boldsymbol{c}}[i]}{\|\bar{\boldsymbol{c}}[i]\|_q}, \quad a_i \triangleq \|\bar{\boldsymbol{c}}[i]\|_q, \tag{5.39}$$

whenever $\bar{\boldsymbol{c}}[i] \neq 0$ and $\bar{\boldsymbol{B}}[i]\bar{\boldsymbol{c}}[i] \neq 0$. Otherwise, we let $\boldsymbol{s}_i$ be an arbitrary vector in $\mathcal{S}_i$ of unit Euclidean norm and $a_i = 0$. Clearly, $\boldsymbol{B}_k \in \mathbb{B}_{\epsilon_q}(\Lambda_k)$ is full column-rank using Corollary 1 when $\epsilon_q \in [0, 1)$. Hence, we have $\boldsymbol{a}_k = (\boldsymbol{B}_k^\top \boldsymbol{B}_k)^{-1}\boldsymbol{B}_k^\top \mathbf{x}$ and consequently,

$$\|\boldsymbol{a}_k\|_1 = \|(\boldsymbol{B}_k^\top \boldsymbol{B}_k)^{-1}\boldsymbol{B}_k^\top \mathbf{x}\|_1. \tag{5.40}$$

Substituting $\boldsymbol{y}$ from (5.37) in the above equation, we obtain

$$\|\boldsymbol{a}_k\|_1 = \|(\boldsymbol{B}_k^\top \boldsymbol{B}_k)^{-1}\boldsymbol{B}_k^\top \boldsymbol{B}_{\widehat{k}}\boldsymbol{a}_{\widehat{k}}\|_1 \leq \|(\boldsymbol{B}_k^\top \boldsymbol{B}_k)^{-1}\boldsymbol{B}_k^\top \boldsymbol{B}_{\widehat{k}}\|_{1,1}\|\boldsymbol{a}_{\widehat{k}}\|_1. \tag{5.41}$$

Using Lemma 3 with $\alpha = \epsilon_q$ and $\beta = \sigma_q$, we have

$$\|(\boldsymbol{B}_k^\top \boldsymbol{B}_k)^{-1}\boldsymbol{B}_k^\top \boldsymbol{B}_{\widehat{k}}\|_1 \leq \frac{\sqrt{(1+\epsilon_q)(1+\sigma_q)}\,\zeta_k}{1-[\,\epsilon_q + (1+\epsilon_q)\zeta_{k-1}\,]}. \tag{5.42}$$

Thus, if the right hand side of the above equation is strictly less than one, i.e., if the condition of the proposition is satisfied, then from (5.41) we have $\|\boldsymbol{a}_k\|_1 < \|\boldsymbol{a}_{\widehat{k}}\|_1$. Finally, using the optimality of $\check{\boldsymbol{c}}^*$ when we choose the blocks indexed by Λ_k, we obtain

$$\sum_{i\in\Lambda_k}\|\check{\boldsymbol{c}}^*[i]\|_q \leq \sum_{i\in\Lambda_k}\|\bar{\boldsymbol{c}}[i]\|_q = \|\boldsymbol{a}_k\|_1 < \|\boldsymbol{a}_{\widehat{k}}\|_1 = \sum_{i\in\Lambda_{\widehat{k}}}\|\widehat{\boldsymbol{c}}^*[i]\|_q, \tag{5.43}$$

which implies that the condition of Theorem 4 is satisfied. Thus, the convex program P_{ℓ_q/ℓ_1} recovers a k-block-sparse representation of a given signal. ∎

The following corollary derives stronger, but simpler to check, sufficient conditions for block-sparse recovery using P_{ℓ_q/ℓ_1}.

Corollary 2 *For any signal that has a unique k-block-sparse representation in $\boldsymbol{B}$, the optimization program P_{ℓ_q/ℓ_1} recovers the unique block-sparse representation if and only if* [3]

$$(k\sqrt{\frac{1+\sigma_q}{1+\epsilon_q}}+k-1)\mu_S < \frac{1-\epsilon_q}{1+\epsilon_q}. \tag{5.44}$$

Proof. The result follows from Proposition 4 by using the fact that $\zeta_k \leq k\mu_S$ from Lemma 2. ∎

For non-redundant blocks, we have $\sigma_q = \epsilon_q$. Thus, in this case, for the convex program P_{ℓ_q/ℓ_1}, the block-sparse recovery condition based on the mutual subspace coherence in (5.44) reduces to

$$(2k-1)\mu_S < \frac{1-\epsilon_q}{1+\epsilon_q}. \tag{5.45}$$

Also, the block-sparse recovery condition based on the cumulative subspace coherence in (5.35) reduces to

$$\zeta_k + \zeta_{k-1} < \frac{1-\epsilon_q}{1+\epsilon_q}, \tag{5.46}$$

which is always weaker than the condition based on the mutual subspace coherence in (5.45).

[3] An intermediate sufficient condition is given by $\sqrt{\frac{1+\sigma_q}{1+\epsilon_q}}\ u_k + u_{k-1} < \frac{1-\epsilon_q}{1+\epsilon_q}$ using the fact that $\zeta_k \leq u_k \leq k\mu_S$.

5.5.4 Block-sparse recovery via P'_{ℓ_q/ℓ_1}

In this section, we study conditions under which P'_{ℓ_q/ℓ_1} recovers the unique block-sparse representation of a given signal for both the case of non-redundant and redundant blocks. Our approach is similar to the one in the previous section.

Let Λ_k be a set of k indices from $\{1,\ldots,n\}$ and $\Lambda_{\widehat{k}}$ be the set of the remaining indices. For a nonzero signal $\mathbf{x}$ in the intersection of $\oplus_{i\in\Lambda_k}\mathcal{S}_i$ and $\oplus_{i\in\Lambda_{\widehat{k}}}\mathcal{S}_i$, let the minimum ℓ_q/ℓ_1-norm coefficient vector when we choose only the blocks of $\boldsymbol{B}$ indexed by Λ_k be

$$\breve{\boldsymbol{c}}^* = \operatorname{argmin} \sum_{i\in\Lambda_k} \|\boldsymbol{B}[i]\boldsymbol{c}[i]\|_q \quad \text{s.t.} \quad \mathbf{x} = \sum_{i\in\Lambda_k} \boldsymbol{B}[i]\boldsymbol{c}[i]. \tag{5.47}$$

Also, let the minimum ℓ_q/ℓ_1-norm coefficient vector when we choose the blocks of $\boldsymbol{B}$ indexed by $\Lambda_{\widehat{k}}$ be

$$\widehat{\boldsymbol{c}}^* = \operatorname{argmin} \sum_{i\in\Lambda_{\widehat{k}}} \|\boldsymbol{B}[i]\boldsymbol{c}[i]\|_q \quad \text{s.t.} \quad \mathbf{x} = \sum_{i\in\Lambda_{\widehat{k}}} \boldsymbol{B}[i]\boldsymbol{c}[i]. \tag{5.48}$$

We have the following result.

Theorem 5 *For any signal that has a unique k-block-sparse representation in $\boldsymbol{B}$, the optimization program P'_{ℓ_q/ℓ_1} recovers the unique block-sparse representation if and only if*

$$\forall \Lambda_k, \forall \mathbf{x} \in (\oplus_{i\in\Lambda_k}\mathcal{S}_i) \cap (\oplus_{i\in\Lambda_{\widehat{k}}}\mathcal{S}_i),\, \mathbf{x}\neq\mathbf{0} \implies \sum_{i\in\Lambda_k} \|\boldsymbol{B}[i]\breve{\boldsymbol{c}}^*[i]\|_q < \sum_{i\in\Lambda_{\widehat{k}}} \|\boldsymbol{B}[i]\widehat{\boldsymbol{c}}^*[i]\|_q. \tag{5.49}$$

Proof. ($\Longleftarrow$) Let $\boldsymbol{y}$ be a signal that lives in the subspace $\oplus_{i\in\Lambda_k}\mathcal{S}_i$. Denote by $\boldsymbol{c}^*$ the solution of the optimization program P'_{ℓ_q/ℓ_1}. If for at most k blocks of $\boldsymbol{c}^*$ we have

$\boldsymbol{B}[i]\boldsymbol{c}^*[i] \neq \mathbf{0}$, then by the uniqueness assumption, these blocks of $\boldsymbol{c}^*$ are indexed by Λ_k. For the sake of contradiction, assume that $\boldsymbol{B}[i]\boldsymbol{c}^*[i] \neq \mathbf{0}$ for some $i \in \Lambda_{\widehat{k}}$. Define

$$\mathbf{x} \triangleq \boldsymbol{y} - \sum_{i \in \Lambda_k} \boldsymbol{B}[i]\boldsymbol{c}^*[i] = \sum_{i \in \Lambda_{\widehat{k}}} \boldsymbol{B}[i]\boldsymbol{c}^*[i]. \tag{5.50}$$

The remaining steps of the proof are analogous to the proof of Theorem 4 except that we replace $\|\boldsymbol{c}^*[i]\|_q$ by $\|\boldsymbol{B}[i]\boldsymbol{c}^*[i]\|_q$ in (5.30) and use the triangle inequality for $\|\boldsymbol{B}[i](\boldsymbol{c}^*[i] + \breve{\boldsymbol{c}}^*[i])\|_q$ in (5.32).

($\Longrightarrow$) We prove this using contradiction. Assume that there exist Λ_k and $\mathbf{x}$ in the intersection of $\oplus_{i \in \Lambda_k} \mathcal{S}_i$ and $\oplus_{i \in \Lambda_{\widehat{k}}} \mathcal{S}_i$ for which the condition in (5.49) does not hold, i.e.,

$$\sum_{i \in \Lambda_{\widehat{k}}} \|\boldsymbol{B}[i]\widehat{\boldsymbol{c}}^*[i]\|_q \leq \sum_{i \in \Lambda_k} \|\boldsymbol{B}[i]\breve{\boldsymbol{c}}^*[i]\|_q. \tag{5.51}$$

Thus, a solution of $\mathbf{x} = \boldsymbol{B}\boldsymbol{c}$ is given by $\widehat{\boldsymbol{c}}^*$ that is not k-block-sparse and whose linear transformation by $\boldsymbol{B}$ has a ℓ_q/ℓ_1-norm that is smaller than or equal to the norm of the transformation by $\boldsymbol{B}$ of any k-block-sparse solution, contradicting the assumption that P'_{ℓ_q/ℓ_1} recovers the uniques k-block-sparse representation of any given signal. ∎

Next, we propose sufficient conditions that incorporate the inter-block and intra-block parameters of the dictionary and can be efficiently checked. Before that we need to introduce the following notation.

Definition 9 *Consider a dictionary $\boldsymbol{B}$ with blocks $\boldsymbol{B}[i] \in \mathbb{R}^{D \times m_i}$. Define ϵ'_q as the smallest constant such that for every i and $\boldsymbol{c}[i]$ we have*

$$(1 - \epsilon'_q)\|\boldsymbol{B}[i]\boldsymbol{c}[i]\|_q^2 \leq \|\boldsymbol{B}[i]\boldsymbol{c}[i]\|_2^2 \leq (1 + \epsilon'_q)\|\boldsymbol{B}[i]\boldsymbol{c}[i]\|_q^2. \tag{5.52}$$

Note that ϵ'_q characterizes the relation between the ℓ_q and ℓ_2 norms of vectors in $\mathbb{R}^D$ and does not depend on whether the blocks are non-redundant or redundant. In addition, for $q = 2$, we have $\epsilon'_2 = 0$.

