Sparse Methods for Learning Multiple Subspaces from Large-scale, Corrupted and Imbalanced Data

by

Chong You

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

Baltimore, Maryland

October, 2018

Abstract

In many practical applications in machine learning, computer vision, data mining and information retrieval one is confronted with datasets whose intrinsic dimension is much smaller than the dimension of the ambient space. This has given rise to the challenge of effectively learning multiple low-dimensional subspaces from such data. Multi-subspace learning methods based on sparse representation, such as sparse representation based classification (SRC) and sparse subspace clustering (SSC) have become very popular due to their conceptual simplicity and empirical success. However, there have been very limited theoretical explanations for the correctness of such approaches in the literature. Moreover, the applicability of existing algorithms to real world datasets is limited due to their high computational and memory complexity, sensitivity to data corruptions as well as sensitivity to imbalanced data distributions.

This thesis attempts to advance our theoretical understanding of sparse representation based multi-subspace learning methods, as well as develop new algorithms for handling large-scale, corrupted and imbalanced data. The first

contribution of this thesis is a theoretical analysis of the correctness of such methods. In our geometric and randomized analysis, we answer important theoretical questions such as the effect of subspace arrangement, data distribution, subspace dimension, data sampling density, and so on.

The second contribution of this thesis is the development of practical subspace clustering algorithms that are able to deal with large-scale, corrupted and imbalanced datasets. To deal with large-scale data, we study different approaches based on active support and divide-and-conquer ideas, and show that these approaches offer a good tradeoff between high accuracy and low running time. To deal with corrupted data, we construct a Markov chain whose stationary distribution can be used to separate between inliers and outliers. Finally, we propose an efficient exemplar selection and subspace clustering method that outperforms traditional methods on imbalanced data.

Primary Reader and Advisor: René Vidal

Secondary Reader: Daniel P. Robinson

Acknowledgments

I am very grateful to my advisor, Professor René Vidal, for introducing me to the fantastic world of machine learning, and for his guidance along the road of my PhD study. René's profound insights and broad vision in research have been a role model to me and have been the incentive to me to pursue my research. I am also thankful to him for always being patient and for giving me the freedom to explore different research ideas and conduct research through trial and error, which not only helped me to shape my independent research interests but also made the research process enjoyable.

I am also grateful to Professor Daniel P. Robinson for always being supportive in my research. I have learned a lot from Daniel on the importance of precise and rigorous writing in scientific papers and reports. I would also like to express my gratitude to Prof. Trac Tran, Prof. Vishal Patel, Prof. Enrique Mallada and Prof. Yi Ma for serving in my thesis proposal and dissertation committees, and to Prof. Gregory Hager, Prof. Sridevi Sarma for serving in my Graduate Board Oral committee.

ACKNOWLEDGMENTS

I also thank Professor Chun-guang Li from the Beijing University of Post and Communications in China for many helpful and enlightening discussions, some of which have already led to collaborations and publications. I was also fortunate to have the chance to work with Dr. Chi Li from the Computer Science department and Professor Stéphane Helleringer at the school of public health on different projects. In particular, from Professor Stéphane Helleringer I got exposed to the fascinating field of research in demography, where the problem of interests, the methodology and the challenges are all dramatically different from my previous research in computer vision. It is also great to bring the two fields together by applying techniques in computer vision to solving problems in demography (and thanks to René again for bring together this collaboration). I also thank Dr. Manolis Tsakiris, now a professor at ShanghaiTech University, for all the wonderful comments during our group meetings.

I am also very grateful to Dr. Jianqiao Feng, a close friend and also a great mentor during my internship period, for bringing me over to Google Inc. in the summer of 2017. This industry experience has been tremendously valuable to me. It not only leads me into thinking more about the practicality of academic research, but also provides me a lot of first hand experience in working with real data on industry-level applications.

It has been an enjoyable experience to work in the Visionlab, and also to be part of the Center for Imaging Science (CIS) and the ECE department at Johns

ACKNOWLEDGMENTS

Hopkins University. I would like to thank Miss Debbie Race, Miss Heather Lockard-Wheeler and Miss Kimberly Biasucci for being very helpful with the administrative issues. I also thank my past and current group members who I met and worked with on a daily basis, with all of you it has been an delightful experience: Dr. Benjamin Bejar, Dr. Ben Haeffele, Dr. Guilherme Franca, Dr. Zhihui Zhu, Dr. Haider Ali, Dr. Manolis Tsakiris, Dr. Bijan Afsari, Dr. Shahin Sefati, Dr. Evan Schwab, Dr. Giann Gorospe, Dr. Lingling Tao, Dr. Hans Lobel, Siddharth Mahendran, Flori Yellin, Efi Mavroudi, Connor Lane, Carolina Pacheco, Soren Wolfers, Hans Lobel, Jacopo Cavazza, Claire Donnat, Congyuan Yang, Ron Boger and many others.

I am grateful for the funding resources that allowed me to pursue my graduate study: the National Science Foundation grants 1447822 and 1618637.

Finally and most importantly, my deepest gratitude to my parents. Without their love and teaching I could not have been here. It is my great honor to dedicate this thesis to them.

Dedication

This thesis is dedicated to my parents, Fengyun Li and Zhendong You, for their eternal love, trust and support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The significant increase in the ability to collect and store diverse information in the past decades has led to an exceptional growth in the availability of data. In the field of computer vision, for instance, portable and affordable digital cameras and smartphones interconnected with high-speed mobile networks have produced image and video datasets of unprecedented scale, which are being collected by giant Internet companies such as Google and Amazon through services they provide to billions of customers. The proliferation in dataset size and complexity is accompanied by the challenge of successfully analyzing the data to discover patterns of interest. Aside from being large-scale, modern datasets very often possess significant amounts of corruptions in various forms such as noise, corrupted entries, outliers and missing entries. All these features pose stark challenges to the development of techniques for modern data

analysis.

1.1 Multi-subspace data

One of the most important observations in data analysis and machine learning is that datasets usually have low-dimensional structures. That is, while modern datasets usually contain thousands or even millions of measurements, the intrinsic degrees of freedom in those measurements are invariably very small. In computer vision, for example, images of a particular face exhibit low-dimensional structures as their variations can be described by a few factors such as the pose of the head, the expression and the lighting conditions. Machine learning methods based on the low-dimensionality of data have found wide applications in various fields that involve data visualization, classification, detection and clustering tasks.

A classic techniques for learning low-dimensional structures from data is principal component analysis (PCA), which assumes that the dataset contain a single low-dimensional affine subspace. While PCA has been extremely popular in many applications, it has the fundamental limitation that it models a *single* subspace and cannot deal with datasets that have multiple subspaces. On the other hand, modern datasets that are mixed with multiple classes are very common, e.g., a face dataset usually contain images from multiple

faces/subjects. In such cases, it is more appropriate to model data as lying in multiple affine subspaces where each subspace corresponds to one class (see Figure 1.1) [18]. Besides face images, a multi-subspace structure appears in many real world datasets such as the feature point trajectories corresponding to multiple rigid moving objects in a video [145], images of handwritten digits [78], gene expression data corresponding to a collection of cancer subtypes [116], and so on.

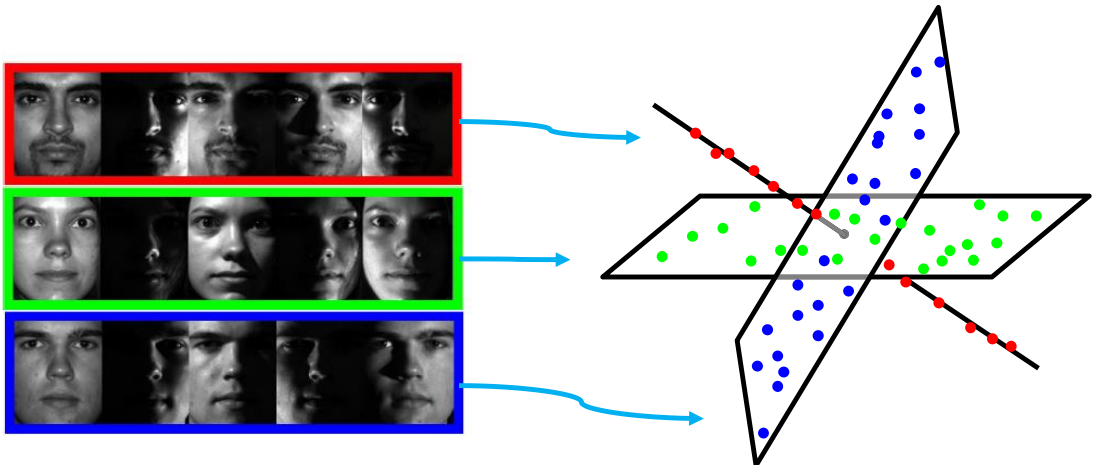


Figure 1.1: An illustration of multi-subspace structure in a face dataset. Images corresponding to the same face lie approximately in a low-dimensional linear subspace, and a face dataset containing images of multiple subjects lie approximately in a union of subspaces where each subspace corresponds to a particular face.

1.2 Multi-subspace learning

In this thesis, we address the problem of learning multi-subspace structure from either labeled or unlabeled data. In the former case, we assume that each

data point in the training set is associated with a label that corresponds to the subspace it comes from, and study the problem of classifying new data points according to their membership to one of the subspaces. In the latter case, we address the more challenging problem of clustering the data into multiple subspaces without knowing the subspaces or the membership of each data point. This gives rise to the subspace classification and subspace clustering problems, which we describe next.

1.2.1 Subspace classification

Classification is one of the most basic topics in machine learning. Given a set of training data points each associated with a groundtruth label, the goal of classification is to assign labels to data points in a test set which is typically drawn from the same distribution as the training set. In particular, the data points in the test set are typically not the same as those in the training set, therefore classification requires the ability to "generalize" from the labels of training data points to the labels of test data points. This can be achieved if certain prior knowledge about the structure of the data in each of the classes is known or provided. In particular, if we know that the dataset has a multi-subspace structure, i.e., that data points from each class are drawn from a low-dimensional subspace of the ambient space, then a testing data point can be classified to the subspace it belongs to. In such cases, the problem is known

as subspace classification.

As a practical application of subspace classification, face recognition is the task of recognizing the face in a image when provided with a training set of face images for multiple human subjects. Since images of the same face under varying illumination conditions lie approximately in the same low-dimensional subspace, the face recognition task may be casted as a subspace classification problem (see Figure 1.2 for an illustration).

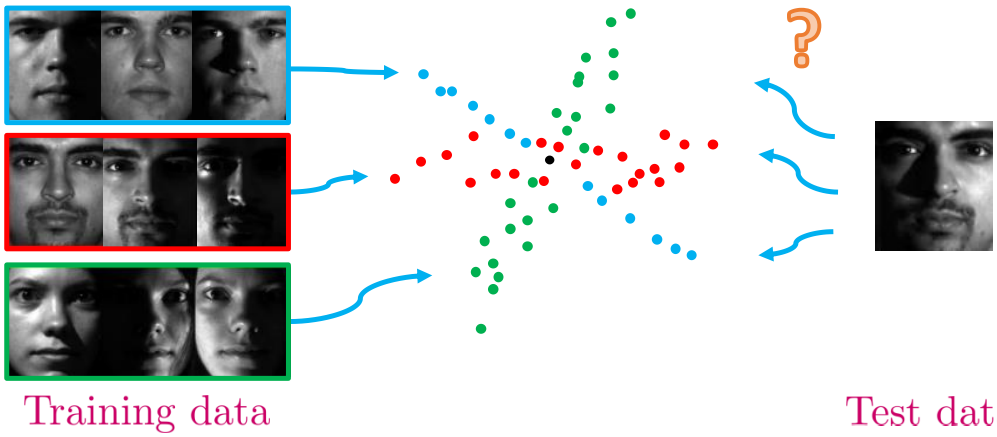


Figure 1.2: An illustration of the subspace classification problem with face recognition as an example. The training set contains face images corresponding to three different faces. The test image correspond to the face of one of the faces in the training set. The goal is to classify the test image according to the subspace it belongs to.

1.2.2 Subspace clustering

Clustering refers to the problem of separating a set of unlabeled data into multiple groups. When each group corresponds to a linear or affine subspace,

the clustering problem is referred to as *subspace clustering*, a problem that has drawn a lot of attention in recent years [164] (see Figure 1.3 for an illustration). Subspace clustering is an important topic particularly in an era of big data, since most of the collected data nowadays are unlabeled, and it takes tremendous human efforts to manually label a huge amount of data. It has found many applications in image representation and compression [82], motion segmentation [135] and temporal video segmentation [163] in computer vision; hybrid system identification in control [14]; community clustering in social networks [83]; and genes expression profiles clustering in bioinformatics [116].

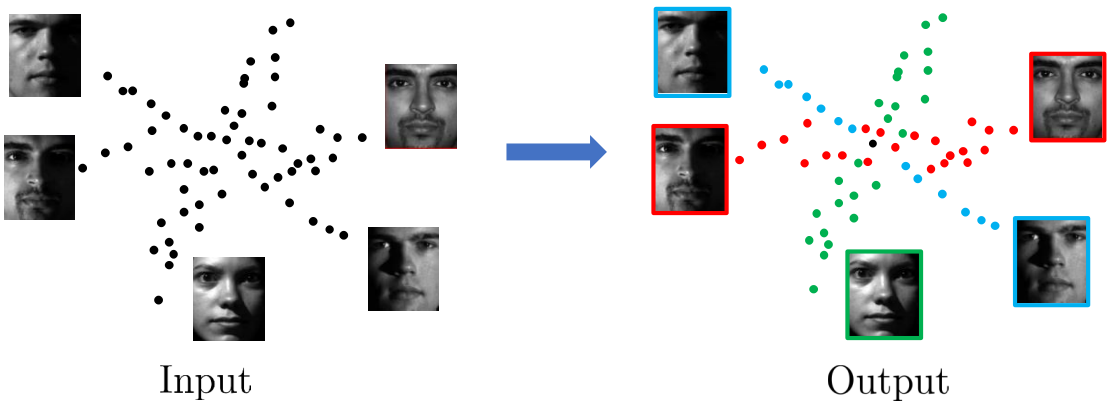


Figure 1.3: An illustration of the subspace clustering problem using face clustering as an example. The input is a set of unlabeled face images corresponding to several different faces. The goal is to segment the input data into several groups where each group corresponds to one particular face. In subspace clustering, this is achieved via separating the data into their respective subspaces.

Subspace clustering is much more challenging than subspace classification because there is no labeled training data at all, which means that there is

no prior knowledge about the number of subspaces, the subspace dimensions, their relative arrangement, and so on. All of these need to be estimated automatically from the data.

1.3 Sparse methods for multi-subspace learning

Sparse methods refers to a general class of methods in which the signal of interest is expressed as a sparse linear combination of other signals that are drawn from a large dictionary. This simple class of methods arises in a surprisingly large number of applications. In the area of signal processing and compressed sensing, in particular, sparse methods have been extensively studied for the purpose of recovering a sparse signal from a few linear measurements. As we will see in Chapter 2, such studies have established the theoretical correctness of several important numerical algorithms for finding sparse solutions to a system of underdetermined linear equations. It has been shown that having a dictionary that is "incoherent" or "isometric" is fundamental for sparse methods to be successful.

The groundbreaking work of Wright et al. [177] and Elhamifar et al. [59] on using sparse methods to solve the subspace classification and subspace clustering problems, respectively, has led to a rapid development of the field of multi-

subspace learning in the past decade. These methods are based on a simple observation that, if a set of data points $\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\} \subseteq \mathbb{R}^D$ lies in a union of low-dimensional subspaces, then a data point $\boldsymbol{x} \in \mathbb{R}^D$ in one of the subspaces can always be expressed as a linear combination of other points from the same subspace. This property is illustrated in Figure 1.4 (left). Mathematically, this could be written as

$$\boldsymbol{x} = \mathbf{X}\boldsymbol{c}, \tag{1.1}$$

where $\mathbf{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N]$ is a matrix containing data points as its columns, and $\boldsymbol{c} \in \mathbb{R}^N$ is the vector of representation coefficients whose nonzero entries correspond to columns in $\mathbf{X}$ that are from the same subspace as $\boldsymbol{x}$. See Figure 1.4 (right) for an illustration. In addition, notice that the number of nonzero coefficients in the representation vector $\boldsymbol{c}$ is equal to the dimension of the subspace that $\boldsymbol{x}$ lies in, therefore is *sparse* if such dimension is small. This motivates us to find such representations via sparse methods.

In general, a representation vector $\boldsymbol{c}$ for which the nonzero entries correspond to points in the data matrix $\mathbf{X}$ that are from the same subspace as $\boldsymbol{x}$ (which are not necessarily *sparse*) are called *subspace-preserving*, a key concept for the study of multi-subspace learning. If subspace-preserving representations can be computed, then one can correctly identify the subspace that any data point lies in as the one that corresponds to the nonzero entries of such representations. Subspace classification and subspace clustering meth-

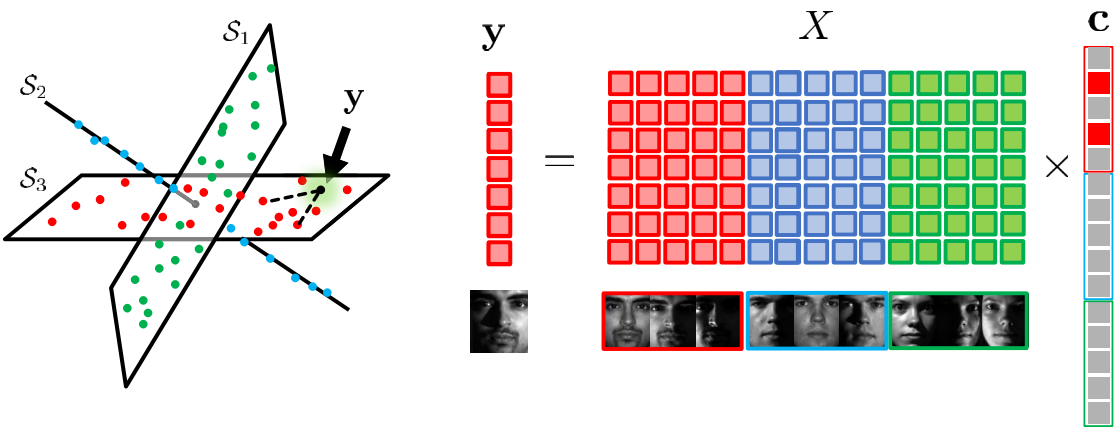


Figure 1.4: An illustration of subspace-preserving representation for multi-subspace data. Left: the data point x in subspace $\mathcal{S}_3$ can be expressed as a linear combination of the two points (shown as the points connected to x via dashed lines) from $\mathcal{S}_3$. Right: x could be expressed as a matrix-vector multiplication that operates on the data matrix X and a coefficient vector c, where the nonzero entries of c correspond to the points that are used in the data representation.

ods based on subspace-preserving representations [59, 177] have been shown to have superior performance and are the state-of-the-art methods.

1.4 Thesis contributions

In this thesis, we develop both theory and algorithms for multi-subspace learning methods that are based on sparse representation. Although theoretical justifications of sparse methods have previously been established in the area of signal processing, data models in that area are inappropriate in the context of multi-subspace learning. In particular, the previously established "incoherent" and "isometric" assumptions on the dictionary are often violated

when data is drawn from a union of subspaces. This calls for the development of novel theoretical analysis that explains the huge success of sparse methods for multi-subspace learning.

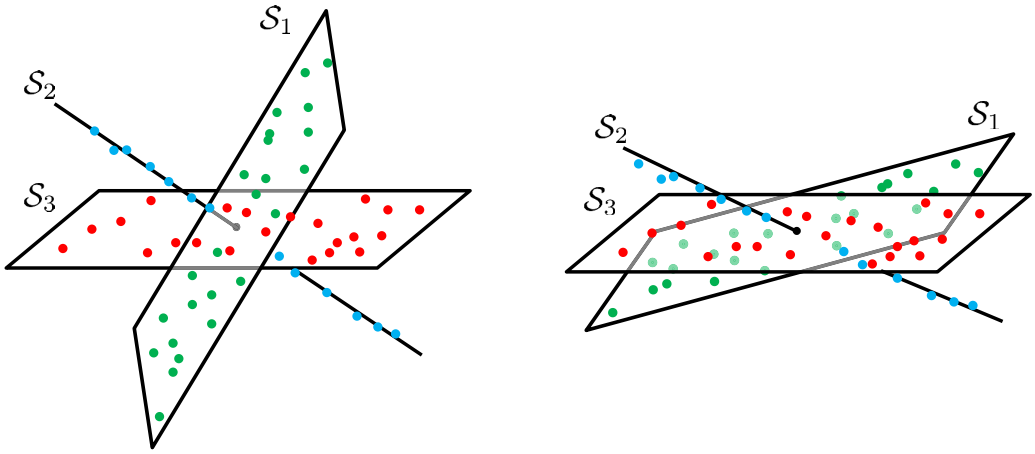
In terms of algorithms, previous subspace clustering methods are designed for data sets that are small scale, clean and balanced across different classes, which are unrealistic assumptions for real world applications. We will see that existing methods are limited to datasets that contain $10,000$ data points, and cannot handle larger datasets due to the high memory and computational complexity. We will also see that the performance of existing methods drops by a significant amount as soon as there are more than 1% percent outliers in the dataset. Finally, imbalanced data distribution can also significantly compromise the performance of existing clustering methods. All of these challenges call for the development of scalable and robust algorithms that can effectively deal with data in real applications.

1.4.1 Geometric and probabilistic analysis of multi-subspace learning methods

The notion of *subspace-preserving* property introduced above plays a central role in the theoretical study of multi-subspace learning. In particular, the study of conditions for subspace-preserving recovery is an essential step in proving

the correctness of the associated multi-subspace learning algorithm. In Chapter 3 we present a systematic study of the theories of subspace-preserving recovery. Our contribution is comprised of the following four parts.

Geometric conditions [195]. Our analysis identifies key geometric quantities associated with data in multiple subspaces that affect the correctness of the sparse methods for learning from such data. Intuitively, if the multiple subspaces are well separated and have large angle between each other, then the multi-subspace learning task is easier. This idea is illustrated in Figure 1.5, in which the three subspaces in Figure 1.5a are more separated than the three subspaces in Figure 1.5b.