Proposition 5 *For any signal that has a unique k-block-sparse representation in $\boldsymbol{B}$, the optimization program P_{ℓ_q/ℓ_1} recovers the unique block-sparse representation if*

$$\zeta_k + \zeta_{k-1} < \frac{1 - \epsilon'_q}{1 + \epsilon'_q}. \tag{5.53}$$

Proof. The proof is provided in the Appendix. ∎

The following corollary derives stronger yet simpler to check sufficient conditions for block-sparse recovery using P'_{ℓ_q/ℓ_1}.

Corollary 3 *For any signal that has a unique k-block-sparse representation in $\boldsymbol{B}$, the optimization program P_{ℓ_q/ℓ_1} recovers the unique block-sparse representation if* [4]

$$(2k - 1)\mu_S < \frac{1 - \epsilon'_q}{1 + \epsilon'_q}. \tag{5.54}$$

Proof. The result follows from Proposition 5 by using the fact that $\zeta_k \leq k\mu_S$ from Lemma 2. ∎

Unlike the conditions for the case of P_{ℓ_q/ℓ_1}, which depend on whether the blocks are non-redundant or redundant, the conditions for the case of P'_{ℓ_q/ℓ_1} do not depend on

[4] An intermediate sufficient condition is given by $u_k + u_{k-1} < \frac{1-\epsilon'_q}{1+\epsilon'_q}$ using the fact that $\zeta_k \leq u_k \leq k\mu_S$.

the redundancy of the blocks. In addition, since $\epsilon_2' = 0$, the condition for block-sparse recovery using P'_{ℓ_2/ℓ_1} based on the mutual subspace coherence reduces to

$$(2k-1)\mu_S < 1, \tag{5.55}$$

and the condition based on the cumulative subspace coherence reduces to

$$\zeta_k + \zeta_{k-1} < 1. \tag{5.56}$$

Remark 11 *Note that the sufficient conditions in* (5.55) *and* (5.56) *are weaker than the sufficient conditions in* (5.45) *and* (5.46)*, respectively. While we can not assert the superiority of P'_{ℓ_2/ℓ_1} over P_{ℓ_2/ℓ_1}, since the conditions are only sufficient not necessary, as we will show in the experimental results P'_{ℓ_2/ℓ_1} is in general more successful than P_{ℓ_2/ℓ_1} for block-sparse recovery.*

Remark 12 *Under the uniqueness assumption, both nonconvex programs P_{ℓ_q/ℓ_0} and P'_{ℓ_q/ℓ_0} find the unique blocks Λ_k and the vectors $\{\boldsymbol{s}_i \in \mathcal{S}_i\}_{i\in\Lambda_k}$ for which $\boldsymbol{y} = \sum_{i\in\Lambda_k} \boldsymbol{s}_i$. Thus, when the conditions for the success of the convex programs P_{ℓ_q/ℓ_1} and P'_{ℓ_q/ℓ_1} hold, their optimal solutions correspond to Λ_k and $\{\boldsymbol{s}_i \in \mathcal{S}_i\}_{i\in\Lambda_k}$. For non-redundant blocks, this implies that the optimal coefficient vectors found by P_{ℓ_q/ℓ_1} and P'_{ℓ_q/ℓ_1} are the same and equal to the true solution.*

5.5.5 Correcting sparse outlying entries

In real-world problems, observed signals might be corrupted by errors [115, 117], hence might not perfectly lie in the range-space of a few blocks of the dictionary [42].

A case of interest, which also happens in practice, is when the observed signal is corrupted with an error that has a few outlying entries. For example, in the face recognition problem, a face image might be corrupted because of occlusions [117], or in the motion segmentation problem, some of the entries of feature trajectories might be corrupted due to objects partially occluding each other or malfunctioning of the tracker [40,88]. In such cases, the observed signal $\boldsymbol{y}$ can be modeled as a superposition of a pure signal $\boldsymbol{y}_0$ and a corruption term $\boldsymbol{e}$ of the form $\boldsymbol{y} = \boldsymbol{y}_0 + \boldsymbol{e}$, where $\boldsymbol{y}_0$ has a block sparse representation in the dictionary $\boldsymbol{B}$ and $\boldsymbol{e}$ has a few large nonzero entries. Thus, $\boldsymbol{y}$ can be written as

$$\boldsymbol{y} = \boldsymbol{y}_0 + \boldsymbol{e} = \boldsymbol{B}\boldsymbol{c} + \boldsymbol{e} = \begin{bmatrix} \boldsymbol{B} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{e} \end{bmatrix}. \tag{5.57}$$

Note that the new dictionary $\begin{bmatrix} \boldsymbol{B} & \boldsymbol{I} \end{bmatrix}$ has still a block structure whose blocks correspond to the blocks of $\boldsymbol{B}$ and the atoms of $\boldsymbol{I}$. Thus, in this new dictionary, $\boldsymbol{y}$ has a block-sparse representation with a few blocks corresponding to $\boldsymbol{B}$ and a few blocks/atoms corresponding to $\boldsymbol{I}$. Assuming that the sufficient conditions of the previous sections hold for the dictionary $\begin{bmatrix} \boldsymbol{B} & \boldsymbol{I} \end{bmatrix}$, we can recover a block-sparse representation of a corrupted signal using the optimization program $\bar{P}_{\ell_q/\ell_1}$ as

$$\bar{P}_{\ell_q/\ell_1} : \min \sum_{i=1}^{n} \|\boldsymbol{c}[i]\|_q + \|\boldsymbol{e}\|_1 \quad \text{s.t.} \quad \boldsymbol{y} = \begin{bmatrix} \boldsymbol{B} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{e} \end{bmatrix}, \tag{5.58}$$

or using the optimization program $\bar{P}'_{\ell_q/\ell_1}$ as

$$\bar{P}'_{\ell_q/\ell_1} : \min \sum_{i=1}^{n} |\boldsymbol{B}[i]\boldsymbol{c}[i]\|_q + \|\boldsymbol{e}\|_1 \qquad \boldsymbol{y} = \begin{bmatrix} \boldsymbol{B} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{e} \end{bmatrix}. \tag{5.59}$$

Here, we used the fact that the blocks of $\boldsymbol{I}$ are of length one, i.e., $\boldsymbol{e}[i] \in \mathbb{R}$. Thus, $\sum_{i=1}^{D} \|\boldsymbol{e}[i]\|_q = \sum_{i=1}^{D} |\boldsymbol{e}[i]| = \|\boldsymbol{e}\|_1$.

As a result, our theoretical analysis in this chapter for block-sparse recovery also provides guarantees under which one can successfully recover the block-sparse representation of a corrupted data and eliminate its sparse corruption. Note also that our results can be easily generalized to the case where the error $\boldsymbol{e}$ has a sparse representation in a dictionary $\boldsymbol{G}$ instead of $\boldsymbol{I}$ by considering the dictionary $\begin{bmatrix} \boldsymbol{B} & \boldsymbol{G} \end{bmatrix}$ in (5.57).

5.6 Experiments with synthetic data

We consider the problem of finding block-sparse representations of signals in dictionaries whose atoms are drawn from a union of disjoint subspaces. We investigate the performance of the two classes of convex programs for various block-sparsity levels.

For simplicity, we assume that all the subspaces have the same dimension d and that the blocks have the same length m. First, we generate random bases $\boldsymbol{A}_i \in \mathbb{R}^{D \times d}$ for n disjoint subspaces $\{\mathcal{S}_i\}_{i=1}^{n}$ in $\mathbb{R}^D$ by orthonormalizing i.i.d. Gaussian matrices where the elements of each matrix are drawn independently from the standard Gaus-

sian distribution.[5] Next, using the subspace bases, we draw $m \in \{d, 2d\}$ random vectors in each subspace to form blocks $\boldsymbol{B}[i] \in \mathbb{R}^{D \times m}$. For a fixed block-sprsity level k, we generate a signal $\boldsymbol{y} \in \mathbb{R}^D$ using a random k-block-sparse vector $\boldsymbol{c}^0 \in \mathbb{R}^{nm}$ where the k nonzero blocks, Λ_k, are chosen uniformly at random from the n blocks and the coefficients in the nonzero blocks are i.i.d. and drawn from the standard Gaussian distribution.

For each class of the convex programs P_{ℓ_q/ℓ_1} and P'_{ℓ_q/ℓ_1} with $q \in \{1, 2, \infty\}$, we measure the following errors. The *reconstruction error* measures how well a signal $\boldsymbol{y}$ can be reconstructed from the blocks of the optimal solution $\boldsymbol{c}^*$ corresponding to the correct support Λ_k and is defined as

$$\text{reconstruction error} = \frac{\|\boldsymbol{y} - \sum_{i \in \Lambda_k} \boldsymbol{B}[i]\boldsymbol{c}^*[i]\|_2}{\|\boldsymbol{y}\|_2}. \tag{5.60}$$

Ideally, if an optimization algorithm is successful in recovering the correct vector in each subspace, i.e., $\boldsymbol{B}[i]\boldsymbol{c}^*[i] = \boldsymbol{B}[i]\boldsymbol{c}^0[i]$ for all i, then the reconstruction error is zero. As we expect that the contribution of the blocks corresponding to $\Lambda_{\widehat{k}}$ to the reconstruction of the given signal be zero, i.e., $\boldsymbol{B}[i]\boldsymbol{c}^*[i] = 0$, we measure the *block-contribution error* as

$$\text{block contribution error} = 1 - \frac{\sum_{i \in \Lambda_k} \|\boldsymbol{B}[i]\boldsymbol{c}^*[i]\|_2}{\sum_{i=1}^{n} \|\boldsymbol{B}[i]\boldsymbol{c}^*[i]\|_2} \in [0, 1]. \tag{5.61}$$

The error is equal to zero when all contributing blocks correspond to Λ_k and it is equal to one when all contributing blocks correspond to $\Lambda_{\widehat{k}}$. For non-redundant blocks, since

[5] In order to ensure that the generated bases correspond to disjoint subspaces, we check that each pair of bases must be full column-rank.

$\boldsymbol{c}^0$ is the unique k-block-sparse vector such that $\boldsymbol{y} = \boldsymbol{B}\boldsymbol{c}^0$, we can also measure the *coefficient recovery error* as

$$\text{coefficient recovery error} = \frac{\|\boldsymbol{c}^* - \boldsymbol{c}^0\|_2}{\|\boldsymbol{c}^0\|_2}. \tag{5.62}$$

We generate $L_1 = 200$ different sets of $n = 40$ blocks in $\mathbb{R}^{100}$ and for each set of n blocks we generate $L_2 = 100$ different block-sparse signals. For a fixed block-sparsity level, we compute the average of the above errors for each optimization program over $L = L_1 \times L_2 = 20,000$ trials.[6]

Figure 5.4 shows the average errors for various block-sparsity levels for non-redundant blocks where $m = d = 4$. As the results show, for a fixed value of q, P'_{ℓ_q/ℓ_1} obtains lower reconstruction, block-contribution, and coefficient recovery errors than P_{ℓ_q/ℓ_1} for all block-sparsity levels. Moreover, while the performance of P_{ℓ_q/ℓ_1} significantly degrades for block-sparsity levels greater than 3, P'_{ℓ_q/ℓ_1} maintains a high performance for a wider range of block-sparsity levels.

Figure 5.5 shows the average errors for various block-sparsity levels for redundant blocks with $m = 2d = 8$. Similar to the previous case, for a fixed q, P'_{ℓ_q/ℓ_1} has a higher performance than P_{ℓ_q/ℓ_1} for all block-sparsity levels. Note that redundancy in the blocks improves the performance of P_{ℓ_q/ℓ_1}. Specifically, compared to the case of non-redundant blocks, the performance of P_{ℓ_q/ℓ_1} degrades at higher sparsity levels.

Notice from the results in Figures 5.4 and 5.5 that P'_{ℓ_q/ℓ_1} performs, in general, better than P_{ℓ_q/ℓ_1} for block-spare recovery in non-redundant blocks, while the gap

[6] In order to solve the convex programs, we use the CVX package which can be downloaded from http://cvxr.com/cvx.

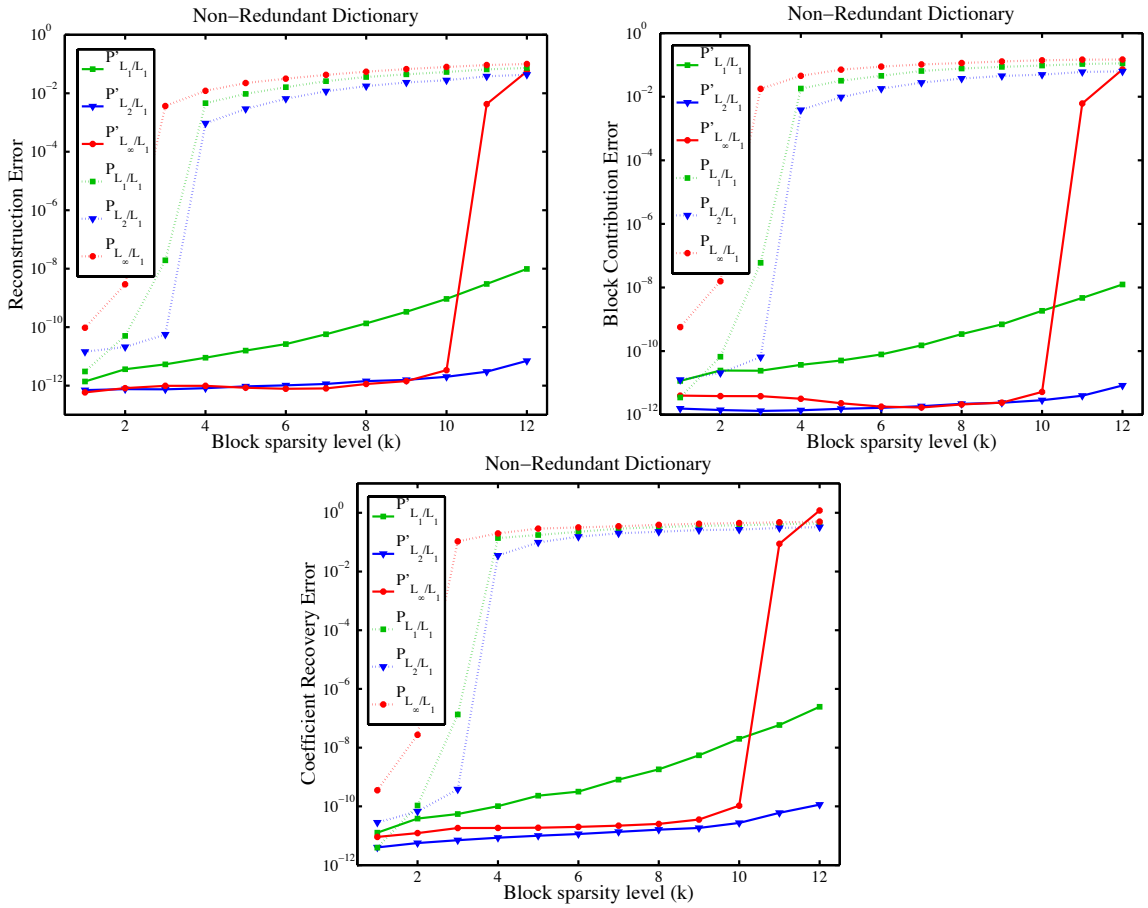


Figure 5.4: Errors of the convex programs on synthetic data with $n = 40$, $D = 100$. Reconstruction error (top left), block-contribution error (top right) and coefficient recovery error (bottom) for non-redundant blocks with $m = d = 4$.

in the performances of the two optimization classes decreases for redundant blocks. This comes from the fact that when the number of vectors in a block $\boldsymbol{B}[i]$ is small, the vectors can be close to a degenerate subspace in range($\boldsymbol{B}[i]$). Hence, the cost $\|\boldsymbol{c}[i]\|_q$ of reconstruction a vector, $\boldsymbol{s}_i \in$ range($\boldsymbol{B}[i]$), that is nearly orthogonal to the direction of the degenerate subspace can be rather high. Thus, the optimization program P_{ℓ_q/ℓ_1} favors selecting vectors from other blocks to reconstruct $\boldsymbol{s}_i$. As the number of vectors in $\boldsymbol{B}[i]$ increases, the probability of obtaining a better distribution,

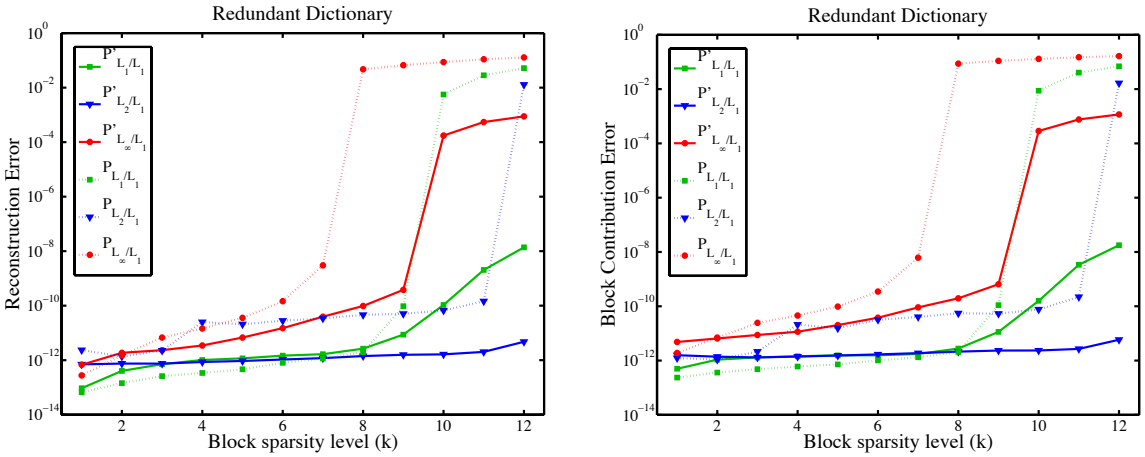


Figure 5.5: Errors of the convex programs on synthetic data with $n = 40$, $D = 100$. Reconstruction error (left) and block-contribution error (right) for redundant blocks with $m = 2d = 8$.

i.e., having vectors in different directions of range($\boldsymbol{B}[i]$), increases. Hence, the cost $\|\boldsymbol{c}[i]\|_q$ of reconstructing the same $\boldsymbol{s}_i$ from the vectors in $\boldsymbol{B}[i]$ decreases. As a result, P_{ℓ_q/ℓ_1} favors selecting vectors from $\boldsymbol{B}[i]$ to reconstruct $\boldsymbol{s}_i$. On the other hand, no matter of how the data are distributed in $\boldsymbol{B}[i]$, the cost $\|\boldsymbol{B}[i]\boldsymbol{c}[i]\|_q$ of reconstructing $\boldsymbol{s}_i$ from the vectors in $\boldsymbol{B}[i]$ remains the same, i.e., $\|\boldsymbol{B}[i]\boldsymbol{c}[i]\|_q = \|\boldsymbol{s}_i\|_q$. Hence, the performance of P'_{ℓ_q/ℓ_1} will be less dependent on the number of vectors in the blocks of the dictionary.

Another observation from the results of Figures 5.4 and 5.5 is that for each class of convex programs, the case of $q = \infty$ either has a lower performance or degrades at lower block-sparsity levels than $q = 1, 2$. In addition, the case of $q = 2$ in general performs better than $q = 1$. This can come from the fact that $q = 1$ promotes sparsity also in each coefficient block for P_{ℓ_q/ℓ_1} and in each block reconstructed vector for P'_{ℓ_q/ℓ_1}, which may result in under-representing the given signal in the desired blocks.

5.7 Experiments with real data

5.7.1 Face recognition: uncorrupted data

In this part, we evaluate the performance of the block-sparse recovery algorithms in the real problem of automatic face recognition. Assume we are given a collection of mn face images of n subjects acquired under the same pose and varying illumination. Under the Lambertian assumption, [6] shows that the face images of each subject live close to a linear subspace of dimension $d = 9$. Thus, the collection of faces of different subjects live close to a union of 9-dimensional subspaces. Let $\boldsymbol{b}_{ij} \in \mathbb{R}^D$ denote the j-th training image for the i-th subject converted into a vector. We denote the collection of m faces for the i-th subject as

$$\boldsymbol{B}[i] \triangleq \begin{bmatrix} \boldsymbol{b}_{i1} & \boldsymbol{b}_{i2} & \cdots & \boldsymbol{b}_{im} \end{bmatrix} \in \mathbb{R}^{D \times m}. \tag{5.63}$$

Thus, the dictionary $\boldsymbol{B}$ consists of the training images of the n subjects. In this dictionary, a new face vector, $\boldsymbol{y} \in \mathbb{R}^D$, which belongs to the i-th subject, can be written as a linear combination of face vectors from the i-th block. However, in reality, a face image is corrupted with cast shadows and specularities. In other words, the columns of $\boldsymbol{B}$ are corrupted by errors and do not perfectly lie in a low-dimensional subspace. Thus, in the optimization programs, instead of the exact equality constraint $\boldsymbol{y} = \boldsymbol{B}\boldsymbol{c}$, we use the constraint $\|\boldsymbol{y} - \boldsymbol{B}\boldsymbol{c}\|_2 \leq \delta$.[7] Following [117], we can find the

[7] In all the experiments of this section, we set $\delta = 0.05$.

subject to which $\boldsymbol{y}$ belongs from

$$\text{identity}(\boldsymbol{y}) = \underset{i}{\text{argmin}} \ \|\boldsymbol{y} - \boldsymbol{B}[i]\boldsymbol{c}^*[i]\|_2. \tag{5.64}$$

We evaluate the performance of each one of the above optimization programs on the Extended Yale B dataset [73], a few images of which are shown in Figure 5.6. The dataset consists of $2,414$ cropped frontal face images of $n = 38$ individuals. For each subject, there are approximately 64 face images of size $192 \times 168 = 32,256$, which are captured under various laboratory-controlled lighting conditions. Since the dimension of the original face vectors is very large, we reduce the dimension of the data using the following methods:

– We use the eigenfaces approach [107] by projecting the face vectors to the first D principal components of the training data covariance matrix.