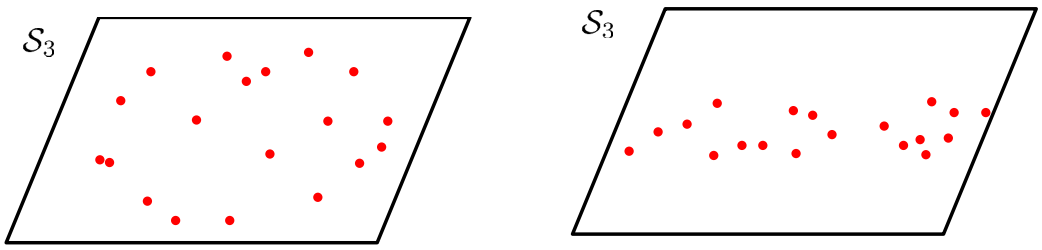


(a) Case 1 **(b)** Case 2

Figure 1.5: The effect of subspace separation on multi-subspace learning. (a) Subspaces are well separated. (b) Subspaces are not well separated.

Besides subspace separation, another factor that affects multi-subspace learning is the distribution of points in the subspaces. Figure 1.6 shows two possible

cases of point distribution in one of the subspaces in a multi-subspace structure. In the case shown in Figure 1.6a, the data points are well distributed in all directions in this subspace. In the case shown in Figure 1.6b, the distribution of data points is skewed towards a specific direction in this subspace. It could be expected that multi-subspace learning with data distribution in the latter case is more difficult than that in the former case.



(a) Case 1 **(b)** Case 2

Figure 1.6: The effect of point distribution in each of the subspaces on multi-subspace learning. (a) Points from the subspace $\mathcal{S}_3$ are well distributed in the subspace. (b) Points from $\mathcal{S}_3$ are not well distributed in the subspace.

Based on these geometric intuitions, we derive geometric conditions that guarantee subspace-preserving recovery, which require the subspaces to be sufficiently well separated and the data in each subspace to be sufficiently well distributed. Our analysis addresses both the problem of *instance* recovery, where the goal is to study subspace-preserving recovery for a particular point in the subspace, as well as the problem of *universal* recovery, where the goal is to find subspace-preserving recovery for all data points in the subspace.

Probabilistic conditions [192]. We further explore the regimes in which the

geometric conditions can be satisfied by considering models where data points are generated according to a probabilistic model. Our analysis reveals that sparse methods for multi-subspace learning work better when the dimension of the subspaces are low relative to the ambient dimension, a phenomenon that has been observed in practice. In addition, our result also reveals how the density of samples in the subspaces affects the correctness of the multi-subspace learning methods.

Conditions for subspace classification [192]. Using these tools developed in the study of subspace-preserving recovery, we provide justification for the correctness of sparsity based subspace classification method. Specifically, we provide conditions under which the subspace classification method in [177] is guaranteed to correctly classify any test data point given a certain training dataset. To the best of our knowledge, this provides the first correctness guarantee of this method for the multi-subspace model.

Conditions for subspace clustering [191,193]. We also provide correctness conditions for the sparse representation based subspace clustering method in [59]. Unlike the case of subspace classification in which subspace-preserving recovery automatically implies the correctness of classification by sparse methods, the subspace clustering task could suffer from the issue of over-segmentation of clusters due to lack of connectivity in the sparse representations. We will address this challenge by using elastic net regularization in lieu of sole sparse

regularization to incur denser solutions. In particular, we provide geometric explanations to rigorously justify that the elastic net has the effect of balancing between connectedness and subspace-preserving properties.

1.4.2 Algorithms for handling large-scale data

Existing subspace clustering methods based on sparse representation cannot effectively deal with large scale data since finding such sparse representations is computationally difficult. In Figure 1.7, we report the running time of a baseline algorithm which is a subspace clustering algorithm based on computing sparse representation with the alternating direction method of multipliers (see Chapter 4 for details). The figure shows that even when dealing with a medium scale dataset that has around $10{,}000$ data points, the running time already goes up to $\sim 10{,}000$ seconds or around 3 hours. In addition, the baseline method cannot handle datasets of size much larger than $10{,}000$ data points on a typical machine with, say, 16 Gigabytes memory as the algorithm has quadratic memory complexity. In Chapter 4 we will present two approaches that are not only significantly faster than the baseline method but also able to handle datasets of much larger scale.

Active support method [191]. Our first approach for handling large-scale data is a novel active support algorithm for solving the sparse recovery problems more efficiently. By exploits the geometry of the solution to the sparse

Figure 1.7: Challenge and contribution for subspace clustering on large-scale data. We generate datasets where 5 subspaces of dimension 6 are sampled independently and uniformly at random from an ambient space of dimension 9, and $N/5$ data points are sampled independently and uniformly at random from each of these subspaces, where N is the total number of data points in the dataset and is varied in the x-axis. We then apply a baseline subspace clustering method (i.e., SSC-BP with sparse recovery solved by ADMM, see Chapter 2) and our method (i.e., SSC-BP with sparse recovery solved by the active support method, see Chapter 4) for clustering the points in these datasets.

recovery problem, we use an iterative procedure to update an active support set, which is guaranteed to converge to the support set of the optimal solution in a finite number of iterations. Our algorithm achieves its efficiency as it decomposes the large-scale sparse recovery problem into a sequence of problems of much smaller size, each of which can be solved much more efficiently. The performance of this method is illustrate in Figure 1.7, where it can be seen that the active support method handles $10{,}000$ in only ~ 10 seconds, a $\sim 1{,}000$ times speedup over the baseline. Moreover, it is able to handle as many as $\sim 350{,}000$ data points in a few hours.

Divide-and-Conquer method [189]. To handle even larger scale data, we also present a novel divide-and-conquer framework for large-scale subspace clustering. In this method, the data is first divided into chunks and subspace clustering is applied to each chunk. After removing potential outliers from each cluster, a new cross-representation measure for the similarity between subspaces is used to merge clusters from different chunks that correspond to the same subspace. A self-representation method is then used to assign outliers to clusters.

1.4.3 Algorithms for handling corrupted data

Existing subspace clustering methods are also susceptible to the presence of outliers which are points that do not lie in the union of subspaces. To illustrate the effect of outliers on subspace clustering, we perform experiments on synthetic datasets where the inliers are drawn from a union of low-dimensional subspaces, while the outliers are points that are randomly sampled from the ambient space. The performance of existing subspace clustering methods is shown as the "baseline" in Figure 1.8. We can see that the baseline performance drops to below 80% accuracy with as few as 5% outliers. This shows that it is essential to detect and reject the outliers before subsequent subspace clustering is performed.

In Chapter 5 we present a novel outlier detection method that can effec-

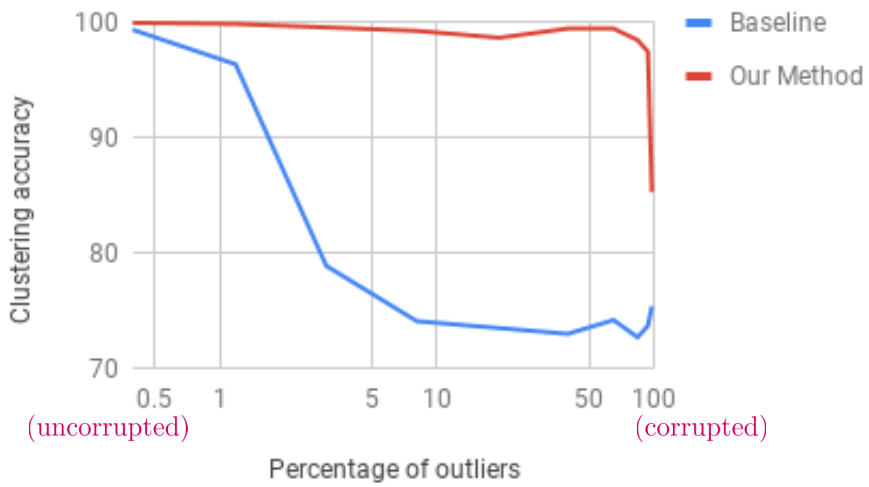


Figure 1.8: Challenge and contribution for subspace clustering on corrupted data. We generate datasets that each is composed of an inlier set and an outlier set. For each inlier set, 4 subspaces of dimension 3 are sampled independently and uniformly at random from an ambient space of dimension 12, then $20, 40, 80$ and 110 data points are sampled independently and uniformly at random from the four subspaces, respectively. For each outlier set, a certain number of points are sampled independently and uniformly at random from the ambient space. We vary the number of outliers to show the performance on datasets with different percentage of outliers. Baseline: apply existing subspace clustering on the entire dataset. Our method: apply outlier detection in Chapter 5 and run existing subspace clustering on the detected set of inliers. In both cases, the clustering performance is evaluated on the set of inliers.

tively detect outliers from a union of subspaces [194]. Our method is based on utilizing random walks on a graph. The observation is that while inliers (i.e., points in the union of subspaces) can be expressed as linear combinations of a few other inliers, outliers express themselves as a linear combination of both inliers and outliers. By exploiting this property, we compute a weighted directed graph from the sparse representation. By defining a suitable Markov Chain from this graph, we establish a connection between inliers/outliers and

essential/inessential states of the Markov chain, which allows us to detect outliers by using random walks. Empirically, by applying this method to detect and remove outliers prior to applying subspace clustering, it is possible to cluster datasets that are corrupted by more than 90% outliers (see Figure 1.8). We will provide a theoretical analysis that justifies the correctness of our method under geometric and connectivity assumptions.

1.4.4 Algorithms for handling imbalanced data

Another regime where existing subspace clustering methods does not work well is imbalanced data distributions, where different classes in a dataset contain dramatically different number of data points. To illustrate the effect of imbalanced data, we perform subspace clustering on dataset that is composed of two classes with varying proportion of points from one of the classes. The clustering performance is shown in Figure 1.9. We can see that while the baseline method (SSC-BP with sparse recovery solved by active support method, see Chapter 4) gives near 100% clustering accuracy when the dataset is balanced, its performance drops to below 80% when the dominant class contains $> 90\%$ data points of the dataset.

In Chapter 6, we present an exemplar based subspace clustering method to tackle the problem of imbalanced data [190]. Our method is based on searching for a subset of the data that best represents all data points as measured by

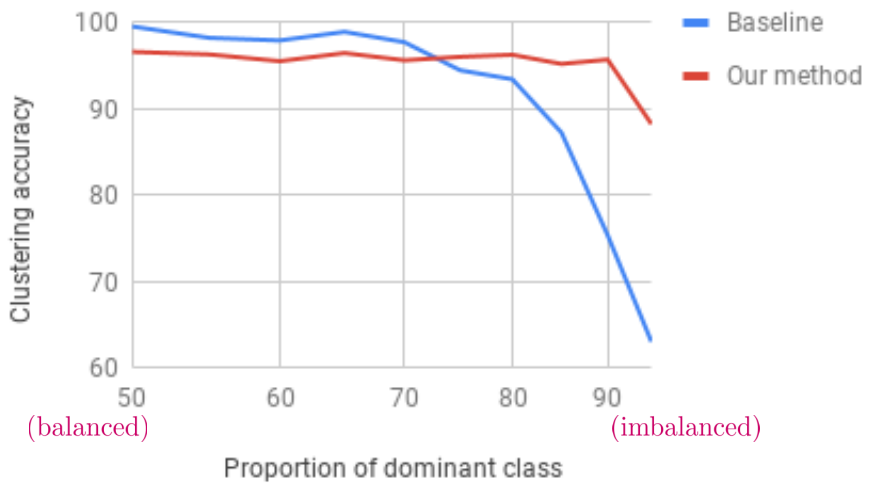


Figure 1.9: Challenge and contribution for subspace clustering on imbalanced data. x and $100 - x$ points (x is varied in the x-axis) are drawn uniformly at random from 2 subspaces of dimension 3 drawn uniformly at random in an ambient space of dimension 5. Baseline: subspace clustering method SSC-BP with sparse recovery solved by active support method, see Chapter 4. Our method: exemplar subspace clustering, see Chapter 6.

the ℓ_1 norm of the sparse representation coefficients. Geometrically, we show that the solution to our model is a subset that best covers all data points as measured by the Minkowski functional of the subset. We introduce a farthest first search algorithm for approximately solving our model, which iteratively selects the least well-represented point as an exemplar. When data comes from a union of subspaces, we prove that the computed subset contains enough exemplars from each subspace for expressing all data points even if the data is imbalanced. The performance of our exemplar based subspace clustering is illustrated in Figure 1.9, in which one can see that our method has much higher clustering accuracy than the baseline method when the dataset is imbalanced.

1.4.5 Evaluation of subspace clustering on real data

In the past decade, a large number of methods have been developed for subspace clustering and significant improvement in clustering performance has been reported in a series of papers. However, the difference in the databases that are used in these papers makes it impossible to directly compare the performance of different methods. Even though there are a few commonly used databases such as the Extended Yale B face database [71], different parts of these databases are used in the experiments from different papers and different data preprocessing procedures are applied. Therefore, a direct comparison of results from these experiments will be inconsistent.

In this thesis, we address this issue by carrying out a thorough experimental evaluation of many representative subspace clustering methods on several real databases. In particular, we use 6 datasets to evaluate different aspects of the performance of the methods. The Extended Yale B face database [71] and the Coil-100 image database [125] are medium scale databases that have been commonly used in previous subspace clustering experiments. The CIFAR-10 dataset [88] and the MNIST dataset [92] are large scale image databases that contain $60{,}000$ and $70{,}000$ images, respectively. We use these two datasets to evaluate the scalability of different subspace clustering methods. Finally, the

Table 1.1: Evaluation of subspace clustering methods on real data: a summary. "-": method is not tested on this dataset. "M": method exceeds 20GB memory limit. "T": method exceeds 24 hours running time limit.

	EYaleB	Coil-100	Cifar-10	MNIST	GTSRB	EMNIST
k-means	ch. 4	ch. 4	ch. 4	ch. 4	ch. 6	ch. 6
Spectral	ch. 4	ch. 4	ch. 4	ch. 4	ch. 6	ch. 6
LRR	ch. 4	ch. 4	M	M	-	-
LRSC	ch. 4	ch. 4	M	M	-	-
LRSC	ch. 4	ch. 4	M	M	-	-
O-LRSC	ch. 4	ch. 4	ch. 4	ch. 4	ch. 6	ch. 6
LSR	ch. 4	ch. 4	M	M	-	-
SSC-ADMM	ch. 4	ch. 4	M	M	-	-
ℓ_0-SSC	ch. 4	ch. 4	T	T	T	T
NSN	ch. 4	ch. 4	T	T	-	-
SBC	-	-	-	-	ch. 6	ch. 6
SSC-OMP	ch. 4	ch. 4	ch. 4	ch. 4	ch. 6	ch. 6
SSC-BP	ch. 4	ch. 4	ch. 4	ch. 4	ch. 6	ch. 6
EnSC	ch. 4	ch. 4	ch. 4	ch. 4	-	-
ESC-FFS	-	-	-	-	ch. 6	ch. 6

GTSRB database [141] and the Extended MNIST (EMNIST) database [44] are two databases that are imbalanced across different classes. Using these two databases, we demonstrate the performance of subspace clustering methods on imbalanced data.

We compare with a wide range of clustering methods including k-means and Spectral, as well as subspace clustering methods such as LRR, LRSC and and so on. A detailed description of these methods can be found in Chapter 4 and Chapter 6. A summary of the methods as well as the datasets that the methods are tested on is given in Table 1.1.

Chapter 2

Background

2.1 Notation

Throughout this thesis, we use $\mathbb{R}$ to denote the set of real numbers, and $\mathbb{R}^D$ to denote the D-dimensional linear space.

We use lowercase letters denote scalars, such as $x \in \mathbb{R}$, lowercase boldface letters denote vectors, such as $\boldsymbol{x} \in \mathbb{R}^D$, and uppercase boldface letters denote matrices, such as $\mathbf{X} \in \mathbb{R}^{D \times N}$. The transpose of the matrix $\mathbf{X} \in \mathbb{R}^{D \times N}$ is denoted as $\mathbf{X}^\top \in \mathbb{R}^{N \times D}$. The uppercase calligraphic letters denote sets, such as $\mathcal{X} \subseteq \mathbb{R}^D$.

For any vector $\boldsymbol{x} = [x_1, \cdots, x_D] \in \mathbb{R}^D$ and $p \geq 1$, the ℓ_p-norm is defined as $\|\boldsymbol{x}\|_p = \left(\sum_{i=1}^D |x_i|^p\right)^{1/p}$. As p approaches infinity, we have the infinity norm $\|\boldsymbol{x}\|_\infty = \max_{i=1}^D |x_i|$. Another particularly interesting case for this thesis is when $p = 0$, for which we define $\|\boldsymbol{x}\|_0$ as the number of nonzero entries in $\boldsymbol{x}$.

2.2 Sparse signal recovery

In this section, we provide a brief review of sparse methods, including both sparse recovery algorithms and sparse recovery theory. For more detailed investigations of results in this area, we refer the reader to [16, 27, 33, 68].

Sparse signal recovery addresses the problem of recovering a *sparse* signal $\boldsymbol{c}_0 \in \mathbb{R}^N$ from a limited number $D \ll N$ of linear observations

$$b_k = \langle \boldsymbol{c}_0, \boldsymbol{a}^{(k)} \rangle, \quad k = 1, \cdots, D \text{ or equivalently, } \boldsymbol{b} = \mathbf{A}\boldsymbol{c}_0, \tag{2.1}$$

where $\mathbf{A} = [\boldsymbol{a}^{(1)}, \cdots, \boldsymbol{a}^{(D)}]^\top \in \mathbb{R}^{D \times N}$ is a full row-rank matrix. Since there are less measurements than the number of unknowns, the system of linear equations in (2.1) is under-determined, i.e., there are infinitely many $\boldsymbol{c}$ that satisfy $\boldsymbol{b} = \mathbf{A}\boldsymbol{c}$. Therefore, at a first glance, it seems impossible to recover the true signal $\boldsymbol{c}_0$. However, if $\boldsymbol{c}_0$ is sparse enough, meaning that among all the N entries in $\boldsymbol{c}_0$, only very few of them are not zero, then the recovery of $\boldsymbol{c}_0$ may be possible. In particular, one can achieve this if $\boldsymbol{c}_0$ is the sparsest one among all possible solutions to the equations $\boldsymbol{b} = \mathbf{A}\boldsymbol{c}$, in which case $\boldsymbol{c}_0$ can be recovered by solving the following optimization problem:

$$\min_{\boldsymbol{c}} \|\boldsymbol{c}\|_0 \quad \text{s.t. } \boldsymbol{b} = \mathbf{A}\boldsymbol{c}. \tag{2.2}$$

How can we guarantee that $\boldsymbol{c}_0$ is the sparsest solution to $\boldsymbol{b} = \mathbf{A}\boldsymbol{c}$ so that it can be recovered through solving (2.2)? It turns out that the answer to this question depends on whether the measurement matrix $\mathbf{A}$ satisfies certain properties. In particular, a key property is the *spark* of the matrix $\mathbf{A}$ [53], also known as the *Kruskal rank*. Its definition is as follows.

Definition 1 (Spark)**.** The spark of a matrix $\mathbf{A}$, denoted as $\mathrm{spark}(\mathbf{A})$, is the smallest number of columns from $\mathbf{A}$ that are linearly dependent.

Theorem 1 (Uniqueness of the solution to the ℓ_0-problem [53])**.** *If* $\boldsymbol{b} = \mathbf{A}\boldsymbol{c}_0$ *and* $\|\boldsymbol{c}_0\|_0 < \mathrm{spark}(\mathbf{A})/2$*, then* $\boldsymbol{c}_0$ *is the unique solution to* (2.2)*.*

It follows from this theorem that large values of the spark are helpful for recovering sparse signals. From its definition, the largest possible value of $\mathrm{spark}(\mathbf{A})$ is $D + 1$ as any $D + 1$ columns from $\mathbf{A}$ must be linearly dependent. In fact, this upper bound can be attained. For example, if the entries of $\mathbf{A}$ are drawn from independent and identically distributed Gaussian random variables, then $\mathrm{spark}(\mathbf{A}) = D + 1$ with probability 1. In this case, correct sparse recovery can be achieved for any $\boldsymbol{a}_0$ that has fewer than $(D + 1)/2$ nonzero entries.

2.2.1 Sparse recovery algorithms

The result from the previous section is promising since it shows that the recovery of a vector $\boldsymbol{c}_0$ with $s_0 := \|\boldsymbol{c}_0\|_0$ nonzero entries can be achieved using only $D + 1 > 2 \cdot s_0$ measurements, regardless of the length N of the signal $\boldsymbol{c}$. In practice, however, the challenge lies in that even if $\boldsymbol{c}_0$ is the unique solution to (2.2), solving this optimization problem can be very difficult. Consider, for example, a naive way of solving (2.2) by an exhaustive search procedure. That is, take every possible support set $\mathcal{J} \subseteq \{1, \cdots, N\}$ of size s with s increasing from 1 until termination, compute the span of those columns of $\mathbf{A}$ that are indexed by $\mathcal{J}$, and terminate once $\boldsymbol{b}$ lies in such span. This procedure requires $\sum_{s=1}^{s_0} \binom{N}{s}$ number of trials, which is exponential in N and is prohibitively large as N and s_0 increase. Unfortunately, no significantly better algorithm is known for solving (2.2) in general. It is now known that the problem (2.2) is NP-hard.

Theorem 2 (Hardness of ℓ_0-problem [124])**.** *The problem* (2.2) *is NP-hard.*

Even though (2.2) is NP-hard, it does not imply that *all* instances of the problem are difficult. As it turns out, many sparse recovery problems in practical scenarios can be solved using much more efficient algorithms. Here, we present two of the most popular such methods, one by means of convex relaxation and the other one by a greedy procedure, which are also mostly related to our study of multi-subspace learning in the later chapters of this thesis.

2.2.1.1 Basis Pursuit (BP)

The difficulty in solving the sparse recovery problem (2.2) has its roots in the discrete and discontinuous nature of the ℓ_0 regularization. To address this issue, one idea is to "relax" the ℓ_0 minimization problem by replacing the ℓ_0 regularization with the ℓ_1 norm, which is in some sense the convex surrogate for the ℓ_0 regularization. Consequently, we solve the following optimization problem, which is commonly known as the basis pursuit (BP) problem [40]

$$\min_{\boldsymbol{c}} \|\boldsymbol{c}\|_1 \quad \text{s.t. } \boldsymbol{b} = \mathbf{A}\boldsymbol{c}, \tag{2.3}$$

where $\|\boldsymbol{c}\|_1 = \sum_{j=1}^{N} |c_j|$. The benefit of (2.3) is that it is a convex optimization problem, therefore it can be solved much more efficiently. There are generic "off the shelf" convex optimization solvers as well as specifically designed techniques which can solve (2.3) in polynomial time. Meanwhile, a remarkable fact about the optimization problem in (2.3) is that, under certain conditions, it has the same solution as that of the optimization problem in (2.2). We will review some of the fundamental results in Section 2.2.2. But before doing that, let us first review another computational strategy for attacking the NP-hard sparse recovery problem in (2.2).