– We multiply the face vectors by a random projection matrix $\Phi \in \mathbb{R}^{D \times 32,256}$, which has i.i.d. entries drawn from a zero mean Gaussian distribution with variance $\frac{1}{D}$ [4, 117].

– We down-sample the images by a factor r such that the dimension of the down-sampled face vectors is D.

In the experiments, we set $D = 132$. For each subject, we randomly select $m \in \{9, 18, 25, 32\}$ training images, to form the blocks $\boldsymbol{B}[i] \in \mathbb{R}^{D \times m}$ and use the remaining images for testing. For every test image, we solve each class of the convex programs for $q \in \{1, 2\}$ and determine the identity of the test image using (5.64).[8] We compute

[8]Similar to the synthetic experiments, the case of $q = \infty$ has lower performance than other values

the classification rate as the average number of correctly classified test images for which the recovered identity matches the ground-truth. We repeat this experiment $L = 20$ times for random choices of m training data for each subject and compute the mean classification rate among all the trials. We compare our results with the nearest subspace (NS) method [62] as well as the Linear SVM classifier [34].

The recognition results for three dimensionality reduction methods are shown in Figure 5.6. As the results show, the NS and SVM methods have lower performance than methods based on sparse representation. This comes from the fact that the linear SVM assumes that the data in different classes are linearly separable while the face images have a multi-subspace structure, hence are not necessarily separable by a hyperplane. In the case of the NS method, subspaces associated to different classes are close to each other, i.e., have a small principal angle [45]. Since the test images are corrupted by errors, they can be close to the intersection of several subspaces, resulting in incorrect recognition. In addition, using the underlying subspaces ignores the distribution of the data inside the subspaces as opposed to the sparsity-based methods that directly use the training data. On the other hand, for a fixed value of q, the convex program P'_{ℓ_q/ℓ_1} almost always outperforms P_{ℓ_q/ℓ_1}. While the performances of different methods are close for a large number of training data in each class, the difference in their performances becomes evident when the number of data in each class decreases. More specifically, while the performance of all the algorithms degrade

of q, hence we only report the results for $q = 1, 2$.

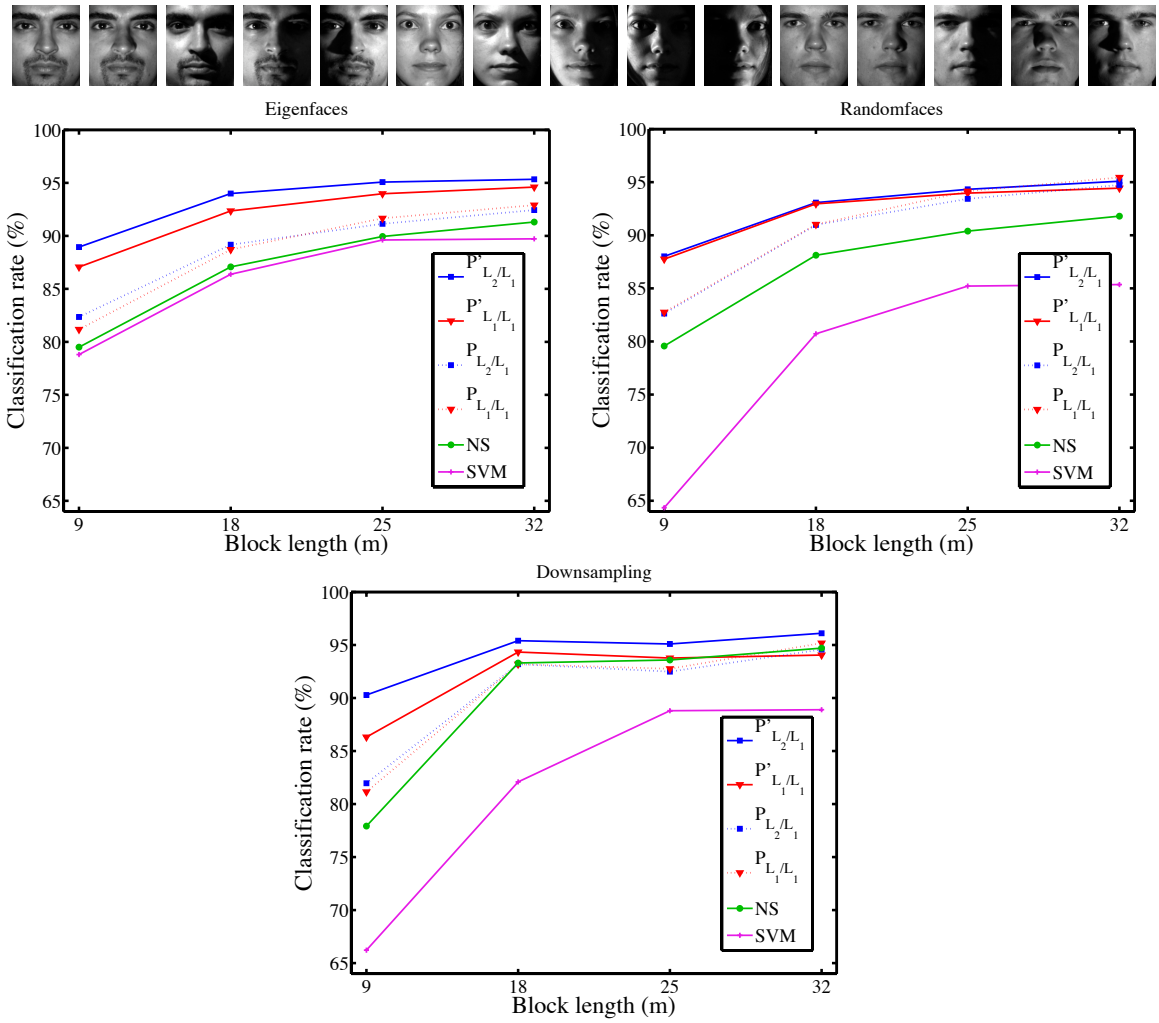


Figure 5.6: First row: sample face images from three subjects in the Extended Yale B dataset. Middle and bottom rows: classification rates for the convex programs on the Extended Yale B database with $n = 38$ and $D = 132$ as a function of the number of training data in each class for using eigen-faces, random projections, and down-sampling.

by decreasing the number of data in each class, the convex programs P'_{ℓ_q/ℓ_1} are more robust to decreasing the number of training data. In other words, when the number of training data in each class is as small as the dimension of the face subspace, i.e., $m = d = 9$, P'_{ℓ_q/ℓ_1} has 5% to 10% higher recognition rate than P_{ℓ_q/ℓ_1}. This

result is similar to the result of synthetic experiments, where we showed that the gap between the performance of the two classes of convex programs is wider for non-redundant blocks than redundant blocks. It is also important to note that the results are independent of the choice of the features, i.e., they follow the same pattern for the three types of features as shown in Figure 5.6. In all of them P'_{ℓ_2/ℓ_1} and P'_{ℓ_1/ℓ_1} achieve the best recognition results.

5.7.2 Face recognition: robustness to random corruptions

In this section, we test the robust versions of the structured sparsity-based algorithms in dealing with random pixel corruption. To that end, we choose images in subset 1 (and 2) of the Extended Yale B database for training and choose images in subset 3 for testing. We downsample the images so that $D = 1,400$. Without corrupting the images, this is not a hard problem and this choice is to isolate the effect of random corruption. Next, we corrupt ρ percentage of randomly chosen pixels in each test image. We replace the values of the chosen pixels by i.i.d. values drawn from a uniform distribution in the range of the image pixel values. We change ρ from 0 to 90 percent and compute the recognition rate. We compare the results of the robust structured sparsity-based classification algorithms $\bar{P}_{\ell_2/\ell_1}$ and $\bar{P}'_{\ell_2/\ell_1}$ with three other methods. First, we use the robust version of the SRC method, $\bar{P}_{\ell_1}$. Next, we use

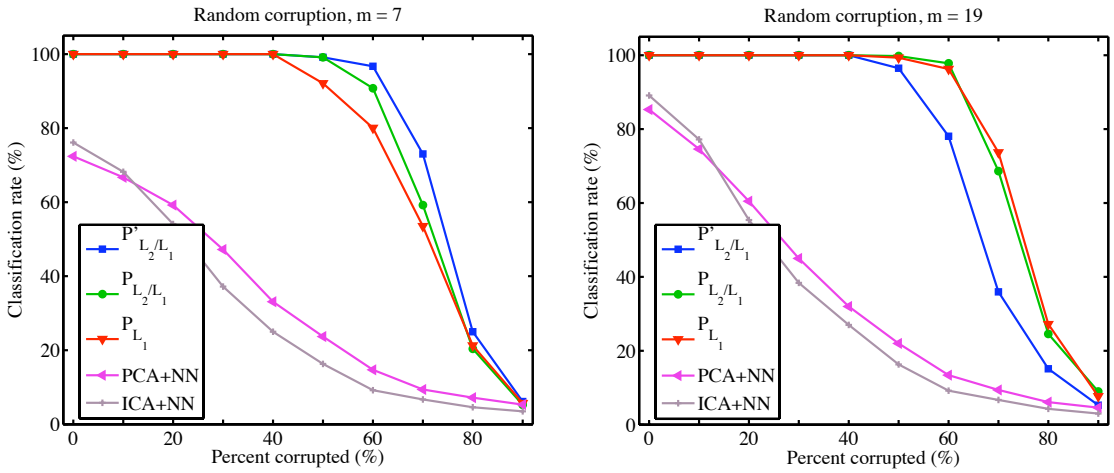


Figure 5.7: Recognition results on the Extended Yale B database as a function of the percentage of corruption.

the basic PCA to project the data into lower dimensions and use the NN classifier. Third, we use the Independent Component Analysis (ICA) architecture I [67] with a NN classifier.[9]

For $m \in \{7, 19\}$ training data in each class, the recognition rates as a function of the percentage of corrupted pixels are shown in Figure 5.7. For both cases, $\bar{P}_{\ell_2/\ell_1}$ and $\bar{P}'_{\ell_2/\ell_1}$ as well as $\bar{P}_{\ell_1}$ achieve almost 100% recognition rate with up to 50% corruption, while the recognition rates of the two other methods drop quickly to less than 30% when we have 50% corrupted pixels. Note that $\bar{P}_{\ell_2/\ell_1}$ and $\bar{P}'_{\ell_2/\ell_1}$ obtain better classification results than $\bar{P}_{\ell_1}$ when the number of training data in each class is small ($m = 7$). However, for $m = 19$, the performances of $\bar{P}_{\ell_2/\ell_1}$ and $\bar{P}_{\ell_1}$ are similar.

[9]For PCA and ICA, we choose the number of basis components over the range $\{100, 200, 300, 400\}$ to give the best test performance.

5.7.3 Face recognition: robustness to random block occlusion

In this section, we test the performance of the structured sparsity-based classification methods in dealing with corrupted data, where corruption appears in a block of a face image instead of being distributed across all image pixels.