2.2.1.2 Orthogonal Matching Pursuit (OMP)

The orthogonal matching pursuit (OMP) algorithm [131] is a greedy method for finding sparse representations. Consider the following alternative formulation of the sparse recovery problem

$$\min_{\boldsymbol{c}} \|\boldsymbol{b} - \mathbf{A}\boldsymbol{c}\|_2 \quad \text{s.t. } \|\boldsymbol{c}\|_0 \leq k_{\max}, \tag{2.4}$$

where $k_{\max}$ is the target sparsity level. OMP aims to solve (2.4) by sequentially choosing one column (referred to as an atom) of A in a *locally* optimal manner, and abandoning the *global* exhaustive search in a naïve algorithm. It keeps track of a residual $\boldsymbol{v}^{(k)}$ at iteration k (initialized as the input signal $\boldsymbol{b}$) and of a support set $\mathcal{W}^{(k)}$ that contains the atoms already chosen (initialized as the empty set). At each step, $\mathcal{W}^{(k)}$ is updated to $\mathcal{W}^{(k+1)}$ by adding the column of A that has the maximum absolute inner product with $\boldsymbol{v}^{(k)}$ (ties are broken arbitrarily), i.e.,

$$\mathcal{W}^{(k+1)} = \mathcal{W}^{(k)} \bigcup \{j^*\}, \text{ where } j^* = \underset{j=1,\ldots,N}{\arg\max} |\boldsymbol{a}_j^\top \boldsymbol{v}^{(k)}|, \tag{2.5}$$

where $\boldsymbol{a}_j$ is the j-th columns of $\mathbf{A}$. It then computes an approximation of $\boldsymbol{b}$ using atoms in $\mathcal{W}^{(k+1)}$, i.e.,

$$\boldsymbol{c}^{(k+1)} = \underset{\boldsymbol{c}:\mathbf{supp}(\boldsymbol{c})\subseteq\mathcal{W}^{(k+1)}}{\arg\min} \|\boldsymbol{b} - \mathbf{A}\boldsymbol{c}\|_2. \tag{2.6}$$

Next, the residual is updated to $\boldsymbol{v}^{(k+1)} = \boldsymbol{b} - \mathbf{A}\boldsymbol{c}^{(k+1)}$, which is the component of $\boldsymbol{b}$ that is orthogonal to the space spanned by the atoms indexed in $\mathcal{W}^{(k+1)}$. The process is typically terminated when the norm of the residual $\|\boldsymbol{v}^{(k)}\|_2$ is smaller than a threshold value $\epsilon \geq 0$ or when the iteration k reaches a maximum allowed value $k_{\max}$. In the theoretical analysis of OMP in Section 2.2.2 and in Chapter 3 we always take the termination condition to be $\epsilon = 0$ and $k_{\max} = \infty$. The overall algorithm is summarized in Algorithm 1. This greedy procedure can be much more efficient than the exhaustive search as the computational complexity is on the order of $k_{\max}DN$ in general (assuming $N \gg D$).

2.2.2 Sparse recovery theory

We have seen that the convex relaxation based approach BP and the greedy based approach OMP are expected to solve the sparse recovery problem (2.2). Are there any formal guarantees for their success? Clearly, there will be no such guarantees for all cases since the problem is NP-hard in general while both BP and OMP are polynomial time algorithms. However, if the true sparse

Algorithm 1 : Orthogonal Matching Pursuit (OMP)

Input: $\mathbf{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_N] \in \mathbb{R}^{D \times N}$, $\boldsymbol{b} \in \mathbb{R}^D$, $k_{\max}$, ϵ.

1: **Initialize** $k = 0$, residual $\boldsymbol{v}_0 = \boldsymbol{b}$, support set $\mathcal{W}^{(0)} = \emptyset$.

2: **while** $k < k_{\max}$ and $\|\boldsymbol{v}^{(k)}\|_2 > \epsilon$ **do**

3: $\mathcal{W}^{(k+1)} = \mathcal{W}^{(k)} \bigcup \{j^*\}$, where $j^* = \underset{j=1,\ldots,N}{\arg\max} |\boldsymbol{a}_j^\top \boldsymbol{v}^{(k)}|$.

4: $\boldsymbol{v}^{(k+1)} = (I - P_{\mathcal{W}^{(k+1)}})\boldsymbol{b}$, where $P_{\mathcal{W}^{(k+1)}}$ is the projection onto the span of the vectors $\{\boldsymbol{a}_j, j \in \mathcal{W}^{(k+1)}\}$.

5: $k \leftarrow k + 1$.

6: **end while**

Output: $\boldsymbol{c}^* = \arg\min_{\boldsymbol{c}:\text{Supp}(\boldsymbol{c}) \subseteq \mathcal{W}^{(k)}} \|\boldsymbol{b} - \mathbf{A}\boldsymbol{c}\|_2$.

signal $\boldsymbol{c}_0$ is "sufficiently sparse" and the measurement matrix $\mathbf{A}$ is "sufficiently isometric", then the success of these algorithms can be guaranteed. We now make this statement precise by presenting some of the most important results in the study of the sparse signal recovery problem.

Mutual coherence condition. A simple observation that motivates the theoretical study of sparse recovery is that if the columns of the measurement matrix $\mathbf{A}$ are linearly independent, then any signal $\boldsymbol{c}_0$ can be recovered as the solution to the system of linear equations $\boldsymbol{b} = \mathbf{A}\boldsymbol{c}$. In particular, if the columns of $\mathbf{A}$ are orthogonal, then $\boldsymbol{c}$ can be easily solved as $\boldsymbol{c} = \mathbf{A}^\top \boldsymbol{b}$. In interesting application scenarios of sparse recovery, however, we typically have $N \gg D$, and therefore the columns of $\mathbf{A}$ cannot be linearly independent or orthogonal. A

fundamental result in compressed sensing is that if the columns of $\mathbf{A}$ are "approximately" orthogonal, then correct sparse recovery can be achieved. This is characterized by the mutual coherence of $\mathbf{A}$, which is defined as follows.

Definition 2 (Mutual coherence)**.** The mutual coherence of matrix $\mathbf{A} = [\boldsymbol{a}_1, \cdots, \boldsymbol{a}_N]$, denoted as $\mu(\mathbf{A})$, is defined as the largest absolute normalized inner product between every pair of columns from $\mathbf{A}$. That is,

$$\mu(\mathbf{A}) = \max_{1 \leq i < j \leq N} \frac{|\langle \boldsymbol{a}_i, \boldsymbol{a}_j \rangle|}{\|\boldsymbol{a}_i\|_2 \|\boldsymbol{a}_j\|_2}. \tag{2.7}$$

Clearly, if the columns of $\mathbf{A}$ are orthogonal then we have $\mu(\mathbf{A}) = 0$. For matrices with more columns than rows, that is $N > D$, the mutual coherence is necessarily strictly positive. Intuitively, mutual coherence captures how close the matrix $\mathbf{A}$ is to being orthogonal. If $\mu(\mathbf{A})$ is small enough we can achieve guaranteed recovery by BP and OMP as stated in the next theorem.

Theorem 3 (Correctness of BP and OMP for sparse recovery via mutual coherence [53, 148])**.** *For a system of linear equations $\boldsymbol{b} = \mathbf{A}\boldsymbol{c}_0$, if*

$$\|\boldsymbol{c}_0\|_0 < \frac{1}{2}\big(1 + \frac{1}{\mu(\mathbf{A})}\big), \tag{2.8}$$

then the solution found by BP and OMP is equal to $\boldsymbol{c}_0$.

The condition in (2.8) can be satisfied if $\|\boldsymbol{c}_0\|_0$ and $\mu(\mathbf{A})$ are small enough. That is, sparse recovery can be achieved by BP and OMP if the true signal is sparse enough and the measurement matrix is incoherent.

Restricted isometry condition. The restricted isometry property is a more refined measure of how close the measurement matrix is to be an isometry than the mutual coherence property. It is defined as follows.

Definition 3 (Restricted isometry property (RIP))**.** A matrix $\mathbf{A}$ is said to satisfy the RIP of order s if there exists a constant $\delta \in (0, 1)$ such that

$$(1 - \delta)\|\boldsymbol{c}\|_2^2 \leq \|\mathbf{A}\boldsymbol{c}\|_2^2 \leq (1 + \delta)\|\boldsymbol{c}\|_2^2 \tag{2.9}$$

holds for any $\boldsymbol{c}$ such that $\|\boldsymbol{c}\|_0 \leq s$. The order-$s$ restricted isometry constant $\delta_s(\mathbf{A})$ is the smallest number δ such that the above inequality holds.

Intuitively, the restricted isometry constant is small if any s columns of $\mathbf{A}$ are close to being an isometry, i.e., being a mapping that preserves the norm of every vector.

The theoretical results for guaranteed sparse recovery in terms of RIP are much richer than those for the mutual coherent condition, see e.g., [30, 31, 35, 38,50,119,120,174]. Here, we only present results from two most recent papers [30] and [174] which developed sharp bounds for BP and OMP, respectively.

The readers are referred to the references therein for a more complete list of results.

Theorem 4 (Correctness of BP for sparse recovery via RIP [30])**.** *For a system of linear equations* $\boldsymbol{b} = \mathbf{A}\boldsymbol{c}_0$ *where* $\|\boldsymbol{c}_0\|_0 \leq s$*, if*

$$\delta_s < \frac{1}{3}, \tag{2.10}$$

then BP is guaranteed to find $\boldsymbol{c}_0$*.*

Theorem 5 (Correctness of OMP for sparse recovery via RIP [174])**.** *For a system of linear equations* $\boldsymbol{b} = \mathbf{A}\boldsymbol{c}_0$ *where* $\|\boldsymbol{c}_0\|_0 \leq s$*, if*

$$\delta_{s+1} < \frac{1}{\sqrt{s+1}}, \tag{2.11}$$

then OMP is guaranteed to find $\boldsymbol{c}_0$*.*

The RIP is particularly useful for studying sparse recovery with random matrices. A well-known result is the following theorem which states that matrices with entries drawn from a Gaussian distribution satisfy the RIP with high probability.

Theorem 6 (RIP of Gaussian matrices [17])**.** *There exists a numerical constant* $C > 0$ *such that if* $\mathbf{A} \in \mathbb{R}^{D \times N}$ *is a matrix whose entries are drawn from independent standard Gaussian distributions, then the restricted isometry constant of*

the matrix $\frac{\mathbf{A}}{\sqrt{D}}$ *satisfies* $\delta_s \leq \delta$ *with probability at least* $1 - 2\exp(-\frac{\delta^2 D}{2C})$*, provided that*

$$D \geq 2Cs\log(eN/s)/\delta^2. \tag{2.12}$$

Combining this result with Theorem 4, we see that with high probability, sparse recovery of any $\|\boldsymbol{c}_0\|_0 \leq s$ can be achieved by BP with Gaussian random matrix, provided that

$$D \geq \mathcal{O}(s\log(eN/s)). \tag{2.13}$$

This result suggests that to recover signals with s nonzero entries, the required number of measurements D is (roughly) linear in s. Similarly, from Theorem 5 we see that sparse recovery by OMP with Gaussian random matrix can be achieved when

$$D \geq \mathcal{O}(s^2\log(eN/s)). \tag{2.14}$$

2.3 Sparse representation classification (SRC)

We start by formally defining the problem of subspace classification.

Definition 4 (Subspace classification)**.** Let $\mathbf{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N] \in \mathbb{R}^{D \times N}$ be a matrix whose columns lie in a union of n subspace $\cup_{\ell=1}^{n} \mathcal{S}_\ell$. Let $\boldsymbol{y} = [y_1, \cdots, y_N] \in \{1, \cdots, n\}^N$ be a vector where $y_j \in \{1, \cdots, n\}$ is the membership of $\boldsymbol{x}_j$ to the subspace it belongs to, i.e., $\boldsymbol{x}_j \in \mathcal{S}_{y_j}$. The goal of subspace classification is that for any test data $\boldsymbol{x} \in \mathbb{R}^D$ we find the label $y \in \{1, \cdots, n\}$ such that $\boldsymbol{x} \in \mathcal{S}_y$.

A traditional method for addressing the subspace classification problem is the Nearest Subspace method [81], where each test data $\boldsymbol{x}$ is assigned to the closest subspace spanned by all training samples from each class. While being conceptually simple and elegant, the nearest subspace method is not very reliable in real data applications since data usually contains large amount of noise and corruptions, making it difficult to have a good estimate of the subspaces.

Recently, sparse representation ideas have been introduced to the areas of machine learning for addressing the challenges of learning from high-dimensional data. A key observation is that, for any test data $\boldsymbol{x}$ in the union of subspaces, there always exists sparse solutions to $\boldsymbol{x} = \mathbf{X}\boldsymbol{c}$, where the nonzero entries of $\boldsymbol{c}$ correspond to data points in $\mathbf{X}$ that lie in the same subspace as $\boldsymbol{x}$. That is, $c_j \neq 0$ only if $\boldsymbol{x}_j$ is from the same subspace as $\boldsymbol{x}$. Such a representation $\boldsymbol{c}$ is called *subspace-preserving* (a formal and more general definition of subspace-preserving representation is given in Chapter 3). Once a subspace-preserving

representation $\boldsymbol{c}$ is recovered, the label of $\boldsymbol{x}$ is given by the labels of the training data points corresponding to the nonzero entries of $\boldsymbol{c}$. In practice, noise and modeling errors may lead to small nonzero entries associated with data points from other classes. Therefore, $\boldsymbol{x}$ is assigned to the class ℓ which gives the smallest reconstruction error $\|\boldsymbol{x} - \mathbf{X}^\ell \boldsymbol{c}^\ell\|_2$, where $\mathbf{X}^\ell$ (resp., $\boldsymbol{c}^\ell$) denotes the submatrix (resp., subvector) containing columns (resp., entries) corresponding to class ℓ. This method is referred to as the sparse representation based classification (SRC) [177]. The overall algorithm is outlined in Algorithm 2.

One of the earliest demonstrations of the effectiveness of SRC is through the example of face recognition [177]. For each person, training images are collected under various illuminations. Each image is represented as a column vector that contains all its pixel intensity values and all training images are put into columns of a dictionary matrix $\mathbf{X}$. Since the images of a single face under various illumination conditions lie approximately in a low-dimensional linear subspace [18], the columns of the dictionary $\mathbf{X}$ lie approximately in a union of subspaces with each subspace corresponding to one face. Following the steps of SRC, a test image $\boldsymbol{x}$ is expressed as a sparse linear combination of the training images, and the membership of $\boldsymbol{x}$ is determined from the representation coefficients.

Note that in principle one can use any sparse recovery algorithm for computing the sparse solution $\boldsymbol{c}$ in step 1 of SRC. In this thesis we will be consider-

ing using either BP or OMP for such purposes, and will refer to the corresponding versions of SRC as SRC-BP and SRC-OMP, respectively.

Algorithm 2 : Sparse representation based classification (SRC)

Input: Training data $\mathbf{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N] \in \mathbb{R}^{D \times N}$ and corresponding labels $\boldsymbol{y} = [y_1, \cdots, y_N] \in \{1, \cdots, n\}^N$, test data $\boldsymbol{x} \in \mathbb{R}^D$ (as in Definition 4).

1: Compute the sparse solution $\boldsymbol{c}$ to the linear system $\boldsymbol{x} = \mathbf{X}\boldsymbol{c}$ via BP or OMP.

2: Compute $r_\ell(\boldsymbol{x}) = \|\boldsymbol{x} - \mathbf{X}^\ell \boldsymbol{c}^\ell\|_2$ for each class $\ell \in \{1, \cdots, n\}$.

3: Assign $\boldsymbol{x}$ to the group $\ell \in \{1, \cdots, n\}$ that maximizes $r_\ell(\boldsymbol{x})$.

Output: Label of $\boldsymbol{x}$.

Compared to the previous subspace classification methods such as the Nearest Subspaces, a significant advantage of SRC is that it does not require an estimation of the subspaces from training data $\mathbf{X}$, making it more robust in practical applications [177]. However, there has been little justification for why and when SRC works. On the one hand, theoretical analysis for sparse signal recovery have been established in compressed sensing, where spark (Definition 1), mutual coherence (Definition 2) and RIP (Definition 3) are identified as key conditions for guaranteeing the correctness of sparse recovery. From this perspective, it is rather surprising that SRC works, since the data matrix $\mathbf{X}$ which is drawn from a union of subspaces do not necessarily satisfy the spark, incoherence, and RIP conditions. For example, the training set can contain face images that are very similar (e.g. images of the same face under similar

illumination conditions) so that the dictionary violates the incoherence condition. The gap between the known theory for sparse recovery and the empirical success of SRC calls for novel analyses for its correctness.

Finally, we mention that there are several existing theoretical studies [178, 187] for establishing the correctness of SRC. However, these works do not model data as coming from a union of subspaces. In fact, the analysis in [178, 187] is based upon sparse signal recovery theories, and therefore their results are not applicable to coherent dictionary. Another related work [173] studies the decision boundary and margin of the SRC classifier. However, the decision boundary is complicated due to the nonlinear mapping induced by sparse coding. Therefore, such analysis does not provide clear geometric interpretations.

2.4 Sparse subspace clustering (SSC)

Subspace clustering is the problem of learning a union of subspaces from data but without knowing which points belong to which subspaces. More formally, the subspace clustering problem is defined as follows.

Definition 5 (Subspace clustering)**.** Let $\mathbf{X} \in \mathbb{R}^{D \times N}$ be a matrix whose columns lie in a union of n subspace $\cup_{\ell=1}^{n} \mathcal{S}_\ell$. Assume that the membership of each column of $\mathbf{X}$ to the subspace it belongs to is *unknown*. The goal of

> subspace clustering is to segment the data points X to their respective subspaces.

Compared to subspace classification, the task of subspace clustering is more challenging as there is no training data for which labels are given thus no explicit information regarding the subspaces that the data points lie in. Classic subspace clustering methods [23, 157, 198] are based on estimating both the position of the subspaces (in terms of a basis for the subspaces) and the membership of each data point in a joint optimization problem. Formulating such an optimization framework requires a good estimation of the dimension of the subspaces and the number of subspaces, which are usually not provided in practical applications. Moreover, algorithms for solving such optimization problems are typically iterative and the convergence to global minimum is not guaranteed [79, 105]. Tremendous efforts have been dedicated to exploring alternative subspace clustering techniques, leading to a vast literature in the study of algebraic-geometry methods [21, 113, 151, 155, 163], statistical methods [8, 77, 112, 185], spectral methods [11, 39, 61, 73, 80, 184, 199], and so on. For an extensive account of these methods the reader is referred to [161, 164].

An important breakthrough in the area of subspace clustering is the work of [59], which proposes a method that combines sparse recovery with spectral clustering techniques. The idea of this method, which is referred to as sparse

subspace clustering (SSC), is similar to that of SRC for subspace classification purposes. That is, each data point $\boldsymbol{x}_j$ in the data matrix $\mathbf{X}$ can be expressed as a sparse linear combination of other data points in $\mathbf{X}$ that are from the same subspace as $\boldsymbol{x}_j$. Mathematically, this can be written as $\boldsymbol{x}_j = \mathbf{X}\boldsymbol{c}_j, c_{jj} = 0$ where $\boldsymbol{c}_j = [c_{1j}, \cdots, c_{Nj}]^\top$ is the vector of representation coefficients with the property that $c_{ij} \neq 0$ only if $\boldsymbol{x}_j$ and $\boldsymbol{x}_i$ are from the same subspace, and $c_{jj} = 0$ is used to exclude $\boldsymbol{x}_j$ itself from the representation. Such a representation vector $\boldsymbol{c}_j$ is referred to as being *subspace-preserving*. Based on this idea, the first step of SSC is to compute a representation matrix denoted by $\mathbf{C} = [\boldsymbol{c}_1, \cdots, \boldsymbol{c}_N]$ using either BP or OMP (see Algorithm 3). To capture the subspace-preserving property of the representation vectors in the columns of $\mathbf{C}$, we use a similarity graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ with vertices $\mathcal{V} = \{\boldsymbol{x}_j\}_{j=1}^N$ corresponding to data points in $\mathbf{X}$, and with the edges $\mathcal{E}$ given by the weight matrix $\mathbf{W} := |\mathbf{C}| + |\mathbf{C}|^\top \in \mathbb{R}^{N \times N}$. Here, the absolute value $|\mathbf{C}|$ is taken element-wise for the matrix $\mathbf{C}$, therefore the weight of the edge that connects $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is give by $w_{ij} = |c_{ij}| + |c_{ji}|$. Note that the weight matrix $\mathbf{W}$ is symmetric and two data points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are connected if and only if either $c_{ij} \neq 0$ or $c_{ji} \neq 0$, or equivalently, either $\boldsymbol{x}_j$ is used in the representation of $\boldsymbol{x}_i$ or vise versa. In particular, if all representation vectors in the columns of $\mathbf{C}$ are subspace-preserving, then the graph $\mathcal{G}$ has the property that every edge with nonzero weight connects two vertices that correspond to data points from the same subspace. Consequently, the set of vertices

$\mathcal{V}$ can be partitioned into multiple connected components and all vertices in each connected component correspond to data points that are from the same subspace. If we assume that there are exactly n connected components in $\mathcal{G}$, then each connected component must correspond to data points from one of the n subspaces. Therefore, clustering of data points in $\mathbf{X}$ can be achieved by finding all connected components of $\mathcal{G}$. (It is also possible that there are two or multiple distinct connected components that correspond to data points from the same subspace, in which case there are altogether more than n connected components in $\mathcal{G}$. Such a phenomenon is discussed in Section 2.5.) In practice, noise and modeling errors may lead to representations that are not exactly subspace-preserving. As a consequence, $\mathcal{G}$ may not have n connected components corresponding to different subspaces, as there may be a small number of edges of $\mathcal{G}$ that connect points from different subspaces. Therefore, spectral clustering techniques [166] are used to obtain a segmentation of data points from the graph $\mathcal{G}$. The overall algorithm is summarized in Algorithm 3. Similar to the case of SRC, the sparse solutions in step 1 can be computed by either BP or OMP, and we refer to the corresponding two versions of SSC as SSC-BP and SSC-OMP.

The success of SSC has led to many relevant works that exploit sparse representation for subspace clustering [55, 100, 101, 129, 140, 156, 182, 188]. Just as sparse recovery theories do not apply to the analysis of SRC due to the fact

Algorithm 3 : Sparse subspace clustering (SSC)

Input: Data matrix $\mathbf{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N] \in \mathbb{R}^{D \times N}$ (as in Definition 5).

1: Compute the sparse solution to $\boldsymbol{c}_j$ to the linear system $\boldsymbol{x}_j = \mathbf{X}\boldsymbol{c}_j, c_{jj} = 0$ via BP or OMP. Set $\mathbf{C} = [\boldsymbol{c}_1, \cdots, \boldsymbol{c}_N]$.

2: Compute the similarity matrix as $\mathbf{W} = |\mathbf{C}| + |\mathbf{C}|^\top$.

3: Compute segmentation from W by spectral clustering.

Output: Label of each data point in X.

that the data matrix $\mathbf{X}$ does not necessarily satisfy the spark, mutual coherence and RIP assumptions, the analysis of SSC and related methods also requires the development of new analytical approaches that are suited for multi-subspace data. In the current literature, there has been a few works in this direction which provide theoretical guarantees for the correctness of SSC. For example, in [59] it is shown that SSC with sparse recovery solved by BP produces subspace-preserving representation vectors when the subspaces are *independent* (see Definition 19). In [139] the correctness is further extended to the more general case where the subspaces could have nontrivial intersections. When the sparse recovery in SSC is solved by OMP in lieu of BP, then the subspace-preserving property of the representation vectors can be guaranteed under an intricate relationship between intra-class properties and inter-class properties [55]. We will provide a more detailed review of these results in Chapter 3.

2.5 Graph connectivity and elastic-net subspace clustering (EnSC)

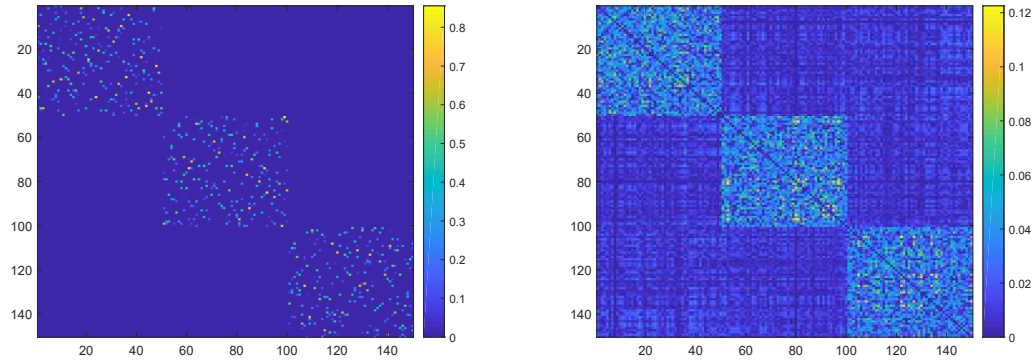
Even though theoretical conditions can be established for BP and OMP to give subspace-preserving solutions, they are not sufficient for SSC to produce correct clustering. This is because the data points from the same subspace may not form a single connected component in the affinity graph $\mathcal{G}$. In such cases, the spectral clustering step of SSC will over-segment points from such a subspace into multiple groups. This is known as the graph connectivity problem of SSC [123, 169]. Intuitively, SSC is prone to suffering from the graph connectivity issue since the representation vectors are sparse, therefore there are very few number of edges in the graph $\mathcal{G}$ constructed from such representation vectors. Theoretically, one could indeed show that there exist examples where over-segmentation happens if the dimension of subspaces is greater than or equal to 4 [123].

As an alternative to SSC, the least squares regression (LSR) methods [84, 111] computes representation vectors $\boldsymbol{c}_j$ under least squares regularizations, i.e., it solves

$$\min_{\boldsymbol{c}_j} \|\boldsymbol{c}_j\|_2^2 \quad \text{s.t. } \boldsymbol{x}_j = \mathbf{X}\boldsymbol{c}_j, c_{jj} = 0. \tag{2.15}$$

The benefit of LSR is that its optimal solution is dense, and therefore it gives

a densely connected similarity graph. However, the representation of LSR is known to be subspace-preserving only when the subspaces are independent, which significantly limits its applicability (see Figure 2.1 for an illustration).



(a) Similarity matrix of SSC-BP **(b)** Similarity matrix of LSR

Figure 2.1: Trade-off between subspace-preserving and connectedness properties in SSC-BP and LSR. 50 points are sampled independently and uniformly at random from 3 subspaces of dimension 5 that are sampled independently and uniformly at random from ambient space of dimension 12. (a) and (b) show the representation matrix C computed by SSC-BP and LSR, respectively. Note that the representation matrix of SSC-BP does not have inter-class connections, but it has very few intra-class connections. In contrast, the representation matrix of LSR has many inter-class connections which may lead to a wrong clustering result, but it also has much denser intra-class connections which may alleviate the connectivity issue in SSC-BP.

To bridge the gap between the subspace-preserving and connectedness properties, [63, 128] propose to use a mixed ℓ_1 and ℓ_2 norm given by $\lambda\|\cdot\|_1 + \frac{1-\lambda}{2}\|\cdot\|_2^2$, and to solve the following optimization problem

$$\min_{\boldsymbol{c}_j} \lambda\|\cdot\|_1 + \frac{1-\lambda}{2}\|\cdot\|_2^2 \quad \text{s.t. } \boldsymbol{x}_j = \mathbf{X}\boldsymbol{c}_j, c_{jj} = 0. \tag{2.16}$$

where $\lambda \in [0, 1]$ controls the trade-off between the two regularizers. In the statistics literature, the optimization program using this regularization is called *Elastic Net* and is used for variable selection in regression problems [202]. Thus we refer to this method as the Elastic Net Subspace Clustering (EnSC). Although EnSC has been shown to perform better than alternative methods in [63, 128], these works do not provide a theoretical justification for the benefits of the method.

Subspace clustering based on other regularizers. There has been a lot of other works on utilizing alternative regularization on the representation coefficients instead of the sparse regularization in SSC, the least squares regularization in LSR and the elastic net regularization in EnSC. For example, low-rank representation [106–109] and low-rank subspace clustering [85, 162] methods use nuclear norm to regularize the representation matrix C. Such low-rankness based methods produce dense solutions in general [172], but suffer from the same drawback as in LSR that they are guaranteed to produce subspace-preserving solutions only if the subspaces are independent. Similar to the idea of EnSC, [172] proposes the low rank sparse subspace clustering (LRSSC) method, which uses a mixed ℓ_1 and nuclear norm regularizer on C to balance between the subspace-preserving and connectedness properties. It has been shown that LRSSC gives a subspace-preserving representation under conditions which are similar to those of SSC. However, the justification for the

improvements in connectivity given by LRSSC over SSC is merely experimental. Other subspace clustering regularizers studied in [110] and [91] use the trace lasso [74] and the k-support norm [9], respectively for achieving a balance between subspace-preserving and connectedness properties. However, no theoretical justification is provided in [91, 110] for the benefit of their methods.

2.6 Open challenges

Despite recent advances in the development of sparse recovery based approaches to subspace classification and subspace clustering, there are many open challenges associated with both the theoretical understanding of the methods and their application to real world data. In particular, we identify the following challenges which will be studied extensively in the rest of this thesis.

2.6.1 Subspace-preserving recovery theory

The key ingredient that leads to the correctness of SRC and SSC is that the sparse representation coefficients are expected to be *subspace-preserving*. That is, the sparse vector $\boldsymbol{c}$ computed in step 1 of Algorithm 2 is expected to have nonzero entries corresponding to points in X that are from the same subspace as $\boldsymbol{x}$. Likewise, the sparse vector $\boldsymbol{c}_j$ computed in step 1 of Algorithm 3 is expected to have nonzero entries corresponding to points in X that are from

the same subspace as $\boldsymbol{x}_j$. To establish the correctness of SRC and SSC for subspace classification and subspace clustering tasks, it is essential to identify the factors that affect subspace-preserving recovery. For example, one would expect that to achieve subspace-preserving recovery, the data points in each of the subspaces need to be *well-behaved* so that they can easily represent each data point in the subspace in terms of a sparse linear combination. In addition, data points from different subspaces need to be well-separated so that sparse representations will pick data points from the same subspace. In terms of the dimension of the subspaces, one would expect that SRC and SSC work the best for small subspace dimensions, as in such cases the sparse representations have fewer number of nonzero entries.

Part of these intuitions have already been validated in a few existing studies for SSC [55, 60, 139]. Particularly, the prominent work [139] has shown that if the points in each subspace are well-distributed in terms of *inradius*, and points from different subspaces are well-separated in terms of *subspace coherence*, then SSC with the sparse recovery problem solved by BP (i.e., SSC-BP) is guaranteed to have subspace-preserving solutions. Moreover, by using a probabilistic model to generate the subspaces and the data, [139] also studies the effect of subspace dimension on the correctness of SSC-BP and shows that the subspace dimensions need to be sufficiently small relative to the ambient dimension in order to guarantee that the subspace preserving property holds

with high probability. As for SSC-OMP, theoretical analysis for its correctness has also appeared in the literature, see e.g. [55]. Nonetheless, previous results are far from being systematic and complete. For example, the studies in [139] and [55], which studied SSC-BP and SSC-OMP respectively, use different quantities to characterize the separation of different subspaces, making it difficult to compare the conditions for subspace-preserving recovery. Moreover, there has been no prior result regarding the effect of subspace dimension for SSC-OMP. Furthermore, it is also surprising that none of previous works has studied conditions for subspace-preserving recovery in the case of SRC. As we will see in Chapter 3, all these challenges are interrelated and they together call for a unified theoretical study of subspace-preserving recovery.

Graph connectivity in subspace clustering. As we have discussed in Section 2.5, the connectivity issue in SSC refers to the problem that even if the sparse solution $\boldsymbol{c}_j$ is always subspace-preserving, there is no guarantee that all points in the same subspace form a connected component of the affinity graph. As a consequence, the final clustering assignment (i.e., the output of spectral clustering) may over-segment points from the same cluster into multiple clusters. To address the connectivity issue, [169] proposes a post-processing procedure and proved its correctness. However, such an approach is not reliable in practice as it is very sensitive to erroneous connections in the data affinity.

The fundamental challenge regarding graph connectivity is that there is

a trade-off between the subspace-preserving property and the density of the representation coefficient vector. On the one hand, SSC is guaranteed to produce subspace-preserving solutions under broad conditions (as we will see in Chapter 3), but the solution is induced to be sparse. On the other hand, least squares and nuclear norm regularization based subspace clustering methods [64, 107, 111, 162] produce representations that are dense in general, but such representations are known to be subspace-preserving only under the restricted condition that the subspaces are independent. To some extent, achieving the subspace-preserving property and dense representations are conflicting goals: if the connections are few, it is more likely that the solution is subspace-preserving, but the similarity graph of each cluster is not well connected. Conversely, as one builds more connections, it becomes more likely that some of them will be incorrect, but the connectivity is improved. To bridge the gap between the subspace preserving and connectedness properties, several works [63, 91, 128, 172] propose to use a weighted sparse and dense inducing regularization. However, no theoretical justification is provided for the benefit of these methods.

2.6.2 Subspace clustering with large-scale data

A practical issue in the application of SSC is that it is limited to small or medium scale datasets and cannot handle real datasets with millions of data

points. The original paper [60] that proposes SSC-BP uses the alternating direction method of multiplier (ADMM) for solving the sparse recovery problem, which requires quadratic memory and has cubic computation complexity in terms of the number of data points. This implies that subspace clustering on a medium scale dataset that contains, say $30{,}000$ data points will require on the order of 10 gigabytes of memory, and our experiments demonstrate that it takes several hours on a standard PC. Therefore, the development of alternative numerical algorithms for solving BP that are more efficient than ADMM is very important for scaling up SSC-BP. On the other hand, SSC-OMP [55] solves the sparse recovery problem by a greedy procedure and is speculated to be more efficient. However, the behavior of SSC-OMP for large-scale problems has not been evaluated in detail. Moreover, the computational complexity of SSC-OMP is still quadratic in the number of data points, therefore it cannot effectively handle datasets containing 1 million points or more.

The majority of prior methods for scaling up SSC are based on subsampling [2, 132] and sketching [147]. Such methods use a small-sized dictionary generated from the given data and expresses each data point as a sparse linear combination of points in this dictionary. In [132], for example, a random sampled subset of the dataset is used as the dictionary. The drawback of such approaches is that there are no theoretical guarantees on the quality of the dictionary for the purpose of subspace clustering. Indeed, the clustering accuracy

is reduced for many cases in the empirical evaluations.

2.6.3 Subspace clustering with outliers

Another issue with existing subspace clustering techniques when applied to real world problems is that practical datasets are often contaminated by points that do not lie in the subspaces, i.e. outliers. In such situations, it is often essential to detect and reject these outliers before any subsequent processing/analysis is performed. SSC in particular can be adversarially affected by outliers. Recall that SSC is based on the idea that each data point in a union of low-dimensional subspaces (i.e., inliers) can be expressed as a linear combination of points from its own subspace, and therefore it is possible to construct a similarity graph in which only points from the same subspace are connected. Outliers in a dataset, on the other hand, are located in the ambient space rather than any low-dimensional subspace, therefore they may generally express themselves as a linear combination of both inliers and outliers. In such cases, inliers from each subspace may be connected to outliers and hence they may no longer form connected components (see Figure 2.2 for an illustration). As a consequence, the spectral clustering step in SSC may no longer identify groups of points from each subspace.

Outlier detection is an important area of machine learning for which a lot of methods have been developed in the past. Traditional methods such as

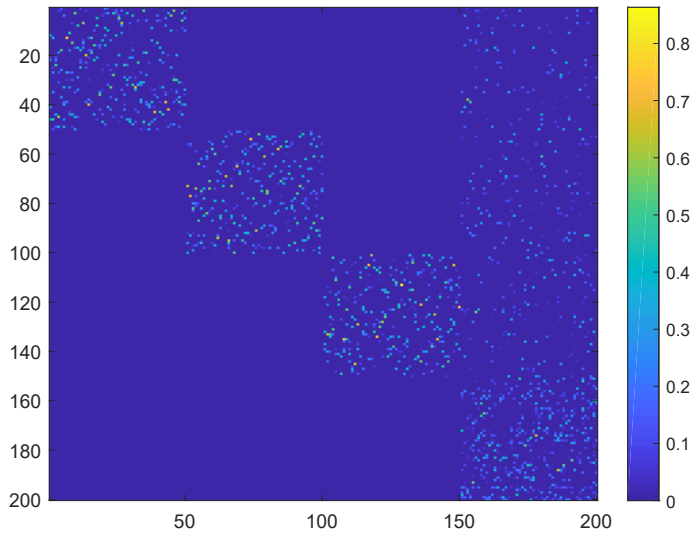


Figure 2.2: Representation matrix of SSC-BP in the presence of outliers. We generate data matrix $\mathbf{X} = [\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \mathbf{X}^{(3)}, \mathbf{X}^{o}]$ where each $\mathbf{X}^{(\ell)}, \ell = 1, 2, 3$ contains 50 points sampled independently and uniformly at random from a subspace $\mathcal{S}_\ell$ of dimension 6 that is sampled uniformly at random from ambient space of dimension 18, and $\mathbf{X}^{o}$ contains 50 points sampled independently and uniformly at random from the ambient space. The picture shows a visualization of $|\mathbf{C}|$ where $\mathbf{C} = [\boldsymbol{c}_1, \cdots, \boldsymbol{c}_N]$ is the representation matrix computed from applying SSC-BP to the data $\mathbf{X}$ and the absolute value is taken entry-wise on the matrix $\mathbf{C}$. We can see that the representation of a point in $\mathbf{X}^{(\ell)}, \ell = 1, 2, 3$ uses points from its own subspace, while the representation of a point in $\mathbf{X}^{o}$ uses points both from $\mathbf{X}^{(\ell)}, \ell = 1, 2, 3$ and $\mathbf{X}^{o}$.

RANSAC [67] and R1-PCA [52] are based on robust statistics. Such methods usually use nonconvex optimization techniques and a good initialization is extremely important for finding the optimal solution. Recently, low-rank and sparse methods that are based on convex optimization techniques are becoming very popular due to their efficient algorithm and provable guarantees. However, many of these methods, such as outlier pursuit [183], REAPER [96] and DPCP [152, 154] model a unique inlier subspace, therefore there performance

for data that has multiple inlier subspaces is unclear. In contrast, very few works have considered the problem of outlier detection in a union of subspaces.

2.6.4 Subspace clustering with imbalanced data

Another challenge in the application of subspace clustering is that practical datasets are often class-imbalanced. For example, a dataset of street signs collected from street view images will contain far more examples of "stop" signs than "unpaved road" signs.

Imbalanced data can significantly compromise the performance of SSC, as one can expect that sparse representation for a data point in an under-represented class is more likely to have nonzero entries corresponding to data points in over-represented classes, leading to false connections in data similarity graph (see Figure 2.3 for an illustration).

The dominant type of approaches for dealing with imbalanced data in machine learning is data sampling. In general, such approaches seek to create a class-balanced dataset from the original dataset by means of oversampling the minority classes, under-sampling the majority classes and so on. However, such approaches are applied to each of the classes in the dataset, and therefore cannot deal with unlabeled data because we do not known which points are from the same class a priori. On the other hand, there are plenty of subset selection methods for generating a representative subset from an unlabeled

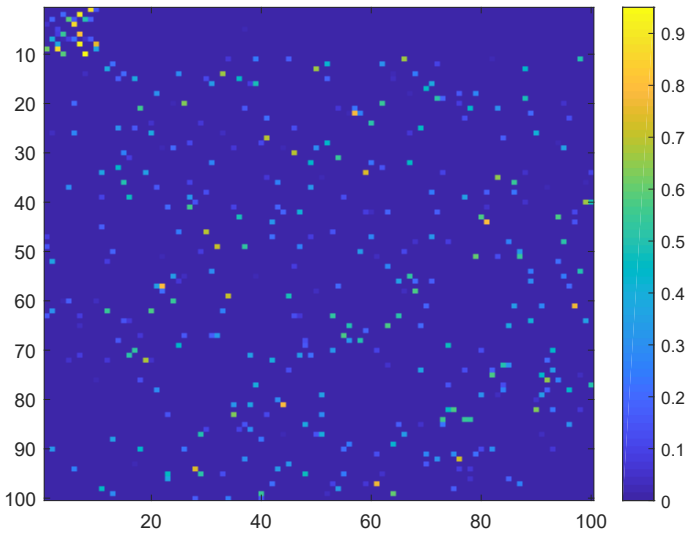


Figure 2.3: Representation matrix of SSC-BP for imbalanced dataset. We generate data matrix $\mathbf{X} = [\mathbf{X}^{(1)}, \mathbf{X}^{(2)}]$ where $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$ respectively contain 10 and 90 points sampled independently and uniformly at random from two subspaces of dimension 4 that are sampled uniformly at random from ambient space of dimension 6. The data $\mathbf{X}$ is imbalanced as the number of data points in $\mathbf{X}^{(2)}$ is 9 times more than that of $\mathbf{X}^{(1)}$. The picture shows a visualization of $|\mathbf{C}|$ where $\mathbf{C} = [\boldsymbol{c}_1, \cdots, \boldsymbol{c}_N]$ is the representation matrix computed from applying SSC-BP to the data $\mathbf{X}$ and the absolute value is taken entry-wise on the matrix $\mathbf{C}$. We can see that the representation of a point in $\mathbf{X}^{(1)}$ not only uses points from its own subspace, but also uses a few points from the other subspace.

dataset, such as Rank Revealing QR [37], Column subset selection [7, 22], and separable Nonnegative Matrix Factorization [12, 90]. However, these methods do not model data as coming from a union of subspaces and there is no evidence that they can select good representatives from such data.

Chapter 3

Subspace-Preserving Recovery Theory

3.1 Problem formulation

Suppose we are given a finite dictionary $\mathcal{A} = \{\boldsymbol{a}_j \in \mathbb{R}^D\}$ that is composed of a subset of points $\mathcal{A}_0 \subseteq \mathcal{A}$ that span a d_0-dimensional subspace $\mathcal{S}_0$, with the remaining points $\mathcal{A}_- := \mathcal{A} \backslash \mathcal{A}_0$ being arbitrary points outside the subspace $\mathcal{S}_0$. Let N_0 and N_- be, respectively, the number of data points in $\mathcal{A}_0$ and $\mathcal{A}_-$. We will use $\mathbf{A}$ to denote the matrix that contains all points from $\mathcal{A}$ as its columns, and likewise for $\mathbf{A}_0$ and $\mathbf{A}_-$. We assume throughout this chapter that all atoms in $\mathcal{A}$ are normalized to have unit ℓ_2 norm.

For any $\boldsymbol{b}$ in subspace $\mathcal{S}_0$, we introduce the concept of *subspace-preserving*

representation which plays a central role in this chapter.

Definition 6 (Subspace-preserving representation)**.** Given a dictionary $\mathcal{A}$ and a vector $\boldsymbol{b}$ as above, a vector $\boldsymbol{c}$ that gives $\boldsymbol{b} = \mathbf{A}\boldsymbol{c}$ is called *subspace-preserving* if $c_j \neq 0$ implies $\boldsymbol{a}_j \in \mathcal{A}_0$.

As we have seen in Chapter 2, a subspace-preserving representation is an important concept in SRC and SSC since it identifies the subspace that the vector $\boldsymbol{b}$ belongs to and consequently guarantees the correctness of these two methods. Specifically, in the subspace classification problem, we can think of $\mathcal{A}_0$ as the set of training data belonging to a particular subspace $\mathcal{S}_0$, and $\mathcal{A}_-$ as the set of training data from all other subspaces. Then, the membership of a point $\boldsymbol{b}$ in the subspace $\mathcal{S}_0$ can be recovered if one can find a subspace-preserving representation of $\boldsymbol{b}$. Likewise, in the subspace clustering problem, we can think of $\boldsymbol{b}$ as a particular data point of the given dataset, $\mathcal{A}_0$ as all other data points in the same subspace as $\boldsymbol{b}$, and $\mathcal{A}_-$ as the set of all data points from all other subspaces. Then, a subspace-preserving representation of $\boldsymbol{b}$ will build connections between $\boldsymbol{b}$ and data points in $\mathcal{A}$ that are from the same subspace as $\boldsymbol{b}$ only.

An important observation for recovering a subspace-preserving representation is that there always exist such representations with at most d_0 nonzero entries. Indeed, one can always find d_0 linearly independent data points from

$\mathcal{A}_0$ which can linearly represent all points $\boldsymbol{b} \in \mathcal{S}_0$. When d_0 is small, such representations are sparse. This suggests that, among all representation vectors $\boldsymbol{c}$ that satisfy $\boldsymbol{b} = \mathbf{A}\boldsymbol{c}$, the sparsest ones are subspace-preserving.