We use images in subset 1 (and 2) of the Extended Yale B database for training and use images in subset 3 for testing. We down-sample the images so that $D = 1,400$. In order to examine the robustness of the methods to occlusions we replace a randomly chosen square block of each test image with an unrelated image and change the percentage of occlusion from 0 to 50 percent. Similar to the previous section, we compare the performance of $\bar{P}_{\ell_2/\ell_1}$ and $\bar{P}'_{\ell_2/\ell_1}$ against the SRC method, PCA+NN and ICA+NN. For $m \in \{7, 19\}$ training data in each class, the results are shown in Figure 5.8. Note that the sparse representation based methods achieve almost 100% recognition rate up to 20% occlusion, while the recognition rates of PCA+NN and ICA+NN quickly drop as we increase the the percentage of occlusion. In addition, for $m = 7$, both $\bar{P}_{\ell_2/\ell_1}$ and $\bar{P}'_{\ell_2/\ell_1}$ obtain better recognition rates than $\bar{P}_{\ell_1}$ for all percentages of occlusion.

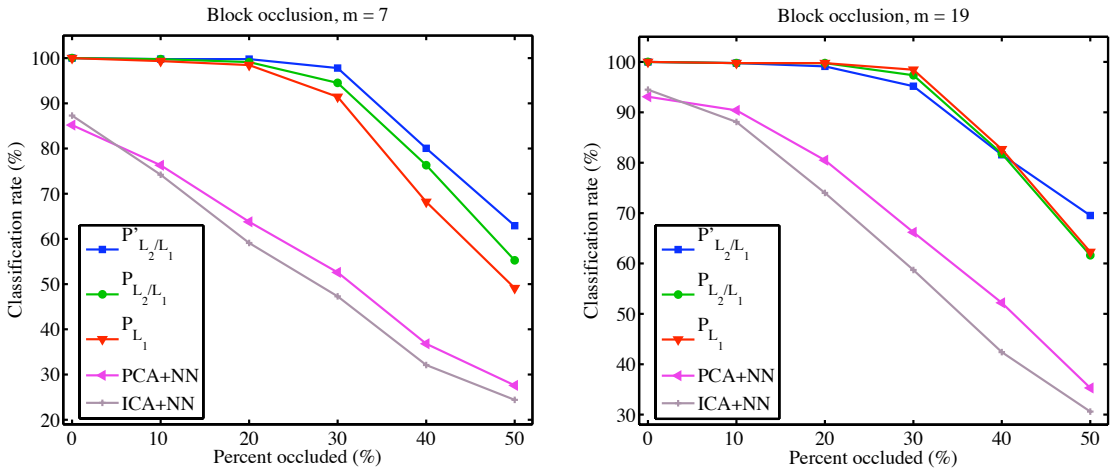


Figure 5.8: Recognition results on the Extended Yale B database as a function of the percentage of block occlusion.

5.7.4 Face recognition: robustness to disguise

In this part, we examine the robustness of the proposed algorithms to real malicious occlusions in images. We use the AR database [79] which consists of face images of $n = 100$ individuals acquired under the same pose with varying illuminations and expressions. Out of the 26 images for each subject, in 6 images the subject is wearing sunglasses, roughly occluding 20% of the image, and in 6 images, the subject is wearing a scarf, occluding nearly 40% of the image. We down-sample the images so that $D = 1,400$. We randomly select $m = 9$ images for each subject as the training data and use the images with sunglasses and scarves as test examples. We evaluate the recognition rates of the structured-sparsity based algorithms as well as the SRC method and the two other algorithms we used in the previous experiments: PCA+NN and ICA+NN. The results are shown in Table 5.1. While $\bar{P}_{\ell_1}$ obtains slightly better recognition rate than $\bar{P}_{\ell_2/\ell_1}$ for images with sunglasses, $\bar{P}_{\ell_2/\ell_1}$ obtains about 25%

Table 5.1: Recognition rates on the AR database for robustness to disguise.

Algorithms	P'_{ℓ_2/ℓ_1}	P_{ℓ_2/ℓ_1}	P_{ℓ_1}	PCA+NN	ICA+NN
Sunglasses	66.5%	80.5%	**84.3%**	57.5%	51.7%
Scarves	41.7%	**59.8%**	35.2%	10.5%	9.2%
All	54.1%	**70.2%**	59.8%	34.0%	30.5%

higher recognition rate than $\bar{P}_{\ell_1}$ for images with scarves.

5.8 Conclusions

We formulated the problem of classification of multi-manifold data as a block-sparse recovery problem using two non-convex optimization programs P_{ℓ_q/ℓ_0} and P'_{ℓ_q/ℓ_0}. To solve them efficiently, we proposed convex relaxations for the two non-convex programs and studied conditions under which they are equivalent to the original non-convex formulations. We showed that the proposed algorithms can be modified to also deal with corrupted data. Our experiments on the face recognition problem showed that the proposed classification methods lead to better recognition results especially when the number of training data in each class is relatively small.

5.9 Appendix

5.9.1 Proof of Proposition 2

Let $\Lambda_k \triangleq \{j_1^*, \ldots, j_k^*\}$ and $i^* \notin \Lambda_k$ be the set of indices for which ζ_k is obtained, i.e.,

$$\zeta_k = \max_{\Lambda_k} \max_{i \notin \Lambda_k} \sum_{j \in \Lambda_k} \mu(\mathcal{S}_i, \mathcal{S}_j) = \sum_{l=1}^{k} \mu(\mathcal{S}_{i^*}, \mathcal{S}_{j_l^*}). \tag{5.65}$$

Denoting the sorted subspace coherences among all pairs of different subspaces by $\mu_S = \mu_1 \geq \mu_2 \geq \cdots$, we have

$$\zeta_k = \sum_{l=1}^{k} \mu(\mathcal{S}_{i^*}, \mathcal{S}_{j_l^*}) \leq u_k = \sum_{l=1}^{k} \mu_l \leq k\mu_S, \tag{5.66}$$

which proves the desired result.

5.9.2 Proof of Proposition 3

We prove this result using contradiction.

($\Longrightarrow$) Assume there exists a $2k$-block-sparse vector $\bar{\boldsymbol{c}} \neq 0$ such that $\bar{\boldsymbol{B}}\,\bar{\boldsymbol{c}} = 0$. We can write $\bar{\boldsymbol{c}}^\top = \begin{bmatrix} \bar{\boldsymbol{c}}_1^\top & \bar{\boldsymbol{c}}_2^\top \end{bmatrix}$ where $\bar{\boldsymbol{c}}_1$ and $\bar{\boldsymbol{c}}_2$ are k-block-sparse vectors. So, we have

$$\bar{\boldsymbol{B}}\,\bar{\boldsymbol{c}} = \begin{bmatrix} \bar{\boldsymbol{B}}_1 & \bar{\boldsymbol{B}}_2 \end{bmatrix} \begin{bmatrix} \bar{\boldsymbol{c}}_1 \\ \bar{\boldsymbol{c}}_2 \end{bmatrix} = 0 \Longrightarrow \bar{\boldsymbol{y}} \triangleq \bar{\boldsymbol{B}}_1 \bar{\boldsymbol{c}}_1 = -\bar{\boldsymbol{B}}_2 \bar{\boldsymbol{c}}_2. \tag{5.67}$$

Thus, there exists a vector $\bar{\boldsymbol{y}}$ that has two k-block-sparse representations in $\boldsymbol{B}$ using different sets of blocks. This contradicts the uniqueness assumption of the proposition.

($\Longleftarrow$) Assume there exists a vector $\boldsymbol{y}$ that has two different k-block-sparse representations using $(\{i_l\}, \{\boldsymbol{s}_{i_l}\}) \neq (\{i'_l\}, \{\boldsymbol{s}'_{i_l}\})$. Since for each block of $\bar{\boldsymbol{B}}$, we have $\text{rank}(\bar{\boldsymbol{B}}[i]) = \text{rank}(\boldsymbol{B}[i])$, there exist $\bar{\boldsymbol{c}}_1$ and $\bar{\boldsymbol{c}}_2$ such that $\boldsymbol{y} = \bar{\boldsymbol{B}}\,\bar{\boldsymbol{c}}_1 = \bar{\boldsymbol{B}}\,\bar{\boldsymbol{c}}_2$, where $\bar{\boldsymbol{c}}_1$ and $\bar{\boldsymbol{c}}_2$ are different k-block-sparse with the indices of their nonzero blocks being $\{i_l\}$ and $\{i'_l\}$, respectively. Also, $\bar{\boldsymbol{B}}[i_l]\bar{\boldsymbol{c}}_1[i_l] = \boldsymbol{s}_{i_l}$ and $\bar{\boldsymbol{B}}[i_l]\bar{\boldsymbol{c}}_2[i_l] = \boldsymbol{s}'_{i_l}$. Thus, we have $\bar{\boldsymbol{B}}\,(\bar{\boldsymbol{c}}_1 - \bar{\boldsymbol{c}}_2) = 0$ that contradicts the assumption of the proposition since $\bar{\boldsymbol{c}}_1 - \bar{\boldsymbol{c}}_2$ is a $2k$-block-sparse vector.

5.9.3 Proof of Corollary 1

We prove the result using contradiction.

($\Longrightarrow$) Assume there exists $\boldsymbol{B}_n \in \mathbb{B}_\tau(\Lambda_n)$ such that $\text{rank}(\boldsymbol{B}_n) < 2k$. So, there exists a $2k$-sparse vector $\boldsymbol{c}_n^\top \triangleq \begin{bmatrix} c_n^1 & \dots & c_n^n \end{bmatrix}$ such that $\boldsymbol{B}_n\boldsymbol{c}_n = \sum_{i=1}^n c_n^i \boldsymbol{s}_i = 0$, where $\boldsymbol{s}_i \in \mathbb{W}_{\tau,i}$ is the i-th column of $\boldsymbol{B}_n$. For each full column-rank submatrix of $\boldsymbol{B}[i]$, denoted by $\bar{\boldsymbol{B}}[i] \in \mathbb{R}^{D\times d_i}$, there exists a unique $\bar{\boldsymbol{c}}[i]$ such that $\bar{\boldsymbol{B}}[i]\bar{\boldsymbol{c}}[i] = c_n^i \boldsymbol{s}_i$. Thus, $\bar{\boldsymbol{B}}\,\bar{\boldsymbol{c}} = 0$, where $\bar{\boldsymbol{B}}$ is defined in (5.22) and $\bar{\boldsymbol{c}}^\top \triangleq \begin{bmatrix} \bar{\boldsymbol{c}}[1]^\top \dots & \bar{\boldsymbol{c}}[n]^\top \end{bmatrix}$ is a $2k$-block-sparse vector. This, contradicts the uniqueness assumption using Proposition 3.

($\Longleftarrow$) Now, assume there exists a signal $\boldsymbol{y}$ that has two different k-block-sparse representations in $\boldsymbol{B}$. From Proposition 3, there exists a $2k$-block-sparse vector $\bar{\boldsymbol{c}} \neq 0$ such that $\bar{\boldsymbol{B}}\,\bar{\boldsymbol{c}} = 0$. We can rewrite $\bar{\boldsymbol{B}}[i]\,\bar{\boldsymbol{c}}[i] = c_n^i \boldsymbol{s}_i$, where $\boldsymbol{s}_i \in \mathbb{W}_{\tau,i}$. Thus, we have $\bar{\boldsymbol{B}}\,\bar{\boldsymbol{c}} = \boldsymbol{B}_n\boldsymbol{c}_n = 0$, where $\boldsymbol{B}_n \triangleq \begin{bmatrix} \boldsymbol{s}_1 & \dots & \boldsymbol{s}_n \end{bmatrix} \in \mathbb{B}_\tau(\Lambda_n)$ and $\boldsymbol{c}_n^\top \triangleq \begin{bmatrix} c_n^1 & \dots & c_n^n \end{bmatrix}$ is a $2k$-sparse vector. This implies $\text{rank}(\boldsymbol{B}_n) < 2k$ that contradicts the assumption.