In this chapter, we aim at providing a theoretical justification for such ideas. A fundamental theoretical question is the following.

Question 1. Is the sparsest solution $\boldsymbol{c}$ that satisfies $\boldsymbol{b} = \mathbf{A}\boldsymbol{c}$ subspace-preserving?

In practical algorithms, the problem of finding the sparsest solution is NP-hard in general (see Theorem 2). Therefore, our analysis focuses on approximate sparse representation algorithms such as BP and OMP, which were introduced in Chapter 2. We are interested in the following theoretical question:

Question 2. Under what conditions on the dictionary $\mathcal{A}$ can we find a subspace-preserving representation for an arbitrary point $\boldsymbol{b} \in \mathcal{S}_0$ by BP and OMP?

We further consider the following question, which is answered under a generative probabilistic model for the data:

Question 3. How do the dimension of the subspace d_0 and the number of points inside and outside of the subspace $\mathcal{S}_0$ affect subspace-preserving recovery by BP and OMP?

Intuitively, the dimension d_0 of the subspace needs to be small relative to D so that the subspace-preserving representation is sparse enough. Moreover, the

number of points should be large enough to sufficiently cover the intersection of the subspace with the surface of the unit ball.

In answering the above questions, we distinguish two different but related cases motivated by different application scenarios. In the first case, we consider subspace-preserving recovery of a particular data point $\boldsymbol{b}$ in the subspace, which we refer to as the *instance* recovery problem. The recovery conditions in this case will depend both on the properties of the dictionary $\mathcal{A}$ and the position of the data point $\boldsymbol{b}$. In the second case, we consider subspace-preserving recovery for all possible data point $\boldsymbol{b} \in \mathcal{S}_0$, which we refer to as the *universal* recovery problem. The recovery conditions in this case depends only on the properties of the dictionary $\mathcal{A}$.

3.2 Geometric analysis

3.2.1 Geometric characterization of the dictionary

Our subspace-preserving recovery conditions rely on geometric properties of the dictionary $\mathcal{A}$ that characterize the distribution of points $\mathcal{A}_0$ in subspace $\mathcal{S}_0$. Let $\mathcal{K}(\pm\mathcal{A}_0) := \text{conv}(\pm\mathcal{A}_0)$ where $\text{conv}(\cdot)$ denotes the convex hull. $\mathcal{K}(\pm\mathcal{A}_0)$ is a symmetric convex body according to the following definition.

Definition 7 (Symmetric convex body)**.** A convex set $\mathcal{P}$ that satisfies $\mathcal{P} = -\mathcal{P}$ is called symmetric. A compact convex set with nonempty interior is called a convex body.

The polar set is defined as follows.

Definition 8 (Polar set)**.** The (relative) polar of a set $\mathcal{P}$ is defined as $\mathcal{P}^o := \{\boldsymbol{v} \in \text{span}(\mathcal{P}) : |\langle \boldsymbol{v}, \boldsymbol{a} \rangle| \leq 1, \forall \boldsymbol{a} \in \mathcal{P}\}$.

By this definition, the polar set of $\mathcal{K}(\pm\mathcal{A}_0)$ is given by

$$\mathcal{K}^o(\pm\mathcal{A}_0) := \{\boldsymbol{v} \in \mathcal{S}_0 : |\langle \boldsymbol{v}, \boldsymbol{a} \rangle| \leq 1, \forall \boldsymbol{a} \in \mathcal{K}(\pm\mathcal{A}_0)\} \tag{3.1}$$

In particular, $\mathcal{K}^o(\pm\mathcal{A}_0)$ is a symmetric convex body, as the polar of a convex body is also a convex body [24].

The polar set $\mathcal{K}^o(\pm\mathcal{A}_0)$ can be written in the following equivalent form:

$$\mathcal{K}^o(\pm\mathcal{A}_0) = \{\boldsymbol{v} \in \mathcal{S}_0 : |\langle \boldsymbol{v}, \boldsymbol{a} \rangle| \leq 1, \forall \boldsymbol{a} \in \pm\mathcal{A}_0\}. \tag{3.2}$$

This equivalency can be seen from the following lemma.

Lemma 1. *For an arbitrary set $\mathcal{A}_0 \subseteq \mathcal{S}_0 \subseteq \mathbb{R}^D$, we have*

$$\{\boldsymbol{v} \in \mathcal{S}_0 : |\langle \boldsymbol{v}, \boldsymbol{a} \rangle| \leq 1, \forall \boldsymbol{a} \in \mathcal{K}(\pm\mathcal{A}_0)\} = \{\boldsymbol{v} \in \mathcal{S}_0 : |\langle \boldsymbol{v}, \boldsymbol{a} \rangle| \leq 1, \forall \boldsymbol{a} \in \pm\mathcal{A}_0\}. \tag{3.3}$$

Proof. It is clear that $\{\boldsymbol{v} \in \mathcal{S}_0 : |\langle \boldsymbol{v}, \boldsymbol{a} \rangle| \leq 1, \forall \boldsymbol{a} \in \mathcal{K}(\pm\mathcal{A}_0)\} \subseteq \{\boldsymbol{v} \in \mathcal{S}_0 : |\langle \boldsymbol{v}, \boldsymbol{a} \rangle| \leq 1, \forall \boldsymbol{a} \in \pm\mathcal{A}_0\}$. To see the other direction, we take any $\bar{\boldsymbol{v}} \in \mathcal{S}_0$ that satisfies $|\langle \bar{\boldsymbol{v}}, \boldsymbol{a} \rangle| \leq 1, \forall \boldsymbol{a} \in \pm\mathcal{A}_0$, and show that $\bar{\boldsymbol{v}} \in \{\boldsymbol{v} \in \mathcal{S}_0 : |\langle \boldsymbol{v}, \boldsymbol{a} \rangle| \leq 1, \forall \boldsymbol{a} \in \mathcal{K}(\pm\mathcal{A}_0)\}$. Take an arbitrary $\bar{\boldsymbol{a}} \in \mathcal{K}(\pm\mathcal{A}_0)$. We only need to show that $|\langle \bar{\boldsymbol{v}}, \bar{\boldsymbol{a}} \rangle| \leq 1$. Recall that $\mathcal{K}(\pm\mathcal{A}_0)$ is the convex hull of points from $\pm\mathcal{A}_0$. Therefore, there exist $\{c_j^+ \geq 0\}_{j:\boldsymbol{a}_j \in \mathcal{A}_0}$ and $\{c_j^- \geq 0\}_{j:\boldsymbol{a}_j \in \mathcal{A}_0}$ such that $\sum_{j:\boldsymbol{a}_j \in \mathcal{A}_0}(c_j^+ + c_j^-) = 1$ and $\bar{\boldsymbol{a}} = \sum_{j:\boldsymbol{a}_j \in \mathcal{A}_0}(c_j^+ \boldsymbol{a}_j - c_j^- \boldsymbol{a}_j)$. Using this fact, we have

$$
\begin{aligned}
|\langle \bar{\boldsymbol{v}}, \bar{\boldsymbol{a}} \rangle| &= |\langle \bar{\boldsymbol{v}}, \sum_{j:\boldsymbol{a}_j \in \mathcal{A}_0} (c_j^+ \boldsymbol{a}_j - c_j^- \boldsymbol{a}_j) \rangle| = \sum_{j:\boldsymbol{a}_j \in \mathcal{A}_0} |c_j^+ - c_j^-| \cdot |\langle \bar{\boldsymbol{v}}, \boldsymbol{a}_j \rangle| \\
&\leq \sum_{j:\boldsymbol{a}_j \in \mathcal{A}_0} |c_j^+ - c_j^-| \leq \sum_{j:\boldsymbol{a}_j \in \mathcal{A}_0} (|c_j^+| + |c_j^-|) = \sum_{j:\boldsymbol{a}_j \in \mathcal{A}_0} (c_j^+ + c_j^-) \\
&= 1.
\end{aligned} \tag{3.4}
$$

This finishes the proof. □

We now introduce three concepts for characterizing the distribution of points $\mathcal{A}_0$. The first concept is the inradius of a convex body.

Definition 9 (Inradius)**.** The (relative) inradius of a convex body $\mathcal{P}$, denoted by $r(\mathcal{P})$, is defined as the radius of the largest Euclidean ball in the space $\text{span}(\mathcal{P})$ inscribed in $\mathcal{P}$.

To formulate our results, we will be using the inradius $r(\mathcal{K}(\pm\mathcal{A}_0))$. Intuitively, $r(\mathcal{K}(\pm\mathcal{A}_0))$ characterizes how well data points $\mathcal{A}_0$ are distributed in $\mathcal{S}_0$:

if all atoms in $\mathcal{A}_0$ are well distributed across all directions in subspace $\mathcal{S}_0$ then the inradius is large, while if the atoms are skewed towards certain directions the inradius is small.

The second concept is *covering radius*, which is equivalent to inradius (see Theorem 9) and in some occasions has better geometric interpretations and is easier to work with. Let $\mathbb{S}^{D-1} := \{\boldsymbol{v} \in \mathbb{R}^D : \|\boldsymbol{v}\|_2 = 1\}$ be the set of unit vectors in $\mathbb{R}^D$. Let $\theta(\boldsymbol{v}, \boldsymbol{w}) := \cos^{-1}\langle \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}, \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|_2} \rangle$ be the spherical distance (acute angle) between points $\{\boldsymbol{v}, \boldsymbol{w}\} \subset \mathbb{R}^D \setminus \{\boldsymbol{0}\}$. It is known that $\theta(\cdot, \cdot)$ defines a metric on $\mathbb{S}^{D-1}$ [29]. With an abuse of notation, we define $\theta(\mathcal{V}, \mathcal{W}) := \inf_{\boldsymbol{w} \in \mathcal{W} \setminus \{\boldsymbol{0}\}, \boldsymbol{v} \in \mathcal{V} \setminus \{\boldsymbol{0}\}} \theta(\boldsymbol{v}, \boldsymbol{w})$, where $\mathcal{V}$ and $\mathcal{W}$ are subsets of $\mathbb{R}^D$. The covering radius is defined as follows.

Definition 10 (Covering radius)**.** The (relative) covering radius of a set of points $\mathcal{V}$ is defined as

$$\gamma(\mathcal{V}) := \max\{\theta(\mathcal{V}, \boldsymbol{w}) : \boldsymbol{w} \in \operatorname{span}(\mathcal{V}) \cap \mathbb{S}^{D-1}\}.$$

We will be working with the covering radius $\gamma(\pm\mathcal{A}_0)$. By Definition 10, the covering radius is computed by finding a point $\boldsymbol{w}$ on the unit sphere of $\mathcal{S}_0$ that is furthest away from all the points in $\pm\mathcal{A}_0$ (see Figure 3.1). It can also be interpreted as the smallest radius such that closed spherical balls of that radius centered at the points of $\pm\mathcal{A}_0$ cover all points in $\mathbb{S}^{D-1} \cap \mathcal{S}_0$. Thus, the covering radius characterizes how well the points in $\pm\mathcal{A}_0$ are distributed, without

leaving a large patch of empty region unfilled by any point from $\pm\mathcal{A}_0$.

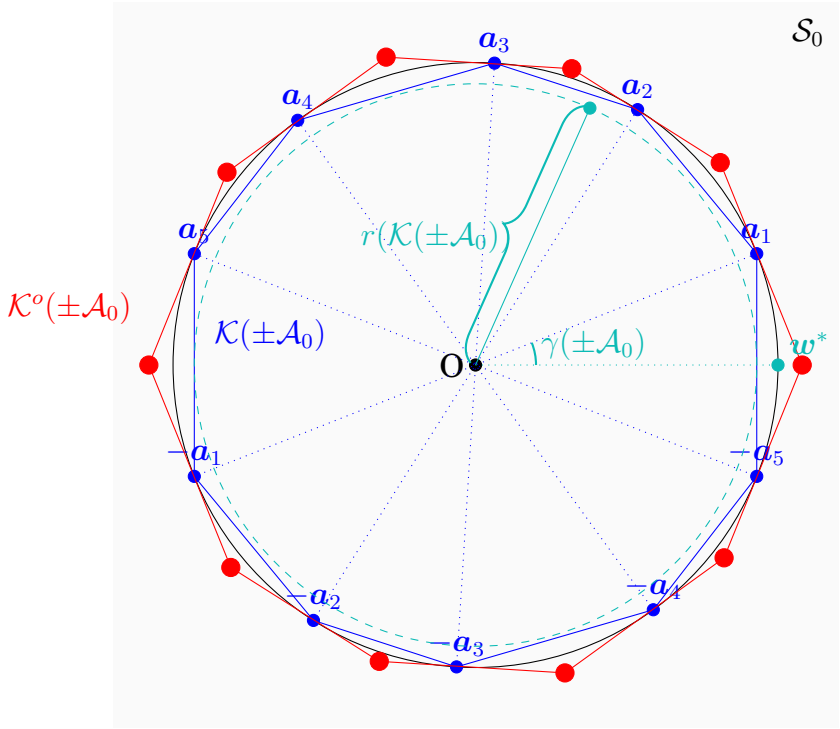


Figure 3.1: Illustration of the geometric characterization of the dictionary. The atoms $\mathcal{A}_0 := \{\boldsymbol{a}_j\}_{j=1}^5$ lie on the unit circle of a two-dimensional subspace $\mathcal{S}_0$. The set $\mathcal{K}(\pm\mathcal{A}_0)$ is illustrated as the blue polygon, and the polar set $\mathcal{K}^o(\pm\mathcal{A}_0)$ is illustrated as the red polygon. In the definition of the inradius $r(\mathcal{K}(\pm\mathcal{A}_0))$, the inscribing ball of $\mathcal{K}(\pm\mathcal{A}_0)$ is shown as the light blue dashed circle, and $r(\mathcal{K}(\pm\mathcal{A}_0))$ is the radius of this circle. In the definition of the covering radius $\gamma(\pm\mathcal{A}_0)$, the maximizer of $\theta(\pm\mathcal{A}_0, \boldsymbol{w}) : \boldsymbol{w} \in \mathcal{S}_0 \cap \mathbb{S}^{D-1}$, denoted as $\boldsymbol{w}^*$, is shown as the light blue point on the rightmost of the figure, and $\gamma(\pm\mathcal{A}_0)$ is the angle between $\boldsymbol{w}^*$ and its closest neighbor (which are $\boldsymbol{a}_1$ and $-\boldsymbol{a}_5$ in this case).

The third characterization of the distribution of $\mathcal{A}_0$ is in terms of the circumradius of the polar set $\mathcal{K}^o(\pm\mathcal{A}_0)$.

> **Definition 11** (Circumradius)**.** The circumradius $R(\mathcal{P})$ of a convex body $\mathcal{P}$ is defined as the radius of the smallest ball containing $\mathcal{P}$.

The inradius $r(\mathcal{K}(\pm\mathcal{A}_0))$, covering radius $\gamma(\pm\mathcal{A}_0)$, and circumradius $R(\mathcal{K}^o(\pm\mathcal{A}_0))$ are illustrated in Figure 3.1.

The following result gives a relationship between the inradius of a set and the circumradius of its polar set.

Theorem 7 (Relation between inradius and circumradius [139])**.** *Let $\mathcal{P}$ be a symmetric convex body and let $\mathcal{P}^o$ be its polar. Then we have $r(\mathcal{P})R(\mathcal{P}^o) = 1$.*

Applying this result to $\mathcal{K}(\pm\mathcal{A}_0)$ we get

$$r(\mathcal{K}(\pm\mathcal{A}_0)) \cdot R(\mathcal{K}^o(\pm\mathcal{A}_0)) = 1. \tag{3.5}$$

The following result shows that the circumradius is also related to covering radius.

Theorem 8 (Relation between circumradius and covering radius)**.** *Given any $\mathcal{A}_0$, we have*

$$R(\mathcal{K}^o(\pm\mathcal{A}_0)) = 1/\cos\gamma(\pm\mathcal{A}_0). \tag{3.6}$$

Proof. From the definition of $\mathcal{K}^o(\pm\mathcal{A}_0)$ in (3.1) we have

$$R(\mathcal{K}^o(\pm\mathcal{A}_0)) = \max_{\substack{\|\mathbf{A}_0^\top \boldsymbol{v}\|_\infty \leq 1 \\ \boldsymbol{v} \in \mathcal{S}_0}} \|\boldsymbol{v}\|_2. \tag{3.7}$$

On the other hand, from the definition of γ_0 we have

$$\gamma(\pm\mathcal{A}_0) = \max_{\boldsymbol{w}\in\mathcal{S}_0\cap\mathbb{S}^{D-1}} \theta(\pm\mathcal{A}_0, \boldsymbol{w}) = \max_{\boldsymbol{w}\in\mathcal{S}_0\cap\mathbb{S}^{D-1}} \inf_{\boldsymbol{v}\in\pm\mathcal{A}_0} \theta(\boldsymbol{v}, \boldsymbol{w}). \tag{3.8}$$

By taking the cosine of both sides of (3.8) we have

$$\cos(\gamma_0) = \min_{\boldsymbol{w}\in\mathcal{S}_0\cap\mathbb{S}^{D-1}} \sup_{\boldsymbol{v}\in\pm\mathcal{A}_0} \cos(\theta(\boldsymbol{v}, \boldsymbol{w})) = \min_{\boldsymbol{w}\in\mathcal{S}_0\cap\mathbb{S}^{D-1}} \|\mathbf{A}_0^\top \boldsymbol{w}\|_\infty. \tag{3.9}$$

To complete the proof, it remains to show that the right hand side of the previous equality and the right hand side of (3.7) are reciprocals of each other, i.e., to show that

$$\max_{\substack{\|\mathbf{A}_0^\top \boldsymbol{v}\|_\infty \leq 1 \\ \boldsymbol{v}\in\mathcal{S}_0}} \|\boldsymbol{v}\|_2 = 1/\min_{\boldsymbol{w}\in\mathcal{S}_0\cap\mathbb{S}^{D-1}} \|\mathbf{A}_0^\top \boldsymbol{w}\|_\infty. \tag{3.10}$$

To see this, let $\boldsymbol{v}^*$ and $\boldsymbol{w}^*$ be, respectively, solutions to the left hand side and right hand side of (3.10). We only need to show that $\|\boldsymbol{v}^*\|_2 = 1/\|\mathbf{A}_0^\top \boldsymbol{w}^*\|_\infty$. For a proof by contradiction, first suppose that $\|\boldsymbol{v}^*\|_2 < 1/\|\mathbf{A}_0^\top \boldsymbol{w}^*\|_\infty$. Let $\bar{\boldsymbol{v}} = \frac{\boldsymbol{w}^*}{\|\mathbf{A}_0^\top \boldsymbol{w}^*\|_\infty}$. Note that $\bar{\boldsymbol{v}}$ satisfies the constraints on the left hand side of (3.10), i.e., $\|\mathbf{A}_0^\top \bar{\boldsymbol{v}}\|_\infty \leq 1$ and $\bar{\boldsymbol{v}} \in \mathcal{S}_0$. Moreover, we have

$$\|\bar{\boldsymbol{v}}\|_2 = \frac{\|\boldsymbol{w}^*\|_2}{\|\mathbf{A}_0^\top \boldsymbol{w}^*\|_\infty} = \frac{1}{\|\mathbf{A}_0^\top \boldsymbol{w}^*\|_\infty} > \|\boldsymbol{v}^*\|_2, \tag{3.11}$$

which contradicts the optimality of $\boldsymbol{v}^*$. Therefore, we have proved that $\|\boldsymbol{v}^*\|_2 \geq 1/\|\mathbf{A}_0^\top \boldsymbol{w}^*\|_\infty$. For the other direction, suppose for the purpose of arriving at a

contradiction that $\|\boldsymbol{v}^*\|_2 > 1/\|\mathbf{A}_0^\top \boldsymbol{w}^*\|_\infty$. Let $\bar{\boldsymbol{w}} = \frac{\boldsymbol{v}^*}{\|\boldsymbol{v}^*\|_2}$. Then $\bar{\boldsymbol{w}}$ satisfies the constraint on the right hand side of (3.10). Moreover, we have

$$\|\mathbf{A}_0^\top \bar{\boldsymbol{w}}\|_\infty = \frac{\|\mathbf{A}_0^\top \boldsymbol{v}^*\|_\infty}{\|\boldsymbol{v}^*\|_2} \leq \frac{1}{\|\boldsymbol{v}^*\|_2} < \|\mathbf{A}_0^\top \boldsymbol{w}^*\|_\infty, \tag{3.12}$$

which contradicts the optimality of $\boldsymbol{w}^*$. Therefore, we have proved that $\|\boldsymbol{v}^*\|_2 \leq 1/\|\mathbf{A}_0^\top \boldsymbol{w}^*\|_\infty$.

Combining the above two parts, we have proved that $\|\boldsymbol{v}^*\|_2 = 1/\|\mathbf{A}_0^\top \boldsymbol{w}^*\|_\infty$, or equivalently, that (3.10) is true. This finishes the proof. □

By combining the relations in (3.6) and in (3.6), we have the following result on the equivalency of inradius and covering radius.

Theorem 9 (Relationship between inradius and covering radius)**.** *Given any $\mathcal{A}_0 \subseteq \mathcal{S}_0$, we have*

$$r(\mathcal{K}(\pm\mathcal{A}_0)) = \cos\gamma(\pm\mathcal{A}_0). \tag{3.13}$$

3.2.2 Instance recovery conditions

We start by considering the problem of subspace-preserving recovery for an arbitrary but *fixed* $\boldsymbol{b}$.

3.2.2.1 Instance recovery by BP

Given a dictionary $\mathcal{A} = \mathcal{A}_0 \cup \mathcal{A}_-$ where $\mathcal{A}_0 \subseteq \mathcal{S}_0$ and a signal $\boldsymbol{b} \in \mathcal{S}_0$, BP searches for a subspace-preserving recovery by solving the following optimization problem

$$\min_{\boldsymbol{c}} \|\boldsymbol{c}\|_1 \quad \text{s.t. } \mathbf{A}\boldsymbol{c} = \boldsymbol{b}. \tag{3.14}$$

We will denote the set of all optimal solutions to (3.14) by $\mathrm{BP}(\mathcal{A}, \boldsymbol{b})$. Note that $\mathrm{BP}(\mathcal{A}, \boldsymbol{b})$ may contain more than one element as the solution to (3.14) is not necessarily unique. Note also that under our problem formulation, there always exist a subspace-preserving representation $\boldsymbol{c}$ such that $\mathbf{A}\boldsymbol{c} = \boldsymbol{b}$, therefore the set $\mathrm{BP}(\mathcal{A}, \boldsymbol{b})$ is always non-empty. We will also denote the optimal objective value of (3.14) by $p(\mathcal{A}, \boldsymbol{b})$, i.e. $p(\mathcal{A}, \boldsymbol{b}) = \|\boldsymbol{c}\|_1$, where $\boldsymbol{c} \in \mathrm{BP}(\mathcal{A}, \boldsymbol{b})$.