5.9.4 Proof of Lemma 3

The idea of the proof follows the approach of Theorem 3.5 in [104]. Let $\boldsymbol{E}_k = \begin{bmatrix} \boldsymbol{e}_{i_1} & \cdots & \boldsymbol{e}_{i_k} \end{bmatrix} \in \mathbb{B}_\alpha(\Lambda_k)$ and $\boldsymbol{E}_{\widehat{k}} = \begin{bmatrix} \boldsymbol{e}_{i_{k+1}} & \cdots & \boldsymbol{e}_{i_n} \end{bmatrix}$ where $\|\boldsymbol{e}_{i_l}\|_2 \leq \sqrt{1+\beta}$ for every $i_l \in \Lambda_{\widehat{k}}$. Using matrix norm properties, we have

$$\|(\boldsymbol{E}_k^\top \boldsymbol{E}_k)^{-1} \boldsymbol{E}_k^\top \boldsymbol{E}_{\widehat{k}}\|_{1,1} \leq \|(\boldsymbol{E}_k^\top \boldsymbol{E}_k)^{-1}\|_{1,1} \|\boldsymbol{E}_k^\top \boldsymbol{E}_{\widehat{k}}\|_{1,1}. \tag{5.68}$$

We can write $\boldsymbol{E}_k^\top \boldsymbol{E}_k = \boldsymbol{I}_k + \boldsymbol{D}$, where

$$\boldsymbol{D} \triangleq \begin{bmatrix} \boldsymbol{e}_{i_1}^\top \boldsymbol{e}_{i_1} - 1 & \cdots & \boldsymbol{e}_{i_1}^\top \boldsymbol{e}_{i_k} \\ \vdots & \ddots & \vdots \\ \boldsymbol{e}_{i_k}^\top \boldsymbol{e}_{i_1} & \cdots & \boldsymbol{e}_{i_k}^\top \boldsymbol{e}_{i_k} - 1 \end{bmatrix}. \tag{5.69}$$

Since $\boldsymbol{E}_k \in \mathbb{B}_\alpha(\Lambda_k)$, for any column of $\boldsymbol{E}_k$, we have $\|\boldsymbol{e}_i\|_2^2 \leq 1+\alpha$. Also, for any two columns $\boldsymbol{e}_i$ and $\boldsymbol{e}_j$ of $\boldsymbol{E}_k$ we have

$$|\boldsymbol{e}_i^\top \boldsymbol{e}_j| \leq \|\boldsymbol{e}_i\|_2 \|\boldsymbol{e}_j\|_2 \, \mu(\mathcal{S}_i, \mathcal{S}_j) \leq (1+\alpha)\, \mu(\mathcal{S}_i, \mathcal{S}_j). \tag{5.70}$$

Thus, we can write

$$\|\boldsymbol{D}\|_{1,1} \leq \alpha + (1+\alpha)\, \zeta_{k-1}. \tag{5.71}$$

If $\|\boldsymbol{D}\|_{1,1} < 1$, we can write $(\boldsymbol{E}_k^\top \boldsymbol{E}_k)^{-1} = (\boldsymbol{I}_k + \boldsymbol{D})^{-1} = \sum_{i=0}^{\infty} (-\boldsymbol{D})^k / k!$ from which we obtain

$$\|(\boldsymbol{E}_k^\top \boldsymbol{E}_k)^{-1}\|_{1,1} \leq \sum_{i=0}^{\infty} \frac{\|\boldsymbol{D}\|_{1,1}^k}{k!} = \frac{1}{1 - \|\boldsymbol{D}\|_{1,1}} \leq \frac{1}{1 - [\,\alpha + (1+\alpha)\zeta_{k-1}\,]}. \tag{5.72}$$

On the other hand, $\boldsymbol{E}_k^\top \boldsymbol{E}_{\widehat{k}}$ has the following form

$$\boldsymbol{E}_k^\top \boldsymbol{E}_{\widehat{k}} = \begin{bmatrix} \boldsymbol{e}_{i_1}^\top \boldsymbol{e}_{i_{k+1}} & \cdots & \boldsymbol{e}_{i_1}^\top \boldsymbol{e}_{i_n} \\ \vdots & \ddots & \vdots \\ \boldsymbol{e}_{i_k}^\top \boldsymbol{e}_{i_{k+1}} & \cdots & \boldsymbol{e}_{i_k}^\top \boldsymbol{e}_{i_n} \end{bmatrix}. \tag{5.73}$$

Since for each column $\boldsymbol{e}_i$ of the matrix $\boldsymbol{E}_k$ we have $\|\boldsymbol{e}_i\|_2^2 \leq 1+\alpha$ and for each column $\boldsymbol{e}_j$ of the matrix $\boldsymbol{E}_{\widehat{k}}$ we have $\|\boldsymbol{e}_j\|_2^2 \leq 1+\beta$, we obtain

$$\|\boldsymbol{E}_k^\top \boldsymbol{E}_{\widehat{k}}\|_{1,1} \leq \sqrt{(1+\alpha)(1+\beta)}\, \zeta_k. \tag{5.74}$$

Finally, substituting (5.72) and (5.74) into (5.68), we get

$$\|(\boldsymbol{E}_k^\top \boldsymbol{E}_k)^{-1} \boldsymbol{E}_k^\top \boldsymbol{E}_{\widehat{k}}\|_{1,1} \leq \|(\boldsymbol{E}_k^\top \boldsymbol{E}_k)^{-1}\|_{1,1} \|\boldsymbol{E}_k^\top \boldsymbol{E}_{\widehat{k}}\|_{1,1} \leq \frac{\sqrt{(1+\alpha)(1+\beta)}\, \zeta_k}{1 - [\, \alpha + (1+\alpha)\zeta_{k-1}\,]}. \tag{5.75}$$

5.9.5 Proof of Proposition 5

Fix a set $\Lambda_k = \{i_1, \ldots, i_k\}$ of k indices from $\{1, \ldots, n\}$ and denote by $\Lambda_{\widehat{k}} = \{i_{k+1}, \ldots, i_n\}$ the set of the remaining indices. For a signal $\mathbf{x}$ in the intersection of $\oplus_{i \in \Lambda_k} \mathcal{S}_i$ and $\oplus_{i \in \Lambda_{\widehat{k}}} \mathcal{S}_i$, let $\breve{\boldsymbol{c}}^*$ be the solution of the optimization program (5.47). We can write

$$\mathbf{x} = \sum_{i \in \Lambda_k} \boldsymbol{B}[i] \boldsymbol{c}^*[i] = \boldsymbol{B}_k \boldsymbol{a}_k, \tag{5.76}$$

where $\boldsymbol{B}_k \triangleq \begin{bmatrix} \boldsymbol{s}_{i_1} & \ldots & \boldsymbol{s}_{i_k} \end{bmatrix}$ and $\boldsymbol{a}_k \triangleq \begin{bmatrix} a_{i_1} & \ldots & a_{i_k} \end{bmatrix}^\top$ are defined as follows. For every $i \in \Lambda_k$, if $\breve{\boldsymbol{c}}^*[i] \neq 0$ and $\boldsymbol{B}[i]\breve{\boldsymbol{c}}^*[i] \neq 0$, define

$$\boldsymbol{s}_i \triangleq \frac{\boldsymbol{B}[i]\breve{\boldsymbol{c}}^*[i]}{\|\boldsymbol{B}[i]\breve{\boldsymbol{c}}^*[i]\|_q}, \quad a_i \triangleq \|\boldsymbol{B}[i]\breve{\boldsymbol{c}}^*[i]\|_q. \tag{5.77}$$

Otherwise, let $\boldsymbol{s}_i$ be an arbitrary vector in $\mathcal{S}_i$ of unit Euclidean norm and $a_i = 0$. According to Definition 9, we have $\boldsymbol{B}_k \in \mathbb{B}_{\epsilon'_q}(\Lambda_k)$.

Now, let $\widehat{\boldsymbol{c}}^*$ be the solution of the optimization program (5.48). We can write

$$\mathbf{x} = \sum_{i \in \Lambda_{\widehat{k}}} \boldsymbol{B}[i]\widehat{\boldsymbol{c}}^*[i] = \boldsymbol{B}_{\widehat{k}}\boldsymbol{a}_{\widehat{k}}, \tag{5.78}$$

where $\boldsymbol{B}_{\widehat{k}} \triangleq \begin{bmatrix} \boldsymbol{s}_{i_{k+1}} & \dots & \boldsymbol{s}_{i_n} \end{bmatrix}$ and $\boldsymbol{a}_{\widehat{k}} \triangleq \begin{bmatrix} a_{i_{k+1}} & \dots & a_{i_n} \end{bmatrix}^\top$ are defined in the following way. For every $i \in \Lambda_{\widehat{k}}$, if $\widehat{\boldsymbol{c}}^*[i] \neq 0$ and $\boldsymbol{B}[i]\widehat{\boldsymbol{c}}^*[i] \neq 0$, define

$$\boldsymbol{s}_i \triangleq \frac{\boldsymbol{B}[i]\widehat{\boldsymbol{c}}^*[i]}{\|\boldsymbol{B}[i]\widehat{\boldsymbol{c}}^*[i]\|_q}, \quad a_i \triangleq \|\boldsymbol{B}[i]\widehat{\boldsymbol{c}}^*[i]\|_q. \tag{5.79}$$

Otherwise, let $\boldsymbol{s}_i$ be an arbitrary vector in $\mathcal{S}_i$ of unit Euclidean norm and $a_i = 0$. Note that from Definition 9, we have $\boldsymbol{B}_{\widehat{k}} \in \mathbb{B}_{\epsilon'_q}(\Lambda_{\widehat{k}})$.

Since $\boldsymbol{B}_k \in \mathbb{B}_{\epsilon'_q}(\Lambda_k)$, assuming $\epsilon'_q \in [0, 1)$, the matrix $\boldsymbol{B}_k$ is full column-rank from Corollary 1. Hence, we have $\boldsymbol{a}_k = (\boldsymbol{B}_k^\top \boldsymbol{B}_k)^{-1}\boldsymbol{B}_k^\top \boldsymbol{y}$ and consequently,

$$\|\boldsymbol{a}_k\|_1 = \|(\boldsymbol{B}_k^\top \boldsymbol{B}_k)^{-1}\boldsymbol{B}_k^\top \mathbf{x}\|_1. \tag{5.80}$$

Substituting $\mathbf{x}$ from (5.78) in the above equation, we obtain

$$\|\boldsymbol{a}_k\|_1 = \|(\boldsymbol{B}_k^\top \boldsymbol{B}_k)^{-1}\boldsymbol{B}_k^\top \boldsymbol{B}_{\widehat{k}}\boldsymbol{a}_{\widehat{k}}\|_1 \leq \|(\boldsymbol{B}_k^\top \boldsymbol{B}_k)^{-1}\boldsymbol{B}_k^\top \boldsymbol{B}_{\widehat{k}}\|_{1,1}\|\boldsymbol{a}_{\widehat{k}}\|_1. \tag{5.81}$$

Using Lemma 3 with $\alpha = \epsilon'_q$ and $\beta = \epsilon'_q$, we have

$$\|(\boldsymbol{B}_k^\top \boldsymbol{B}_k)^{-1}\boldsymbol{B}_k^\top \boldsymbol{B}_{\widehat{k}}\|_{1,1} \leq \frac{(1+\epsilon'_q)\zeta_k}{1 - [\epsilon'_q + (1+\epsilon'_q)\zeta_{k-1}]}. \tag{5.82}$$

Thus, if the right hand side of the above equation is strictly less than one, i.e., the sufficient condition of the proposition is satisfied, then from (5.81) we have $\|\boldsymbol{a}_k\|_1 <$

$\|\boldsymbol{a}_{\widehat{k}}\|_1$. Finally, using the definitions of $\boldsymbol{a}_k$ and $\boldsymbol{a}_{\widehat{k}}$, we obtain

$$\sum_{i\in\Lambda_k} \|\boldsymbol{B}[i]\breve{\boldsymbol{c}}^*[i]\|_q = \|\boldsymbol{a}_k\|_1 < \|\boldsymbol{a}_{\widehat{k}}\|_1 = \sum_{i\in\Lambda_{\widehat{k}}} \|\boldsymbol{B}[i]\widehat{\boldsymbol{c}}^*[i]\|_q, \tag{5.83}$$

which implies that the condition of Theorem 5 is satisfied. Thus, P'_{ℓ_q/ℓ_1} recovers the unique block-sparse representation of the given signal.

Chapter 6

Conclusions

In this thesis, we considered the machine learning of the multi-manifold data. We considered the three fundamental tasks of clustering, dimensionality reduction and classification. We proposed algorithms based on sparse representation techniques to address these problems efficiently.

In the case of clustering and dimensionality reduction, we showed that the proposed algorithms can effectively address the challenges of these tasks and solve the issues of existing algorithms, such as eliminating the need to know the dimensions of the manifolds a priori and dealing with manifolds that are spatially close or even intersect. As we showed through extensive experiments, our proposed algorithms have significantly improved the state-of-the-art results in segmentation of different motions in videos and clustering of human face images. For subspaces, we also showed that the proposed clustering algorithm has theoretical guarantees and works under moderate

conditions on the arrangement of the subspaces and the data distribution.

In the case of the multi-manifold data classification, by exploiting the structure of the manifolds, we showed that we can have more effective classification algorithms to deal with a small number of training samples as well as the noise and corruption in the data. We investigated the theoretical guarantees of the studied algorithms and, through extensive experiments, we verified their efficacy in the real-world problem of face recognition.

Bibliography

[1] E. Amaldi and V. Kann, "On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems," *Theoretical Computer Science*, vol. 209, pp. 237–260, 1998.

[2] E. Arias-Castro, G. Chen, and G. Lerman, "Spectral clustering based on local linear approximations," *Electron. J. Statist.*, vol. 5, pp. 1537–1587, 2011.

[3] F. Bach, "Consistency of the group lasso and multiple kernel learning," *Journal of Machine Learning Research*, vol. 9, pp. 1179–1225, 2008.

[4] 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.

[5] D. Barbará and P. Chen, "Using the fractal dimension to cluster datasets," in *KDD '00: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2000, pp. 260–264.

BIBLIOGRAPHY

[6] R. Basri and D. Jacobs, "Lambertian reflection and linear subspaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 3, pp. 218–233, 2003.

[7] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Neural Information Processing Systems*, 2002, pp. 585–591.

[8] ——, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[9] R. E. Bellman, *Dynamic programming*. Princeton University Press, 1957.

[10] D. Bertsekas, A. Nedic, and A. Ozdaglar, *Convex Analysis and Optimization*. Athena Scientific, 2003.

[11] P. Boufounos, G. Kutyniok, and H. Rauhut, "Sparse recovery from combined fusion frame measurements," *IEEE Trans. Inform. Theory*, to appear.

[12] T. Boult and L. Brown, "Factorization-based segmentation of motions," in *IEEE Workshop on Motion Understanding*, 1991, pp. 179–186.

[13] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2010.

BIBLIOGRAPHY

[14] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[15] P. S. Bradley and O. L. Mangasarian, "k-plane clustering," *J. of Global Optimization*, vol. 16, no. 1, pp. 23–32, 2000.

[16] T. Brox and J. Malik, "Object segmentation by long term analysis of point trajectories," *European Conf. on Computer Vision*, 2010.

[17] E. Candès, "The restricted isometry property and its implications for compressed sensing," *Comptes Rendus Mathematique*, vol. 346, no. 9-10, pp. 589–592, 2008.

[18] E. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis," *Journal of the ACM*, vol. 58, 2011.

[19] E. Candès and B. Recht, "Exact matrix completion via convex optimization," *Foundations of Computational Mathematics*, vol. 9, pp. 717–772, 2008.

[20] E. Candès and M. Wakin, "An introduction to compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 21–30, Mar. 2008.

[21] E. Candès and T. Tao, "Decoding by linear programming," *IEEE Trans. on Information Theory*, vol. 51, no. 12, pp. 4203–4215, 2005.

[22] G. Chen and G. Lerman, "Spectral curvature clustering (SCC)," *International Journal of Computer Vision*, vol. 81, no. 3, pp. 317–330, 2009.

[23] J. Chen and X. Huo, "Theoretical results on sparse representation of multiple-measurement vectors," *IEEE Trans. on Signal Processing*, vol. 54, no. 12, pp. 4634–4643, Dec. 2006.

[24] R. Coifman and S. Lafon, "Diffusion maps," *Applied and Computational Harmonic Analysis*, vol. 21, no. 1, pp. 5–30, 2006.

[25] J. Costeira and T. Kanade, "A multibody factorization method for independently moving objects." *Int. Journal of Computer Vision*, vol. 29, no. 3, 1998.

[26] S. Cotter, B. Rao, K. Engan, and K. Kreutz-Delgado, "Sparse solutions to linear inverse problems with multiple measurement vectors," *IEEE Trans. on Signal Processing*, vol. 7, no. 7, pp. 2477–2488, Jul. 2005.

[27] D. Donoho, "High-dimensional centrally symmetric polytopes with neighborliness proportional to dimension," *Discrete and Computational Geometry*, vol. 102, no. 27, pp. 617–652, 2006.

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

[29] D. Donoho and X. Huo, "Uncertainty principles and ideal atomic decomposition," *IEEE Trans. Information Theory*, vol. 47, no. 7, pp. 2845–2862, Nov. 2001.

[30] D. L. Donoho and M. Elad, "Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ_1 minimization," *PNAS*, vol. 100, no. 5, pp. 2197–2202, 2003.

[31] D. L. Donoho, M. Elad, and V. N. Temlyakov, "Stable recovery of sparse overcomplete representations in the presence of noise," *IEEE Trans. on Information Theory*, vol. 52, no. 1, pp. 6–18, Jan. 2006.

[32] D. L. Donoho, "Neighborly polytopes and sparse solution of underdetermined linear equations," *Technical Report, Stanford University*, 2005.

[33] ——, "For most large underdetermined systems of linear equations the minimal ℓ^1-norm solution is also the sparsest solution," *Communications on Pure and Applied Mathematics*, vol. 59, no. 6, pp. 797–829, 2006.

[34] R. Duda, P. Hart, and D. Stork, *Pattern Classification*. Wiley-Interscience, October 2004.

[35] M. Elad, M. A. T. Figueiredo, and Y. Ma, "On the role of sparse and redundant representations in image processing," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 972–982, 2010.

[36] Y. C. Eldar, P. Kuppinger, and H. Bolcskei, "Compressed sensing of block-sparse signals: Uncertainty relations and efficient recovery," *IEEE Trans. Signal Processing*, vol. 58, no. 6, pp. 3042–3054, June 2010.

[37] Y. C. Eldar and M. Mishali, "Robust recovery of signals from a structured union of subspaces," *IEEE Trans. Inform. Theory*, vol. 55, no. 11, pp. 5302–5316, 2009.

[38] E. Elhamifar, G. Sapiro, and R. Vidal, "Finding exemplars from pairwise dissimilarities via simultaneous sparse recovery," in *Neural Information Processing Systems*, 2012.

[39] ——, "See all by looking at a few: Sparse modeling for finding representative objects," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[40] E. Elhamifar and R. Vidal, "Sparse subspace clustering," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[41] ——, "Clustering disjoint subspaces via sparse representation," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2010.

[42] ——, "Robust classification using structured sparse representation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[43] ——, "Sparse manifold clustering and embedding," in *Neural Information Processing Systems*, 2011.

[44] ——, "Block-sparse recovery via convex optimization," *IEEE Transactions on Signal Processing*, 2012.

[45] ——, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, submitted., Available: http://arxiv.org/abs/1203.1005.

[46] P. Favaro, R. Vidal, and A. Ravichandran, "A closed form solution to robust subspace estimation and clustering," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[47] M. A. Fischler and R. C. Bolles, "RANSAC random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 26, pp. 381–395, 1981.

[48] S. Foucart, "A note on guaranteed sparse recovery via ℓ_1-minimization," *Appl. Comput. Harmon. Anal.*, 2010 (To appear).

[49] D. Gabay and B. Mercier, "A dual algorithm for the solution of nonlinear variational problems via finite-element approximations," *Comp. Math. Appl.*, vol. 2, pp. 17–40, 1976.

[50] A. Ganesh, Z. Zhou, and Y. Ma, "Separation of a subspace-sparse signal: Algorithms and conditions," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Apr. 2009.

[51] C. W. Gear, "Multibody grouping from motion images," *Int. Journal of Computer Vision*, vol. 29, no. 2, pp. 133–150, 1998.

[52] J. Gemmeke and B. Cranen, "Noise robust digit recognition using sparse representations," in *ISCA ITRW*, 2008.

[53] ——, "Using sparse representations for missing data imputation in noise robust speech recognition," in *EUSIPCO*, 2008.

[54] A. Gionis, A. Hinneburg, S. Papadimitriou, and P. Tsaparas, "Dimension induced clustering," in *KDD '05: Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, 2005, pp. 51–60.

[55] A. Goh and R. Vidal, "Segmenting motions of different types by unsupervised manifold clustering," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[56] ——, "Clustering and dimensionality reduction on Riemannian manifolds," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[57] H. Golub and C. V. Loan, *Matrix Computations*, 2nd ed. Johns Hopkins University Press, 1996.

[58] R. Gribonval and M. Nielsen, "Sparse representations in unions of bases," *IEEE Transactions on Information Theory*, vol. 49, no. 12, pp. 3320–3325, 2003.

[59] A. Gruber and Y. Weiss, "Multibody factorization with uncertainty and missing data using the EM algorithm," in *IEEE Conf. on Computer Vision and Pattern Recognition*, vol. I, 2004, pp. 707–714.