The Lagrangian function of the optimization problem in (3.14) is given by

$$L(\boldsymbol{c}, \boldsymbol{v}) := \|\boldsymbol{c}\|_1 - \langle \boldsymbol{v}, \mathbf{A}\boldsymbol{c} - \boldsymbol{b} \rangle, \tag{3.15}$$

where $\boldsymbol{v}$ is the dual variable. By the optimality condition, if $\boldsymbol{c}^* \in \mathrm{BP}(\mathcal{A}, \boldsymbol{b})$ is an optimal solution to the (primal) optimization problem in (3.14), then there exists a $\boldsymbol{v}^*$ such that

$$\mathbf{A}^\top \boldsymbol{v}^* \in \partial \|\boldsymbol{c}^*\|_1, \tag{3.16}$$

where $\partial \|\boldsymbol{c}^*\|_1 = \{\boldsymbol{w} : \|\boldsymbol{w}\|_\infty \leq 1 \text{ and } w_j = \mathrm{sgn}(c_j^*) \text{ for } c_j \neq 0\}$ is the subgradient

of the function $\|\boldsymbol{c}\|_1$ evaluated at $\boldsymbol{c}^*$, and $\boldsymbol{v}^*$ is an optimal solution to the dual of the optimization problem (3.14) which is the following

$$\max_{\boldsymbol{v}} \langle \boldsymbol{v}, \boldsymbol{b} \rangle \quad \text{s.t.} \ \|\mathbf{A}^\top \boldsymbol{v}\|_\infty \leq 1, \tag{3.17}$$

Now, from the optimality condition in (3.16) one can see that if $|\langle \boldsymbol{a}_j, \boldsymbol{v}^* \rangle| < 1$ for all $\boldsymbol{a}_j \in \mathcal{A}_-$ then the vector $\boldsymbol{c}^*$ must be subspace-preserving. This suggests that the condition for $\text{BP}(\mathbf{A}, \boldsymbol{b})$ to be subspace-preserving must depend on the dot product between the data points and the solution to the dual optimization problem. This motivates the following lemma.

Lemma 2. *Given a dictionary $\mathcal{A} = \mathcal{A}_0 \cup \mathcal{A}_-$ where $\mathcal{A}_0 \subseteq \mathcal{S}_0$ and a signal $\boldsymbol{b} \in \mathcal{S}_0$, all elements in $\text{BP}(\mathcal{A}, \boldsymbol{b})$ are subspace-preserving representation if*

$$\exists \boldsymbol{v} \in \text{BP}_{\text{Dual}}(\mathcal{A}_0, \boldsymbol{b}), \quad \max_{\boldsymbol{a} \in \mathcal{A}_-} |\boldsymbol{v}^\top \boldsymbol{a}| < 1, \tag{3.18}$$

where $\text{BP}_{\text{Dual}}(\mathcal{A}_0, \boldsymbol{b})$ is the set of all solutions to the following optimization problem

$$\arg\max_{\boldsymbol{v}} \langle \boldsymbol{v}, \boldsymbol{b} \rangle \quad \text{s.t.} \ \|\mathbf{A}_0^\top \boldsymbol{v}\|_\infty \leq 1. \tag{3.19}$$

Proof. Let $\boldsymbol{v}^* \in \text{BP}_{\text{Dual}}(\mathcal{A}_0, \boldsymbol{b})$ be any point that satisfies the condition

$$\max_{\boldsymbol{a} \in \mathcal{A}_-} |\boldsymbol{v}^{*\top} \boldsymbol{a}| < 1. \tag{3.20}$$

Consider the following optimization problem

$$\min_{\boldsymbol{c}_0} \|\boldsymbol{c}_0\|_1 \quad \text{s.t. } \boldsymbol{b} = \mathbf{A}_0 \boldsymbol{c}_0. \tag{3.21}$$

The solution set of (3.21) is non-empty since $\boldsymbol{b} \in \mathcal{S}_0 = \text{span}(\mathbf{A}_0)$. Let $\boldsymbol{c}_0^*$ be any of its optimal solution. Note that the dual problem of (3.21) is given by (3.19). Therefore, by strong duality we have $\|\boldsymbol{c}^*\|_1 = \langle \boldsymbol{v}^*, \boldsymbol{b} \rangle$.

We now consider the optimization problem in (3.14) and show that all elements in its solution set $\text{BP}(\mathbf{A}, \boldsymbol{b})$ are subspace-preserving. Without the loss of generality, we assume $\mathbf{A} = [\mathbf{A}_0, \mathbf{A}_-]$. First, note that a feasible solution to (3.14) is the vector $\tilde{\boldsymbol{c}} = [(\boldsymbol{c}_0^*)^\top, \mathbf{0}^\top]^\top$ where $\mathbf{0}$ is a vector of all zeros of appropriate size. To prove by contradiction, assume that there is an optimal solution $\bar{\boldsymbol{c}} = [\bar{\boldsymbol{c}}_0^\top, \bar{\boldsymbol{c}}_-^\top]^\top$ where $\bar{\boldsymbol{c}}_- \neq \mathbf{0}$, i.e. $\bar{\boldsymbol{c}}$ is not subspace-preserving. Note that $\mathbf{A}\bar{\boldsymbol{c}} = \boldsymbol{b}$. We have

$$\|\tilde{\boldsymbol{c}}\|_1 = \|\boldsymbol{c}_0^*\|_1 = \langle \boldsymbol{v}^*, \boldsymbol{b} \rangle = \langle \boldsymbol{v}^*, \mathbf{A}\bar{\boldsymbol{c}} \rangle = \langle \boldsymbol{v}^*, \mathbf{A}_0 \bar{\boldsymbol{c}}_0 \rangle + \langle \boldsymbol{v}^*, \mathbf{A}_- \bar{\boldsymbol{c}}_- \rangle \tag{3.22}$$

$$= \langle \mathbf{A}_0^\top \boldsymbol{v}^*, \bar{\boldsymbol{c}}_0 \rangle + \langle \mathbf{A}_-^\top \boldsymbol{v}^*, \bar{\boldsymbol{c}}_- \rangle \leq \|\mathbf{A}_0^\top \boldsymbol{v}^*\|_\infty \cdot \|\bar{\boldsymbol{c}}_0\|_1 + \|\mathbf{A}_-^\top \boldsymbol{v}^*\|_\infty \cdot \|\bar{\boldsymbol{c}}_-\|_1. \tag{3.23}$$

Note that $\|\mathbf{A}_0^\top \boldsymbol{v}^*\|_\infty \leq 1$ from the definition of $\boldsymbol{v}^*$. Note also that $\|\mathbf{A}_-^\top \boldsymbol{v}^*\|_\infty < 1$ from (3.20). Hence, we have $\|\tilde{\boldsymbol{c}}\|_1 < \|\bar{\boldsymbol{c}}_0\|_1 + \|\bar{\boldsymbol{c}}_-\|_1 = \|\bar{\boldsymbol{c}}\|_1$, which is a contradiction of the fact that $\bar{\boldsymbol{c}} \in \text{BP}(\mathcal{A}, \boldsymbol{b})$. □

Lemma 2 shows that BP produces subspace-preserving solutions if there exists a dual optimal solution in $\mathrm{BP}_{\mathrm{Dual}}(\mathcal{A}_0, \boldsymbol{b})$ such that the absolute value of the dot product between the dual solution and all points in $\mathcal{A}_{-}$ is smaller than one. Such a condition, however, is not very insightful from the perspective of subspace learning as it does not directly rely on the geometry of the problem. In order to derive conditions for subspace-preserving recovery that capture the relative configuration of the subspace relative to the points in $\mathcal{A}_{-}$ or the distribution of data points in $\mathcal{A}_0$, we first rewrite the condition in (3.18) as follows:

$$\exists \boldsymbol{v} \in \mathrm{BP}_{\mathrm{Dual}}(\mathcal{A}_0, \boldsymbol{b}), \quad \max_{\boldsymbol{a} \in \mathcal{A}_{-}} |\langle \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}, \boldsymbol{a} \rangle| < \frac{1}{\|\boldsymbol{v}\|_2}. \tag{3.24}$$

Here, the point $\frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}$ and all points in $\boldsymbol{a} \in \mathcal{A}_{-}$ have unit ℓ_2 norm (recall we assume that all points in $\mathcal{A}$ have unit ℓ_2 norm). Therefore, the left hand side of (3.24) captures the similarity (in angle) between the dual solution $\boldsymbol{v}$ and all points in $\mathcal{A}_{-}$. However, the geometric interpretation of this similarity is still unclear as it is unknown where the point $\boldsymbol{v}$ is. Moreover, it is also unclear what the norm $\|\boldsymbol{v}\|_2$ on the right hand side of the condition (3.24) entails.

In order to derive an upper bound on $\|\boldsymbol{v}\|_2$ or equivalently a lower bound on the right hand side of (3.24), we observe from the definition of $\mathrm{BP}_{\mathrm{Dual}}(\mathcal{A}_0, \boldsymbol{b})$ that

$$\forall \boldsymbol{v} \in \mathrm{BP}_{\mathrm{Dual}}(\mathcal{A}_0, \boldsymbol{b}), \quad \boldsymbol{v} + \mathcal{S}_0^{\perp} \subseteq \mathrm{BP}_{\mathrm{Dual}}(\mathcal{A}_0, \boldsymbol{b}), \tag{3.25}$$

where $\mathcal{S}_0^\perp$ denotes the orthogonal complement of $\mathcal{S}_0$. This implies that the set $\mathrm{BP}_{\mathrm{Dual}}(\mathcal{A}_0, \boldsymbol{b})$ is composed of a collection of affine subspaces of dimension $D - d_0$ that are perpendicular to the subspace $\mathcal{S}_0$. On the one hand, this result shows that the set $\mathrm{BP}_{\mathrm{Dual}}(\mathcal{A}_0, \boldsymbol{b})$ is unbounded, therefore it is impossible to provide an upper bound on $\|\boldsymbol{v}\|_2$ in general. On the other hand, this result also shows that among all points in $\mathrm{BP}_{\mathrm{Dual}}(\mathcal{A}_0, \boldsymbol{b})$, the ones that lie in the subspace $\mathcal{S}_0$ have the smallest ℓ_2 norm. This observation motivates us to restrict our attention to the solutions in $\mathrm{BP}_{\mathrm{Dual}}(\mathcal{A}_0, \boldsymbol{b})$ that lie in $\mathcal{S}_0$. As we will see, this allows us to derive an upper bound on $\|\boldsymbol{v}\|_2$ in terms of the inradius of the set $\mathcal{K}(\pm\mathcal{A}_0)$.

We first formally introduce the concept of dual points.

Definition 12 (Dual points)**.** Given any $\mathcal{A}_0 \subseteq \mathcal{S}_0$ and any $\boldsymbol{b} \in \mathcal{S}_0$, the set of dual points, denoted as $\mathcal{D}(\mathcal{A}_0, \boldsymbol{b})$, is defined as the set of all solutions to the following optimization problem:

$$\arg\max_{\boldsymbol{v}} \langle \boldsymbol{v}, \boldsymbol{b} \rangle \quad \text{s.t.} \ \ \|\mathbf{A}_0^\top \boldsymbol{v}\|_\infty \leq 1, \boldsymbol{v} \in \mathcal{S}_0. \tag{3.26}$$

Note that the only difference between the optimization problems (3.19) and (3.26) is that (3.26) has the additional constraint $\boldsymbol{v} \in \mathcal{S}_0$. Therefore, we have

$$\mathcal{D}(\mathcal{A}_0, \boldsymbol{b}) = \mathrm{BP}_{\mathrm{Dual}}(\mathcal{A}_0, \boldsymbol{b}) \cap \mathcal{S}_0, \tag{3.27}$$

i.e., $\mathcal{D}(\mathcal{A}_0, \boldsymbol{b})$ is the subset of $\mathrm{BP}_{\mathrm{Dual}}(\mathcal{A}_0, \boldsymbol{b})$ that lies in the subspace $\mathcal{S}_0$. To further understand the structure of the set $\mathcal{D}(\mathcal{A}_0, \boldsymbol{b})$, we use the fact that the optimization problem in (3.26) is a linear program where the constraint set is the polar set $\mathcal{K}^o(\pm\mathcal{A}_0)$ (see Eq. (3.2)). Since $\mathcal{K}^o(\pm\mathcal{A}_0)$ is non-empty and is bounded in the space of $\mathcal{S}_0$, it is known in linear program that the solution set $\mathcal{D}(\mathcal{A}_0, \boldsymbol{b})$ is a *face* of the convex polyhedron $\mathcal{K}^o(\pm\mathcal{A}_0)$ (see e.g. [197]). Therefore, the set of dual points $\mathcal{D}(\mathcal{A}_0, \boldsymbol{b})$ may contain only one point when the optimal face is 0-dimensional (i.e. an extreme point of $\mathcal{K}^o(\pm\mathcal{A}_0)$), and more than one point when the optimal face is 1-dimensional (i.e. an edge of $\mathcal{K}^o(\pm\mathcal{A}_0)$) or higher.

We also comment on the fact that this definition of dual points is different from the definition in [139]. In [139], dual point is defined as the solution to (3.26) that has the minimum ℓ_2 norm. Therefore, when the solution to (3.26) is not unique, there is a unique dual point in the definition of [139] while there are multiple dual points in our definition.

We can now provide an upper bound on the ℓ_2 norm of any dual point $\boldsymbol{v}$ in $\mathcal{D}(\mathcal{A}_0, \boldsymbol{b})$. By the definition of dual points, we know $\boldsymbol{v} \in \mathcal{K}^o(\pm\mathcal{A}_0)$, therefore from the definition of circumradius (i.e., Definition 11) and the result in (3.5) we have

$$\|\boldsymbol{v}\|_2 \leq R(\mathcal{K}^o(\pm\mathcal{A}_0)) = 1/r(\mathcal{K}(\pm\mathcal{A}_0)). \tag{3.28}$$

By combining this result with (3.24) we can readily derive the following theorem, which is the major result for instance recovery for BP.

Theorem 10 (Instance recovery by BP)**.** *Given a dictionary $\mathcal{A} = \mathcal{A}_0 \cup \mathcal{A}_-$ with $\mathcal{A}_0 \subseteq \mathcal{S}_0$ and a signal $\boldsymbol{b} \in \mathcal{S}_0$, all elements in $BP(\mathcal{A}, \boldsymbol{b})$ are subspace-preserving if there exists a dual point $\boldsymbol{v} \in \mathcal{D}(\mathcal{A}_0, \boldsymbol{b})$ such that*

$$r(\mathcal{K}(\pm\mathcal{A}_0)) > |\langle \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}, \boldsymbol{a} \rangle|, \quad \forall \boldsymbol{a} \in \mathcal{A}_-. \tag{3.29}$$

This result is more general than the result stated in [139]. Specifically, [139] requires that the condition (3.29) be satisfied for the *specific* $\boldsymbol{v}$ in $\mathcal{D}(\mathcal{A}_0, \boldsymbol{b})$ that has the minimum ℓ_2 norm, therefore is more restrictive.

Proof. Let $\boldsymbol{v} \in \mathcal{D}(\mathcal{A}_0, \boldsymbol{b})$ be a dual point that satisfies the relation in (3.29). By combining (3.29) with (3.28), we get

$$\frac{1}{\|\boldsymbol{v}\|_2} > |\langle \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}, \boldsymbol{a} \rangle|, \quad \forall \boldsymbol{a} \in \mathcal{A}_-, \tag{3.30}$$

which can be written equivalently as

$$\max_{\boldsymbol{a} \in \mathcal{A}_-} |\boldsymbol{v}^\top \boldsymbol{a}| < 1. \tag{3.31}$$

Now, from (3.27) we know $\mathcal{D}(\mathcal{A}_0, \boldsymbol{b}) \subseteq \mathrm{BP}_{\mathrm{Dual}}(\mathcal{A}_0, \boldsymbol{b})$, therefore we have $\boldsymbol{v} \in \mathrm{BP}_{\mathrm{Dual}}(\mathcal{A}_0, \boldsymbol{b})$. Thus, the conclusion of the theorem follows from applying Lemma 2.

□

3.2.2.2 Instance recovery by OMP

Recall from Chapter 2 that given dictionary $\mathcal{A}$ and signal $\boldsymbol{b}$, OMP computes a sparse representation by a greedy procedure presented in Algorithm 1. In this work, we take termination conditions as $\epsilon = 0$, which will always be reached under the problem formulation of this chapter. That is, at the termination iteration k we will have that the residual $\boldsymbol{v}_{k+1} = \boldsymbol{b} - \mathbf{A}\boldsymbol{c}_{k+1}$ is zero; we then take $\boldsymbol{c}_{k+1}$ as the solution returned by OMP. Note that depending on how ties are broken when the support set is updated in each iteration, OMP may return different solutions. We will denote the set of all such solutions by $\mathrm{OMP}(\mathcal{A}, \boldsymbol{b})$.

Now, it is easy to see that a sufficient condition for $\mathrm{OMP}(\mathcal{A}, \boldsymbol{b})$ to be subspace preserving is that for each k in step 3 of Algorithm 1, the point that maximizes the dot product lies in the same subspace as $\boldsymbol{b}$. Since $\boldsymbol{v}_0 = \boldsymbol{b}$ and $\boldsymbol{v}_1$ is equal to $\boldsymbol{b}$ minus the projection of $\boldsymbol{b}$ onto the subspace spanned by the selected point, say $\hat{\boldsymbol{a}}$, it follows that if $\boldsymbol{b}, \hat{\boldsymbol{a}} \in \mathcal{S}_0$ then $\boldsymbol{v}_1 \in \mathcal{S}_0$. By a simple induction argument, it follows that if all the selected points are in $\mathcal{S}_0$, then so are the residuals $\{\boldsymbol{v}_k\}$. This suggests that the condition for $\mathrm{OMP}(\mathcal{A}, \boldsymbol{b})$ to be subspace-preserving must depend on the dot products between the data points and the set of residuals. This motivates the following definition and lemma.

Definition 13 (Residual points)**.** Given any $\mathcal{A}_0 \subseteq \mathcal{S}_0$ and any $\boldsymbol{b} \in \mathcal{S}_0$, the set of residual points, denoted as $\mathcal{R}(\mathcal{A}_0, \boldsymbol{b})$, is defined as the set of all nonzero residual vectors computed in step 4 of $\mathrm{OMP}(\mathcal{A}_0, \boldsymbol{b})$.

Lemma 3. *Given a dictionary $\mathcal{A} = \mathcal{A}_0 \cup \mathcal{A}_-$ and a signal $\boldsymbol{b}$, all elements in $\mathrm{OMP}(\mathcal{A}, \boldsymbol{b})$ are subspace-preserving representations if*

$$\forall \boldsymbol{v} \in \mathcal{R}(\mathcal{A}_0, \boldsymbol{b}), \quad \max_{\boldsymbol{a} \in \mathcal{A}_-} |\boldsymbol{v}^\top \boldsymbol{a}| < \max_{\boldsymbol{a} \in \mathcal{A}_0} |\boldsymbol{v}^\top \boldsymbol{a}|. \tag{3.32}$$

The proof of this lemma follows straight forwardly by comparing inductively the steps of the procedure $\mathrm{OMP}(\mathcal{A}, \boldsymbol{b})$ and the procedure of the fictitious problem $\mathrm{OMP}(\mathcal{A}_0, \boldsymbol{b})$. The idea is that these two procedures follow the same "path" if the condition of the lemma is satisfied.

Proof. Let k^* be the number of iterations computed by the procedure $\mathrm{OMP}(\mathcal{A}, \boldsymbol{b})$ so that the residual vector $\boldsymbol{v}_{k^*} = \boldsymbol{0}$. We prove that the solution to $\mathrm{OMP}(\mathcal{A}, \boldsymbol{b})$ is subspace-preserving by showing that $\mathcal{W}_{k^*}$ only contains indexes of points from $\mathcal{S}_0$. This is shown by induction, in the way that $\mathcal{W}_k$ contains points from the i-th subspace for every $0 \leq k \leq k^*$.

The set of residual points $\mathcal{R}(\mathcal{A}_0, \boldsymbol{b})$ introduced in Definition 13 plays an essential role in this proof. For notational clarity, we denote $\hat{\boldsymbol{v}}_k$ to be the residual vector generated at iteration k of the algorithm $\mathrm{OMP}(\mathcal{A}_0, \boldsymbol{b})$ (note that this is

the fictitious problem). The residual vectors of $\mathrm{OMP}(\mathcal{A}, \boldsymbol{b})$ are denoted by $\boldsymbol{v}_k$. In the induction, we also show that $\mathrm{OMP}(\mathcal{A}_0, \boldsymbol{b})$ does not terminate at any $k < k^*$, and that $\boldsymbol{v}_k = \hat{\boldsymbol{v}}_k$ whenever $k \leq k^*$.

First, in the case of $k = 0$, the argument that $\mathcal{W}_0$ only contains indexes of points that are from $\mathcal{S}_0$ is trivially satisfied since $\mathcal{W}_0$ is empty. Also, $\boldsymbol{v}_0 = \hat{\boldsymbol{v}}_0$ is satisfied because they are both set to be $\boldsymbol{b}$ in line 1 of Algorithm 1.

Now, given that $\boldsymbol{v}_k = \hat{\boldsymbol{v}}_k$ for some $k < k^*$ and that $\mathcal{W}_k$ contains points only from subspace $\mathcal{S}_0$, we show that $\boldsymbol{v}_{k+1} = \hat{\boldsymbol{v}}_{k+1}$ and that $\mathcal{W}_{k+1}$ contains indexes of points from subspace $\mathcal{S}_0$. This could be shown by noticing that the added entry in step 3 of Algorithm 1 is given by $\underset{j=1,\ldots,N}{\arg\max} |\boldsymbol{a}_j^T \boldsymbol{v}_k|$. Here, since $\boldsymbol{v}_k = \hat{\boldsymbol{v}}_k$, we have that $\boldsymbol{v}_k$ is in the set $\mathcal{R}(\mathcal{A}_0, \boldsymbol{b})$. Then, by using condition (3.32), we know that the $\arg\max$ will give an index that corresponds to a point in $\mathcal{S}_0$. This guarantees that $\mathcal{W}_{k+1}$ only contains points from subspace $\mathcal{S}_0$. Moreover, the picked point is evidently the same as the point picked at iteration k of the $\mathrm{OMP}(\mathcal{A}_0, \boldsymbol{b})$. It then follows from step 4 of Algorithm 1 that the resultant residuals, $\boldsymbol{v}_{k+1}$ and $\hat{\boldsymbol{v}}_{k+1}$, are also equal. In the case of $k + 1 < k^*$, this means that $\boldsymbol{v}_{k+1} = \hat{\boldsymbol{v}}_{k+1} \neq 0$, so the fictitious problem $\mathrm{OMP}(\mathcal{A}_0, \boldsymbol{b})$ does not terminate at this step. This finishes the mathematical induction.

The fact that OMP terminates in at most d_0 iterations follows from the following facts: (i) we have established that $\mathrm{OMP}(\mathcal{A}, \boldsymbol{b})$ produces the same computations as does $\mathrm{OMP}(\mathcal{A}_0, \boldsymbol{b})$; (ii) the collection of vectors selected by $\mathrm{OMP}(\mathcal{A}_0, \boldsymbol{b})$

are linearly independent and contained in subspace $\mathcal{S}_0$; and (iii) the dimension of $\mathcal{S}_0$ is equal to d_0. □

Intuitively, Lemma 3 tells us that if the dot product between the residual points and the data points in $\mathcal{A}_-$ is smaller than the dot product between the residual points and all points in $\mathcal{A}_0$, then OMP gives a subspace-preserving representation. While such a condition is very intuitive from the perspective of OMP, it is not as intuitive from the perspective of subspace learning as it does not rely on the geometry of the problem. Specifically, it does not directly depend on the relative configuration of the subspace or the distribution of the data in the subspaces. In what follows, we present conditions on the subspace and the data that guarantee that the condition in 3 holds.

Theorem 11 (Instance recovery by OMP)**.** *Given a dictionary $\mathcal{A} = \mathcal{A}_0 \cup \mathcal{A}_-$ and a signal $\boldsymbol{b}$, all elements in $OMP(\mathcal{A}, \boldsymbol{b})$ are subspace-preserving if for all points $\boldsymbol{v} \in \mathcal{R}(\mathcal{A}_0, \boldsymbol{b})$ we have*

$$r(\mathcal{K}(\pm\mathcal{A}_0)) > |\langle \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}, \boldsymbol{a} \rangle|, \quad \forall \boldsymbol{a} \in \mathcal{A}_-. \tag{3.33}$$

Proof. Recall from the definition of covering radius that $\gamma(\pm\mathcal{A}_0) = \max\{\theta(\pm\mathcal{A}_0, \boldsymbol{w}) : \boldsymbol{w} \in \mathcal{S}_0 \cap \mathbb{S}^{D-1}\}$. Therefore, for any $\boldsymbol{w} \in \mathcal{S}_0 \cap \mathbb{S}^{D-1}$ it has $\gamma(\pm\mathcal{A}_0) \geq \theta(\pm\mathcal{A}_0, \boldsymbol{w})$. By

taking $\cos$ on both sides we have

$$r(\mathcal{K}(\pm\mathcal{A}_0)) = \cos\gamma(\pm\mathcal{A}_0) \leq \cos\theta(\pm\mathcal{A}_0, \boldsymbol{w}) = \max_{\boldsymbol{a}\in\mathcal{A}_0} |\langle \frac{\boldsymbol{w}}{\|\boldsymbol{w}|_2}, \boldsymbol{a}\rangle|, \tag{3.34}$$

where we have used the relation between inradius and covering radius in (3.13).

Now, take any $\boldsymbol{v} \in \mathcal{R}(\mathcal{A}_0, \boldsymbol{b})$. We know $\frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2} \in \mathcal{S}_0 \cap \mathbb{S}^{D-1}$. By applying the result in (3.34) and using the condition (3.33) we get

$$\max_{\boldsymbol{a}\in\mathcal{A}_0} |\langle \frac{\boldsymbol{v}}{\|\boldsymbol{v}|_2}, \boldsymbol{a}\rangle| \geq r(\mathcal{K}(\pm\mathcal{A}_0)) > \max_{\boldsymbol{a}\in\mathcal{A}_-} |\langle \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}, \boldsymbol{a}\rangle|. \tag{3.35}$$

Then, the conclusion of the theorem follows from Lemma 3.

□

3.2.2.3 Comparison of recovery conditions

Comparing the instance recovery conditions for BP in (3.29) and for OMP in (3.33), we can see that they have the same structure. That is, the left hand sides are both the inradius of the symmetrized convex hull of the data points $\mathcal{A}_0$, while the right hand sides are both the maximum inner product (in absolute value) between $\mathcal{A}_-$ and a certain subset of $\mathcal{S}_0$. The only different is that the condition in (3.29) and in (3.33) use the set of dual points $\mathcal{D}(\mathcal{A}_0, \boldsymbol{b})$ and the set of residual points $\mathcal{R}(\mathcal{A}_0, \boldsymbol{b})$, respectively, in the calculation of the inner product

on their right hand sides. These two sets are not directly related, therefore the conditions (3.29) and (3.33) are not directly comparable. Nonetheless, notice that the number of dual points is 1 in general (i.e., when the solution to $\mathrm{BP}(\mathcal{A}_0, \boldsymbol{b})$ is unique), while the number of residual points is equal to the number of nonzero entries of $\mathrm{OMP}(\mathcal{A}_0, \boldsymbol{b})$ in general (i.e., when the maximizer in step 3 of $\mathrm{OMP}(\mathcal{A}_0, \boldsymbol{b})$ is unique). Therefore, if we assume that the points in $\mathcal{D}(\mathcal{A}_0, \boldsymbol{b})$ and $\mathcal{R}(\mathcal{A}_0, \boldsymbol{b})$ are distributed uniformly at random on the unit sphere of $\mathcal{S}_0$, then the condition in (3.29) is expected to be more likely to be satisfied. We will show that this is indeed the case in Section 3.3.

3.2.3 Universal recovery conditions

We now consider the problem of subspace-preserving recovery for all possible $\boldsymbol{b}$ in subspace $\mathcal{S}_0$.

As we have already established instance recovery conditions in Theorem 10 and Theorem 11, a naive approach to establish universal recovery condition is to apply instance recovery conditions for all $\boldsymbol{b} \in \mathcal{S}_0$. For example, to establish a universal recovery condition for BP, we can require the condition in (3.29) to be satisfied for all $\boldsymbol{b} \in \mathcal{S}_0$. This can be achieved by requiring that

$$r(\mathcal{K}(\pm\mathcal{A}_0)) > |\langle \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}, \boldsymbol{a} \rangle|, \;\; \forall \boldsymbol{a} \in \mathcal{A}_-, \forall \boldsymbol{v} \in \cup_{\boldsymbol{b} \in \mathcal{S}_0} \mathcal{D}(\mathcal{A}_0, \boldsymbol{b}). \tag{3.36}$$

However, this result is not very interpretable since it is unclear what the set $\cup_{\boldsymbol{b}\in\mathcal{S}_0}\mathcal{D}(\mathcal{A}_0,\boldsymbol{b})$ contains. Nonetheless, we do know that for any $\boldsymbol{b}\in\mathcal{S}_0$, we have $\mathcal{D}(\mathcal{A}_0,\boldsymbol{b})\subseteq\mathcal{S}_0$. Therefore, the condition in (3.36) is implied by the following condition

$$r(\mathcal{K}(\pm\mathcal{A}_0))>|\langle\frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2},\boldsymbol{a}\rangle|,\ \ \forall\boldsymbol{a}\in\mathcal{A}_-,\forall\boldsymbol{v}\in\mathcal{S}_0-\{\boldsymbol{0}\}. \tag{3.37}$$

This condition shows that if the inradius $r(\mathcal{K}(\pm\mathcal{A}_0))$ is large enough, and the coherence (i.e. cosine of acute angle) between any two points $\boldsymbol{a}$ and $\boldsymbol{v}$ each taken from $\mathcal{A}_-$ and the subspace $\mathcal{S}_0$, then BP gives subspace-preserving solution for all points $\boldsymbol{b}\in\mathcal{S}_0$. One can apply a similar argument to the condition in (3.33), and arrive at the conclusion that the condition in (3.37) also guarantees subspace-preserving recovery for all points $\boldsymbol{b}\in\mathcal{S}_0$ by OMP. In summary, we can establish the following result for universal recovery condition.

Theorem 12 (Principal recovery condition)**.** *Given a dictionary $\mathcal{A}=\mathcal{A}_0\cup\mathcal{A}_-$ where $\mathcal{A}_0\subseteq\mathcal{S}_0$, all elements in both $\mathrm{BP}(\mathcal{A},\boldsymbol{b})$ and $\mathrm{OMP}(\mathcal{A},\boldsymbol{b})$ are subspace-preserving for any $\boldsymbol{b}\in\mathcal{S}_0$ if the following principal recovery condition (PRC) holds*

$$r(\mathcal{K}(\pm\mathcal{A}_0))>|\langle\frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2},\boldsymbol{a}\rangle|,\ \ \forall\boldsymbol{a}\in\mathcal{A}_-,\forall\boldsymbol{v}\in\mathcal{S}_0-\{\boldsymbol{0}\}. \tag{3.38}$$

Next, we show that PRC can be improved by replacing the $\mathcal{S}_0$ on the right hand side of (3.38) by a finite subset of $\mathcal{S}_0$. Specifically, we define the set of dual points as follows.

Definition 14 (Dual points)**.** The set of dual points of the set $\mathcal{A}_0$, denoted as $\mathcal{D}(\mathcal{A}_0)$, is defined as the extreme points of $\mathcal{K}^o(\pm\mathcal{A}_0)$.

Geometrically, the dual points $\mathcal{D}(\mathcal{A}_0)$ are the vertices of the polyhedron $\mathcal{K}^o(\pm\mathcal{A}_0)$. In Figure 3.1, the dual points are illustrated as the red dots.

We note that the concept of dual points in Definition 14 needs to be distinguished from the concept of dual points in Definition 12. In Definition 12, the set of dual points is denoted as $\mathcal{D}(\mathcal{A}_0, \boldsymbol{b})$ and depends on both $\mathcal{A}_0$ and $\boldsymbol{b}$. In Definition 14, the set of dual points is denoted as $\mathcal{D}(\mathcal{A}_0)$ and is a function of $\mathcal{A}_0$ only. These two definitions of dual points are also closely related. Since any continuous convex function attains its maximum over a compact set at an extreme point of the convex set, we know that for any $\boldsymbol{b} \in \mathcal{S}_0$, there exists a solution to (3.26) that is an extreme point of its constraint set, which is $\mathcal{K}^o(\pm\mathcal{A}_0)$. This implies that for any $\boldsymbol{b} \in \mathcal{S}_0$, there exists a $\boldsymbol{v} \in \mathcal{D}(\mathcal{A}_0, \boldsymbol{b})$ such that $\boldsymbol{v} \in \mathcal{D}(\mathcal{A}_0)$. In particular, if the solution to (3.26) is unique for some $\boldsymbol{b}$, then we have $\mathcal{D}(\mathcal{A}_0, \boldsymbol{b}) \subseteq \mathcal{D}(\mathcal{A}_0)$. On the other hand, if the solution to (3.26) is not unique, then the solution set $\mathcal{D}(\mathcal{A}_0, \boldsymbol{b})$ is a face of $\mathcal{K}^o(\pm\mathcal{A}_0)$. In such cases, we no longer have $\mathcal{D}(\mathcal{A}_0, \boldsymbol{b}) \subseteq \mathcal{D}(\mathcal{A}_0)$.

A core result for universal recovery of subspace-preserving representation is the following.

Theorem 13 (Dual recovery condition)**.** *Given a dictionary $\mathcal{A} = \mathcal{A}_0 \cup \mathcal{A}_-$ where*

$\mathcal{A}_0 \subseteq \mathcal{S}_0$, all elements in both $\mathrm{BP}(\mathcal{A}, \boldsymbol{b})$ and $\mathrm{OMP}(\mathcal{A}, \boldsymbol{b})$ are subspace-preserving for any $\boldsymbol{b} \in \mathcal{S}_0$ if the following dual recovery condition (DRC) holds

$$r(\mathcal{K}(\pm\mathcal{A}_0)) > |\langle \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}, \boldsymbol{a} \rangle|, \quad \forall \boldsymbol{a} \in \mathcal{A}_-, \forall \boldsymbol{v} \in \mathcal{D}(\mathcal{A}_0). \tag{3.39}$$

The set of dual points $\mathcal{D}(\mathcal{A}_0)$ in (3.39) is a finite subset of $\mathcal{S}_0$ (see Lemma 4 below). Therefore, the DRC is implied by the PRC, which makes Theorem 13 a stronger result than Theorem 12. Intuitively, the PRC requires points in $\mathcal{A}_-$ to be sufficiently far away from the entire subspace $\mathcal{S}_0$, while the DRC only requires that $\mathcal{A}_-$ be sufficiently far away from a finite subset of $\mathcal{S}_0$.

Lemma 4. *The set $\mathcal{D}(\mathcal{A}_0)$ is finite. Specifically,*

$$\mathrm{card}(\mathcal{D}(\mathcal{A}_0)) \leq 2^{d_0} \binom{N_0}{d_0}. \tag{3.40}$$

Proof. Consider a linear program with variable $\boldsymbol{v}$, constraint $\boldsymbol{v} \in \mathcal{K}^o(\pm\mathcal{A}_0)$, and arbitrary objective function. Since the dual points $\mathcal{D}(\mathcal{A}_0)$ are precisely the extreme points of $\mathcal{K}^o(\pm\mathcal{A}_0)$, they are the same as the basic feasible solutions of the linear program [126]. Since each basic feasible solution is determined by d_0 linearly independent constraints from among the $2N_0$ constraints of $\|\mathbf{A}_0^\top \boldsymbol{v}\|_\infty \leq 1$, there are at most $2^{d_0}\binom{N_0}{d_0}$ ways to choose such a set of constraints (here we haver used the fact that at most one of the two constraints $-1 \leq (\mathbf{A}_0^\top \boldsymbol{v})_i \leq 1$ can be chosen for each $i \in \{1, 2, \ldots, N_0\}$). $\square$

The bound in Lemma 4 is not tight in general since not every set of constraints in the $2^{d_0}\binom{N_0}{d_0}$ number of combinations produces a basic feasible solution. For example, in Figure 3.1 the combination of constraints $\langle \boldsymbol{a}_1, \boldsymbol{v} \rangle \leq 1$ and $\langle \boldsymbol{a}_2, \boldsymbol{v} \rangle \leq 1$ produces a basic feasible solution, while the combination of constraints $\langle \boldsymbol{a}_1, \boldsymbol{v} \rangle \leq 1$ and $\langle \boldsymbol{a}_3, \boldsymbol{v} \rangle \leq 1$ does not. Nonetheless, this bound is sufficient for the study in this thesis[1].

To interpret these results geometrically, note from the relationship between inradius and covering radius in (3.13) that the PRC is equivalent to the following condition:

$$\gamma(\pm \mathcal{A}_0) < \theta(\mathcal{A}_-, \mathcal{S}_0), \tag{3.41}$$

Therefore, the PRC is satisfied if the angle between any data point in $\mathcal{A}_-$ and the subspace $\mathcal{S}_0$ is larger than the covering radius $\gamma(\pm \mathcal{A}_0)$. Similarly, the DRC in (3.39) can be rewritten as

$$\gamma(\pm \mathcal{A}_0) < \theta(\mathcal{A}_-, \mathcal{D}(\mathcal{A}_0)). \tag{3.42}$$

Therefore, the DRC is satisfied if the angle between any data point in $\mathcal{A}_-$ and any dual point in $\mathcal{D}(\mathcal{A}_0)$ is larger than $\gamma(\pm \mathcal{A}_0)$. Figure 3.2 gives a geometric illustration of the PRC and the DRC using an example of a two dimensional

[1] A tighter bound is possible, see, [69, 127].

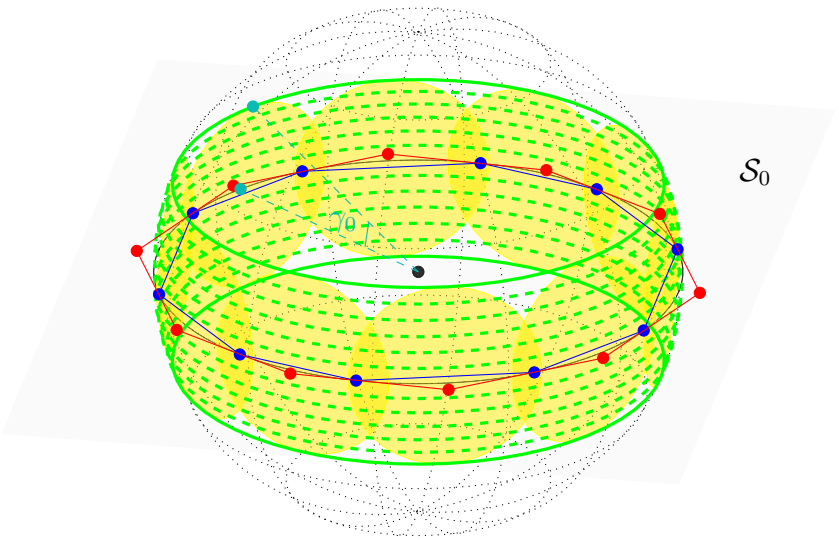


Figure 3.2: Illustration of the geometry associated with subspace-preserving representations. The atoms $\mathcal{A}_0 := \{\boldsymbol{a}_j\}_{j=1}^5$ (drawn in blue) lie on the unit circle of a two-dimensional subspace $\mathcal{S}_0$. The atoms $\mathcal{A}_-$ (not drawn) lie on the unit sphere in the ambient space $\mathbb{R}^3$. The region enclosed by the two solid green circles corresponds to the set $\{\boldsymbol{a} \in \mathbb{R}^3 : \gamma(\pm\mathcal{A}_0) < \theta(\boldsymbol{a}, \mathcal{S}_0)\}$. The region colored in yellow correspond to the set $\{\boldsymbol{a} \in \mathbb{R}^3 : \gamma(\pm\mathcal{A}_0) < \theta(\boldsymbol{a}, \mathcal{D}(\mathcal{A}_0))\}$. The PRC (resp., the DRC) is satisfied if no point from $\mathcal{A}_-$ lies in the former (resp, latter) region.

subspace $\mathcal{S}_0$ in $\mathbb{R}^3$. The dictionary $\mathcal{A}_0$ and the dual points $\mathcal{D}(\mathcal{A}_0)$ are illustrated in blue and red, respectively. Also, see Figure 3.1 for an illustration in the 2D plane of the subspace $\mathcal{S}_0$. The two solid green circles have latitude $\pm\gamma(\pm\mathcal{A}_0)$ on the unit sphere, they illustrate that the PRC holds if and only if the atoms $\mathcal{A}_-$ are such that they do not lie in the region enclosed by these two circles (i.e., they all have latitude larger than $\gamma(\pm\mathcal{A}_0)$ or smaller than $-\gamma(\pm\mathcal{A}_0)$). The DRC is illustrated by the yellow region which is composed of a union of the yellow circles in the space $\mathbb{S}^2$. Each circle is centered at a normalized dual point (note the red dots illustrate the unnormalized dual points) with radius $\gamma(\pm\mathcal{A}_0)$. The

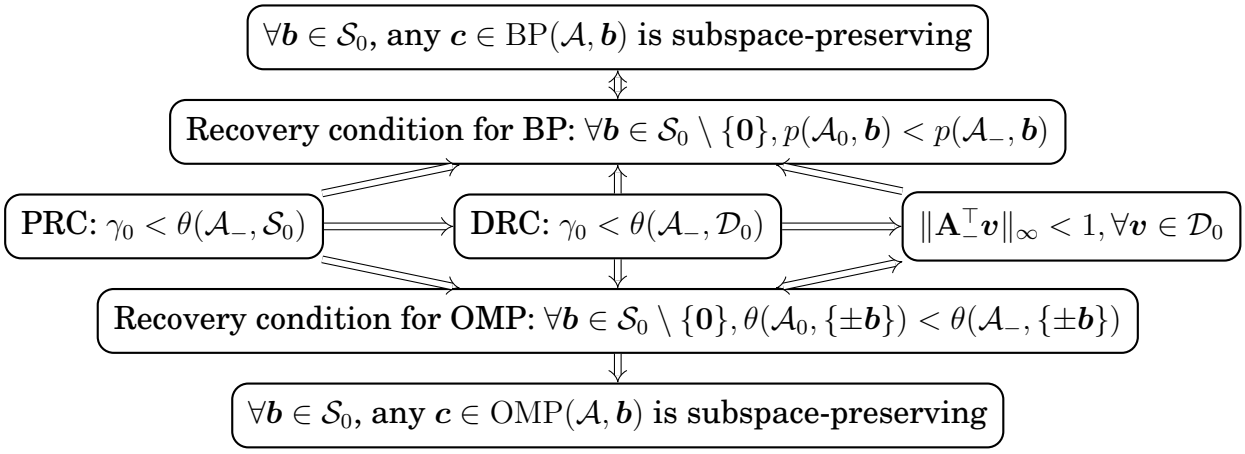


Figure 3.3: Summary of the results of universal recovery conditions with dictionary $\mathcal{A} = \mathcal{A}_0 \cup \mathcal{A}_-$. Each box contains a proposition, and arrows denote implications. The topmost (resp., bottommost) box is the property of subspace-preserving recovery by BP (resp., OMP). Two major conditions for subspace-preserving recovery are the PRC and the DRC.

DRC holds if and only if no point from $\mathcal{A}_-$ lies in the yellow region.

In the following, we provide a detailed analysis of the subspace-preserving representation property of BP and OMP in the deterministic model setting, and formally prove the PRC and the DRC in Theorems 12 and 13, respectively. Our analyses also give several related results. A diagram that summarizes the relation between these results is given in Figure 3.3. To avoid cluttered notations, we will use $\mathcal{K}_0 := \mathcal{K}(\pm\mathcal{A}_0)$, $\mathcal{K}_0^o := \mathcal{K}^o(\pm\mathcal{A}_0)$, $\mathcal{D}_0 := \mathcal{D}(\mathcal{A}_0)$, $r_0 := r(\mathcal{K}(\pm\mathcal{A}_0))$, $R_0 = R(\mathcal{K}^o(\pm\mathcal{A}_0))$, and $\gamma_0 := \gamma(\pm\mathcal{A}_0)$. The topmost and bottommost boxes of Figure 3.3 are, respectively, the subspace-preserving recovery properties for BP and OMP that we are pursuing. Note that both of them are implied by the PRC and the DRC.