[60] G. Haro, G. Randall, and G. Sapiro, "Translated poisson mixture model for stratification learning," *International Journal of Computer Vision*, 2008.

[61] T. Hastie and P. Simard, "Metrics and models for handwritten character recognition," *Statistical Science*, vol. 13, no. 1, pp. 54–65, 1998.

[62] J. Ho, M. H. Yang, J. Lim, K. Lee, and D. Kriegman, "Clustering appearances of objects under varying illumination conditions." in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2003.

[63] W. Hong, J. Wright, K. Huang, and Y. Ma, "Multi-scale hybrid linear models for lossy image representation," *IEEE Trans. on Image Processing*, vol. 15, no. 12, pp. 3655–3671, 2006.

[64] R. Jenatton, J. Y. Audibert, and F. Bach, "Structured variable selection with sparsity-inducing norms," *Journal of Machine Learning Research*, vol. 12, pp. 2777–2824, 2011.

[65] I. Jolliffe, *Principal Component Analysis*, 2nd ed. Springer-Verlag, 2002.

[66] K. Kanatani, "Motion segmentation by subspace separation and model selection," in *IEEE Int. Conf. on Computer Vision*, vol. 2, 2001, pp. 586–591.

[67] J. Kim, J. Choi, J. Yi, and M. Turk, "Effective representation using ica for face recognition robust to local distortion and partial occlusion," *PAMI*, vol. 27, no. 12, pp. 1977–1981, 2005.

[68] S. J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky, "An interior-point method for large-scale l1-regularized least squares," *IEEE Journal on Selected Topics in Signal Processing*, vol. 1, no. 4, pp. 606–617, 2007.

[69] M. Knott and D. Bartholomew, *Latent variable models and factor analysis*. London: Edward Arnold, 1999.

[70] M. J. Lai and Y. Liu, "The null space property for sparse recovery from multiple measurement vectors," *Applied and Computational Harmonic Analysis*, vol. 30, no. 3, pp. 402–406, 2011.

[71] F. Lauer and C. Schnörr, "Spectral clustering of linear subspaces for motion segmentation," in *IEEE International Conference on Computer Vision*, 2009.

[72] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, 1998, pp. 2278 – 2324.

[73] K.-C. Lee, J. Ho, and D. Kriegman, "Acquiring linear subspaces for face recognition under variable lighting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 5, pp. 684–698, 2005.

[74] E. Levina and P. J. Bickel, "Maximum likelihood estimation of intrinsic dimension." in *NIPS*, 2004.

[75] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma, "Robust recovery of subspace

structures by low-rank representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012.

[76] G. Liu, Z. Lin, and Y. Yu, "Robust subspace segmentation by low-rank representation," in *International Conference on Machine Learning*, 2010.

[77] G. Liu and S. Yan, "Latent low-rank representation for subspace segmentation and feature extraction," *International Conference on Computer Vision*, 2011.

[78] Y. Ma, A. Yang, H. Derksen, and R. Fossum, "Estimation of subspace arrangements with applications in modeling and segmenting mixed data," *SIAM Review*, 2008.

[79] A. Martinez and R. Benavente, "The ar face database," *CVC Technical Report 24*, 1998.

[80] M. Mishali and Y. C. Eldar, "Blind multi-band signal reconstruction: Compressed sensing for analog signals," *IEEE Transactions on Signal Processing*, vol. 57, no. 3, pp. 993–1009, Mar. 2009.

[81] ——, "From theory to practice: Sub-nyquist sampling of sparse wideband analog signals," *IEEE J. Sel. Topics Signal Process.*, vol. 4, no. 2, pp. 375–391, April 2010.

[82] P. Mordohai and G. G. Medioni, "Unsupervised dimensionality estimation and

manifold learning in high-dimensional spaces by tensor voting." in *International Joint Conference on Artificial Intelligence*, 2005, pp. 798–803.

[83] B. Nasihatkon and R. Hartley, "Graph connectivity in sparse subspace clustering," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[84] A. Ng, Y. Weiss, and M. Jordan, "On spectral clustering: analysis and an algorithm," in *Neural Information Processing Systems*, 2001, pp. 849–856.

[85] G. Obozinski, M. J. Wainwright, and M. I. Jordan, "Support union recovery in high-dimensional multivariate regression," *Annals of Statistics*, vol. 39, no. 1, pp. 1–17, 2011.

[86] F. Parvaresh, H. Vikalo, S. Misra, and B. Hassibi, "Recovering sparse signals using sparse measurement matrices in compressed dna microarrays," *IEEE Journal of Selected Topics in Signal Processing*, vol. 2, no. 3, pp. 275–285, Jun. 2008.

[87] M. Polito and P. Perona, "Grouping and dimensionality reduction by locally linear embedding," in *Neural Information Processing Systems*, 2002.

[88] S. Rao, R. Tron, Y. Ma, and R. Vidal, "Motion segmentation via robust subspace separation in the presence of outlying, incomplete, or corrupted trajectories," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[89] S. Rao, R. Tron, R. Vidal, and Y. Ma, "Motion segmentation in the presence of

outlying, incomplete, or corrupted trajectories," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009.

[90] B. Recht, M. Fazel, and P. Parrilo, "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization," *SIAM Review*, vol. 52, no. 3, pp. 471–501, 2010.

[91] S. Roweis and L. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[92] B. Shaw and T. Jebara, "Minimum volume embedding," in *Artificial Intelligence and Statistics*, 2007.

[93] ——, "Structure preserving embedding," in *International Conference on Machine Learning*, 2009.

[94] M. Soltanolkotabi and E. J. Candes, "A geometric analysis of subspace clustering with outliers," *Annals of Statistics*, 2012.

[95] R. Souvenir and R. Pless, "Manifold clustering," in *IEEE International Conference on Computer Vision*, vol. I, 2005, pp. 648–653.

[96] M. Stojnic, F. Parvaresh, and B. Hassibi, "On the reconstruction of block-sparse signals with and optimal number of measurements," *IEEE Trans. Signal Processing*, vol. 57, no. 8, pp. 3075–3085, Aug. 2009.

[97] Y. Sugaya and K. Kanatani, "Geometric structure of degeneracy for multi-body motion segmentation," in *Workshop on Statistical Methods in Video Processing*, 2004.

[98] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[99] R. Tibshirani, "Regression shrinkage and selection via the LASSO," *Journal of the Royal Statistical Society B*, vol. 58, no. 1, pp. 267–288, 1996.

[100] M. Tipping and C. Bishop, "Mixtures of probabilistic principal component analyzers," *Neural Computation*, vol. 11, no. 2, pp. 443–482, 1999.

[101] ——, "Probabilistic principal component analysis," *Journal of the Royal Statistical Society*, vol. 61, no. 3, pp. 611–622, 1999.

[102] C. Tomasi and T. Kanade, "Shape and motion from image streams under orthography," *International Journal of Computer Vision*, vol. 9, no. 2, pp. 137–154, 1992.

[103] R. Tron and R. Vidal, "A benchmark for the comparison of 3-D motion segmentation algorithms," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[104] J. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Trans. Information Theory*, vol. 50, no. 10, pp. 2231–2242, Oct. 2004.

[105] J. A. Tropp., "Algorithms for simultaneous sparse approximation. part ii: Convex relaxation," *Signal Processing, special issue "Sparse approximations in signal and image processing"*, vol. 86, pp. 589–602, 2006.

[106] P. Tseng, "Nearest q-flat to m points," *Journal of Optimization Theory and Applications*, vol. 105, no. 1, pp. 249–252, 2000.

[107] M. Turk and A. Pentland, "Face recognition using eigenfaces," in *IEEE Conference on Computer Vision and Pattern Recognition*, 1991, pp. 586–591.

[108] E. van den Berg and M. Friedlander, "Theoretical and empirical results for recovery from multiple measurements," *IEEE Trans. Information Theory*, vol. 56, no. 5, pp. 2516–2527, 2010.

[109] R. Vidal, "Subspace clustering," *Signal Processing Magazine*, vol. 28, no. 2, pp. 52–68, 2011.

[110] R. Vidal and R. Hartley, "Motion segmentation with missing data by Power-Factorization and Generalized PCA," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. II, 2004, pp. 310–316.

[111] 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. 1–15, 2005.

[112] R. Vidal, R. Tron, and R. Hartley, "Multiframe motion segmentation with missing data using PowerFactorization and GPCA," *International Journal of Computer Vision*, vol. 79, no. 1, pp. 85–105, 2008.

[113] U. von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, 2007.

[114] K. Q. Weinberger and L. Saul, "Unsupervised learning of image manifolds by semidefinite programming," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2004, pp. 988–955.

[115] J. Wright and Y. Ma, "Dense error correction via ℓ_1-minimization," *IEEE Transactions on Information Theory*, vol. 56, no. 7, pp. 3540–3560, 2010.

[116] J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. S. Huang, and S. Yan, "Sparse representation for computer vision and pattern recognitiong," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1031–1044, 2010.

[117] 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, Feb. 2009.

[118] J. Yan and M. Pollefeys, "A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate," in *European Conf. on Computer Vision*, 2006, pp. 94–106.

[119] A. Yang, R. Jafari, P. Kuryloski, S. Iyengar, S. Sastry, and R. Bajcsy, "Distributed segmentation and classification of human actions using a wearable sensor network," *CVPR Workshop on Human Communicative Behavior Analysis*, 2008.

[120] A. Yang, J. Wright, Y. Ma, and S. Sastry, "Unsupervised segmentation of natural images via lossy data compression," *Computer Vision and Image Understanding*, vol. 110, no. 2, pp. 212–225, 2008.

[121] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *Journal of the Royal Statistical Society: Series B*, vol. 68, no. 1, pp. 49–67, 2006.

[122] L. Zelnik-Manor and M. Irani, "Degeneracies, dependencies and their implications in multi-body and multi-sequence factorization," in *IEEE Conf. on Computer Vision and Pattern Recognition*, vol. 2, 2003, pp. 287–293.

[123] T. Zhang, A. Szlam, and G. Lerman, "Median k-flats for hybrid linear modeling with many outliers," in *Workshop on Subspace Methods*, 2009.

[124] T. Zhang, A. Szlam, Y. Wang, and G. Lerman, "Hybrid linear modeling via local

best-fit flats," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 1927–1934.

[125] Z. Zhang and H. Zha, "Principal manifolds and nonlinear dimensionality reduction via tangent space alignment," *SIAM J. Sci. Comput.*, vol. 26, no. 1, pp. 313–338, 2005.

[126] B. Zhao, G. Rocha, and B. Yu, "The composite absolute penalties family for grouped and hierarchical selection," *The Annals of Statistics*, vol. 37, no. 6, pp. 3468–3497, 2009.

Vita

Ehsan Elhamifar received his B.S. degree in Biomedical Engineering from Amirkabir University of Technology Iran in 2004, awarded as the best undergraduate student. He obtained his M.S. degree in Electrical Engineering from Sharif University of Technology Iran in 2006. He also obtained an M.S.E. degree in Applied Mathematics and Statistics from The Johns Hopkins University in 2011. His research interests include machine learning and computer vision, in particular, sparse signal representation, high-dimensional data analysis, as well as clustering, dimensionality reduction, and classification of multi-manifold data.