3.2.3.1 Universal recovery by BP

We first establish an equivalent condition for subspace-preserving recovery for BP, and then show that this condition is implied by PRC and DRC. See the upper half of Figure 3.3 for an illustration.

A recovery condition for BP. The work of [60] has established a condition that is necessary and sufficient for when $\mathrm{BP}(\mathcal{A}, \boldsymbol{b})$ is subspace-preserving. Here, we rephrase the result for our problem as well as provide a proof for completeness. We first introduce some definitions. Consider the following two optimization problems:

$$\arg\min_{\boldsymbol{c}_0} \|\boldsymbol{c}_0\|_1 \quad \text{s.t.} \;\; \boldsymbol{b} = \mathbf{A}_0 \boldsymbol{c}_0, \tag{3.43}$$

$$\arg\min_{\boldsymbol{c}_-} \|\boldsymbol{c}_-\|_1 \quad \text{s.t.} \;\; \boldsymbol{b} = \mathbf{A}_- \boldsymbol{c}_-. \tag{3.44}$$

Let $\mathrm{BP}(\mathcal{A}_0, \boldsymbol{b})$ and $\mathrm{BP}(\mathcal{A}_-, \boldsymbol{b})$ be the set of all solutions to (3.43) and (3.44), respectively. Since $\boldsymbol{b}$ lies in the span of the set $\mathcal{A}_0$, which is the subspace $\mathcal{S}_0$, we know that the optimization problem (3.43) is always feasible, i.e., the set $\mathrm{BP}(\mathcal{A}_0, \boldsymbol{b})$ is nonempty. On the other hand, $\boldsymbol{b}$ may not lie in the span of $\mathcal{A}_-$, in which case the set $\mathrm{BP}(\mathcal{A}_-, \boldsymbol{b})$ is empty. We denote the objective values that correspond to $\mathrm{BP}(\mathcal{A}_0, \boldsymbol{b})$ and $\mathrm{BP}(\mathcal{A}_-, \boldsymbol{b})$ by $p(\mathcal{A}_0, \boldsymbol{b})$ and $p(\mathcal{A}_-, \boldsymbol{b})$, respectively. In particular, if $\mathrm{BP}(\mathcal{A}_-, \boldsymbol{b})$ is empty then we define $p(\mathcal{A}_-, \boldsymbol{b}) = \infty$.

Theorem 14 (A recovery condition for BP)**.** *[60] Any $\boldsymbol{c} \in \mathrm{BP}(\mathcal{A}, \boldsymbol{b})$ is subspace-preserving for all $\boldsymbol{b} \in \mathcal{S}_0$ if and only if $p(\mathcal{A}_0, \boldsymbol{b}) < p(\mathcal{A}_-, \boldsymbol{b})$ for all $\boldsymbol{b} \in \mathcal{S}_0 \setminus \{\mathbf{0}\}$.*

Proof. Throughout this proof we assume without loss of generality that $\mathbf{A} = [\mathbf{A}_0, \mathbf{A}_-]$.

For the "only if" direction, take any $\boldsymbol{b} \in \mathcal{S}_0 \setminus \{\mathbf{0}\}$. We prove $p(\mathcal{A}_0, \boldsymbol{b}) < p(\mathcal{A}_-, \boldsymbol{b})$ by showing that $p(\mathcal{A}_0, \boldsymbol{b}) \leq p(\mathcal{A}, \boldsymbol{b})$ and that $p(\mathcal{A}, \boldsymbol{b}) < p(\mathcal{A}_-, \boldsymbol{b})$.

We first show $p(\mathcal{A}_0, \boldsymbol{b}) \leq p(\mathcal{A}, \boldsymbol{b})$. Take any $\hat{\boldsymbol{c}} = [(\hat{\boldsymbol{c}}_0)^\top, (\hat{\boldsymbol{c}}_-)^\top]^\top \in \mathrm{BP}(\mathcal{A}, \boldsymbol{b})$. By our current assumption, $\hat{\boldsymbol{c}}$ is subspace-preserving, hence $\hat{\boldsymbol{c}}_- = \mathbf{0}$. Therefore, we have $\boldsymbol{b} = \mathbf{A}\hat{\boldsymbol{c}} = \mathbf{A}_0\hat{\boldsymbol{c}}_0 + \mathbf{A}_-\hat{\boldsymbol{c}}_- = \mathbf{A}_0\hat{\boldsymbol{c}}_0$. This implies that $\hat{\boldsymbol{c}}_0$ is a feasible solution to (3.43), hence $p(\mathcal{A}_0, \boldsymbol{b}) \leq \|\hat{\boldsymbol{c}}_0\|_1$. From $\hat{\boldsymbol{c}}_- = \mathbf{0}$ we also have $\|\hat{\boldsymbol{c}}_0\|_1 = \|\hat{\boldsymbol{c}}\|_1$. Combining these results we see that $p(\mathcal{A}_0, \boldsymbol{b}) \leq \|\hat{\boldsymbol{c}}_0\|_1 = \|\hat{\boldsymbol{c}}\|_1 = p(\mathcal{A}, \boldsymbol{b})$.

We now prove $p(\mathcal{A}, \boldsymbol{b}) < p(\mathcal{A}_-, \boldsymbol{b})$ by using contradiction. Assume that $p(\mathcal{A}, \boldsymbol{b}) \geq p(\mathcal{A}_-, \boldsymbol{b})$. Under this assumption, the set $\mathrm{BP}(\mathcal{A}_-, \boldsymbol{b})$ is nonempty since $p(\mathcal{A}_-, \boldsymbol{b}) \leq p(\mathcal{A}, \boldsymbol{b}) < \infty$. Let $\bar{\boldsymbol{c}}_-$ be any vector in $\mathrm{BP}(\mathcal{A}_-, \boldsymbol{b})$, and let $\bar{\boldsymbol{c}} = [\mathbf{0}_{N_0}^\top, (\bar{\boldsymbol{c}}_-)^\top]^\top$ where $\mathbf{0}_{N_0}$ is a vector of length N_0. We can see that $\mathbf{A}\bar{\boldsymbol{c}} = \mathbf{A}_-\bar{\boldsymbol{c}}_- = \boldsymbol{b}$, which implies that $\bar{\boldsymbol{c}}$ is a feasible solution to (3.14). Furthermore, from the fact that all optimal solutions to (3.14) are subspace-preserving and that $\bar{\boldsymbol{c}}$ is not subspace-preserving, we know that $\bar{\boldsymbol{c}}$ is not an optimal solution to (3.14). Therefore, it must have $p(\mathcal{A}, \boldsymbol{b}) < \|\bar{\boldsymbol{c}}\|_1$. We can now see that $p(\mathcal{A}, \boldsymbol{b}) < \|\bar{\boldsymbol{c}}\|_1 = \|\bar{\boldsymbol{c}}_-\|_1 = p(\mathcal{A}_-, \boldsymbol{b})$, which contradicts the assumption that $p(\mathcal{A}, \boldsymbol{b}) \geq p(\mathcal{A}_-, \boldsymbol{b})$.

For the "if" direction, suppose (for the purpose of reaching a contradiction) that there exists some $\boldsymbol{b} \in \mathcal{S}_0$ and some $\hat{\boldsymbol{c}} = [(\hat{\boldsymbol{c}}_0)^\top, (\hat{\boldsymbol{c}}_-)^\top]^\top \in \mathrm{BP}(\mathcal{A}, \boldsymbol{b})$ such that $\hat{\boldsymbol{c}}$ is not subspace-preserving. From the constraint in (3.14), we have $\mathbf{A}_0\hat{\boldsymbol{c}}_0 + \mathbf{A}_-\hat{\boldsymbol{c}}_- = \boldsymbol{b}$. Let

$$\tilde{\boldsymbol{b}} := \boldsymbol{b} - \mathbf{A}_0\hat{\boldsymbol{c}}_0 = \mathbf{A}_-\hat{\boldsymbol{c}}_-. \tag{3.45}$$

Note that $\tilde{\boldsymbol{b}} \in \mathcal{S}_0$ since it is a linear combination of $\boldsymbol{b}$ and all columns of $\mathbf{A}_0$, all of which lie in $\mathcal{S}_0$. Moreover, we can also see that $\tilde{\boldsymbol{b}} \neq \mathbf{0}$ from the following proof by contradiction. Suppose $\tilde{\boldsymbol{b}} = \mathbf{0}$, in which case we have $\mathbf{A}_0\hat{\boldsymbol{c}}_0 = \boldsymbol{b}$. Let us take $\bar{\boldsymbol{c}} = [(\hat{\boldsymbol{c}}_0)^\top, \mathbf{0}_{N_-}^\top]^\top$. We have $\mathbf{A}\bar{\boldsymbol{c}} = \mathbf{A}_0\hat{\boldsymbol{c}}_0 = \boldsymbol{b}$, which implies that $\bar{\boldsymbol{c}}$ is a feasible solution to (3.14). Moreover, we have $\|\bar{\boldsymbol{c}}\|_1 = \|\hat{\boldsymbol{c}}_0\|_1 < \|\hat{\boldsymbol{c}}_0\|_1 + \|\hat{\boldsymbol{c}}_-\|_1 = \|\hat{\boldsymbol{c}}\|_1$, where we have used the fact that $\hat{\boldsymbol{c}}$ is not subspace-preserving hence $\hat{\boldsymbol{c}}_- \neq 0$. This contradicts the fact that $\hat{\boldsymbol{c}}$ is an optimal solution to (3.14). Therefore, we have established that $\tilde{\boldsymbol{b}} \neq \mathbf{0}$.

Take any $\tilde{\boldsymbol{c}}_0$ from the set $\mathrm{BP}(\mathcal{A}_0, \tilde{\boldsymbol{b}})$ (which is nonempty since $\tilde{\boldsymbol{b}} \in \mathcal{S}_0 = \mathrm{span}(\mathcal{A}_0)$). We have $\|\tilde{\boldsymbol{c}}_0\|_1 = p(\mathcal{A}_0, \tilde{\boldsymbol{b}})$ and $\tilde{\boldsymbol{b}} = \mathbf{A}_0\tilde{\boldsymbol{c}}_0$. We now consider the vector $[(\hat{\boldsymbol{c}}_0 + \tilde{\boldsymbol{c}}_0)^\top, \mathbf{0}]^\top$. This vector is a feasible solution to the problem $\mathrm{BP}(\mathcal{A}, \boldsymbol{b})$ since $\mathbf{A}_0(\hat{\boldsymbol{c}}_0 + \tilde{\boldsymbol{c}}_0) = \mathbf{A}_0\hat{\boldsymbol{c}}_0 + \tilde{\boldsymbol{b}} = \boldsymbol{b}$ (by recalling (3.45)). Moreover, this feasible solution has smaller objective value than the solution $\hat{\boldsymbol{c}} = [(\hat{\boldsymbol{c}}_0)^\top, (\hat{\boldsymbol{c}}_-)^\top]^\top$ since $\|\hat{\boldsymbol{c}}_0 + \tilde{\boldsymbol{c}}_0\|_1 \leq \|\hat{\boldsymbol{c}}_0\|_1 + \|\tilde{\boldsymbol{c}}_0\|_1 < \|\hat{\boldsymbol{c}}_0\|_1 + \|\hat{\boldsymbol{c}}_-\|_1$. To get the last inequality, we have used our current assumption that $p(\mathcal{A}_0, \tilde{\boldsymbol{b}}) < p(\mathcal{A}_-, \tilde{\boldsymbol{b}})$ (since $\tilde{\boldsymbol{b}} \in \mathcal{S}_0 \backslash \{\mathbf{0}\}$) which implies $\|\tilde{\boldsymbol{c}}_0\|_1 < p(\mathcal{A}_-, \tilde{\boldsymbol{b}})$, and the fact that $\hat{\boldsymbol{c}}_-$ is a feasible solution to the

optimization problem

$$\arg\min_{\boldsymbol{c}_-} \|\boldsymbol{c}_-\|_1 \quad \text{s.t. } \tilde{\boldsymbol{b}} = \mathbf{A}_-\boldsymbol{c}_-, \tag{3.46}$$

which implies $p(\mathcal{A}_-, \tilde{\boldsymbol{b}}) \leq \|\hat{\boldsymbol{c}}_-\|_1$. This provides a contradiction to the assumption that $\hat{\boldsymbol{c}}$ is an optimal solution. □

While this result provides a necessary and sufficient condition, it relies on solving the optimization problem and does not reveal explicit properties required of the dictionary $\mathcal{A}$. The PRC and DRC have the advantages that they rely only on the properties of the dictionary and they have good geometric interpretation. They do, however, come with the disadvantage that they are only sufficient conditions.

The PRC result. We proceed to the proof of PRC. As we have seen in the discussion prior to Theorem 12, the PRC result can be proved from the results for instance recovery in Theorem 10 and Theorem 11. However, we believe that a direct proof that PRC implies the equivalent condition established in Theorem 14 offers a clearer understanding of PRC.

In the condition in Theorem 14, the LHS $p(\mathcal{A}_0, \boldsymbol{b})$ depends purely on the properties of $\mathcal{A}_0$, while RHS $p(\mathcal{A}_-, \boldsymbol{b})$ depends on a relation between the atoms $\mathcal{A}_-$ and the subspace $\mathcal{S}_0$. This enlightens us to upper bound the former by characterizing $\mathcal{S}_0$, and to lower bound the latter by using some relationship

between $\mathcal{S}_0$ and $\mathcal{A}_-$.

Theorem 15 (Correctness of BP for subspace-preserving recovery via PRC)**.** *If the PRC* $\gamma_0 < \theta(\mathcal{A}_-, \mathcal{S}_0)$ *holds then* $\forall \boldsymbol{b} \in \mathcal{S}_0 \setminus \{0\}, p(\mathcal{A}_0, \boldsymbol{b}) < p(\mathcal{A}_-, \boldsymbol{b})$.

Proof. We start by providing an upper bound on $p(\mathcal{A}_0, \boldsymbol{b})$ in terms of γ_0. Let $\boldsymbol{v}^*$ be any optimal solution to the dual optimization problem of (3.43), i.e.,

$$\boldsymbol{v}^* \in \arg\max_{\boldsymbol{v}} \langle \boldsymbol{v}, \boldsymbol{b} \rangle \quad \text{s.t. } \|\mathbf{A}_0^\top \boldsymbol{v}\|_\infty \leq 1. \tag{3.47}$$

Note that $p(\mathcal{A}_0, \boldsymbol{b}) = \langle \boldsymbol{v}^*, \boldsymbol{b} \rangle$ by strong duality. We decompose $\boldsymbol{v}^*$ into two orthogonal components $\boldsymbol{v}^* = \boldsymbol{v}^\perp + \boldsymbol{v}^\parallel$ in which $\boldsymbol{v}^\parallel \in \mathcal{S}_0$. From the fact that both $\boldsymbol{b}$ and all columns of $\mathbf{A}_0$ lie in $\mathcal{S}_0$, we have $\langle \boldsymbol{v}^*, \boldsymbol{b} \rangle = \langle \boldsymbol{v}^\parallel, \boldsymbol{b} \rangle$ and $\|\mathbf{A}_0^\top \boldsymbol{v}^\parallel\|_2 = \|\mathbf{A}_0^\top \boldsymbol{v}^*\|_2 \leq 1$. Furthermore, from the definition of the polar set in (3.1), we have $\boldsymbol{v}^\parallel \in \mathcal{K}_0^o$. Therefore, we can apply Lemma 8 and get

$$p(\mathcal{A}_0, \boldsymbol{b}) = \langle \boldsymbol{v}^\parallel, \boldsymbol{b} \rangle \leq \|\boldsymbol{b}\|_2 \cdot \|\boldsymbol{v}^\parallel\|_2 \leq \|\boldsymbol{b}\|_2 / \cos\gamma_0. \tag{3.48}$$

We now turn to providing a lower bound on $p(\mathcal{A}_-, \boldsymbol{b})$. If $\mathrm{BP}(\mathcal{A}_-, \boldsymbol{b})$ is empty, then we have $p(\mathcal{A}_-, \boldsymbol{b}) = \infty$. Otherwise, take any $\boldsymbol{c}^*$ from $\mathrm{BP}(\mathcal{A}_-, \boldsymbol{b})$, we have $\boldsymbol{b} = \mathbf{A}_- \boldsymbol{c}^*$. Left multiply by $\boldsymbol{b}^\top$ and manipulate the right hand side we have the

following:

$$\begin{aligned}\|\boldsymbol{b}\|_2^2 = \boldsymbol{b}^\top \mathbf{A}_- \boldsymbol{c}^* &\leq \|\mathbf{A}_-^\top \boldsymbol{b}\|_\infty \|\boldsymbol{c}^*\|_1 \\ &= \|\mathbf{A}_-^\top \frac{\boldsymbol{b}}{\|\boldsymbol{b}\|_2}\|_\infty \|\boldsymbol{b}\|_2 \cdot p(\mathcal{A}_-, \boldsymbol{b}) \\ &\leq \cos\theta(\mathcal{A}_-, \mathcal{S}_0) \cdot \|\boldsymbol{b}\|_2 \cdot p(\mathcal{A}_-, \boldsymbol{b}),\end{aligned} \tag{3.49}$$

where we have used the fact that $\|\boldsymbol{c}^*\|_1 = p(\mathcal{A}_-, \boldsymbol{b})$ in the second equality and that $\boldsymbol{b} \in \mathcal{S}_0$ in the second inequality. We have now established that

$$p(\mathcal{A}_-, \boldsymbol{b}) \geq \|\boldsymbol{b}\|_2 / \cos\theta(\mathcal{A}_-, \mathcal{S}_0). \tag{3.50}$$

The conclusion thus follows by combining (3.48) and (3.50) and the condition of PRC. □

In the proof of Theorem 15, the LHS and RHS of the equivalent condition $p(\mathcal{A}_0, \boldsymbol{b}) < p(\mathcal{A}_-, \boldsymbol{b})$ are bounded *separately*, without using the fact that the $\boldsymbol{b}$ on both sides of the equivalent condition are the same. In other words, this proof shows that the following result is true, which is a stronger result than that stated in Theorem 15: if PRC holds then $\forall \boldsymbol{b}, \boldsymbol{b}' \in \mathcal{S}_0 \setminus \{0\}$ with $\|\boldsymbol{b}\|_2 = \|\boldsymbol{b}'\|_2$, we have $p(\mathcal{A}_0, \boldsymbol{b}) < p(\mathcal{A}_-, \boldsymbol{b}')$. Next, we prove that DRC is a sufficient condition for the recovery condition in (14).

The DRC result. We start by showing that the condition in the rightmost box in Figure 3.3 is less restrictive than the DRC.

Lemma 5. *If the DRC $\gamma_0 < \theta(\mathcal{A}_-, \mathcal{D}_0)$ holds then it has $\|\mathbf{A}_-^\top \boldsymbol{v}\|_\infty < 1, \forall \boldsymbol{v} \in \mathcal{D}_0$.*

Proof. Take any $\boldsymbol{v} \in \mathcal{D}_0$. Since $\mathcal{D}_0$ is the set of extreme points of $\mathcal{K}_0^o$, we know that $\boldsymbol{v} \in \mathcal{K}_0^o$. Thus, we can apply Lemma 8 which gives us $\|\boldsymbol{v}\|_2 \leq 1/\cos\gamma_0$. Consequently, we have

$$\|\mathbf{A}_-^\top \boldsymbol{v}\|_\infty = \|\mathbf{A}_-^\top \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}\|_\infty \|\boldsymbol{v}\|_2 \leq \frac{\cos\theta(\mathcal{A}_-, \mathcal{D}_0)}{\cos\gamma_0} < 1. \tag{3.51}$$

□

We now prove the result for the DRC.

Theorem 16 (Correctness of BP for subspace-preserving recovery via DRC)**.** *If $\|\mathbf{A}_-^\top \boldsymbol{v}\|_\infty < 1, \forall \boldsymbol{v} \in \mathcal{D}_0$ holds then $\forall \boldsymbol{b} \in \mathcal{S}_0 \setminus \{0\}$, $p(\mathcal{A}_0, \boldsymbol{b}) < p(\mathcal{A}_-, \boldsymbol{b})$.*

Proof. Consider the following dual problem to (3.43):

$$\max_{\boldsymbol{v}} \langle \boldsymbol{v}, \boldsymbol{b} \rangle \quad \text{s.t. } \|\mathbf{A}_0^\top \boldsymbol{v}\|_\infty \leq 1. \tag{3.52}$$

We first show that there always exists a solution to (3.52) that is from the set of dual points $\mathcal{D}_0$. To do this, we consider the following optimization problem

$$\max_{w} \langle \boldsymbol{v}, \boldsymbol{b} \rangle \quad \text{s.t. } \|\mathbf{A}_0^\top \boldsymbol{v}\|_\infty \leq 1, \boldsymbol{v} \in \mathcal{S}_0. \tag{3.53}$$

Note that this program differs from (3.52) only in the feasible region. Specifi-

cally, the feasible region of (3.53) is $\mathcal{K}_0^o$, and it is bounded. Since any continuous convex function attains its maximum over a compact set at an extreme point of the convex set, there must have a solution $\boldsymbol{v}^*$ to (3.53) that is from the set of extreme points of $\mathcal{K}_0^o$, i.e, $\boldsymbol{v}^* \in \mathcal{D}_0$. Moreover, such $\boldsymbol{v}^*$ is also an optimal solution to (3.52). To see this, assume for the purpose of reaching at a contradiction that there is a solution $\bar{\boldsymbol{v}}$ to (3.52) such that $\langle \bar{\boldsymbol{v}}, \boldsymbol{b} \rangle > \langle \boldsymbol{v}^*, \boldsymbol{b} \rangle$. We decompose $\bar{\boldsymbol{v}}$ into two parts as $\bar{\boldsymbol{v}} = \bar{\boldsymbol{v}}^{\|} + \bar{\boldsymbol{v}}^{\perp}$, where $\bar{\boldsymbol{v}}^{\|}$ lies in $\mathcal{S}_0$ and $\bar{\boldsymbol{v}}^{\perp}$ is perpendicular to $\mathcal{S}_0$. Then, it is easy to check that $\bar{\boldsymbol{v}}^{\|}$ is a feasible solution to (3.53). Moreover, we have

$$\langle \bar{\boldsymbol{v}}^{\|}, \boldsymbol{b} \rangle = \langle \bar{\boldsymbol{v}}, \boldsymbol{b} \rangle > \langle \boldsymbol{v}^*, \boldsymbol{b} \rangle, \tag{3.54}$$

which contradicts the fact that $\boldsymbol{v}^*$ is an optimal solution to (3.53).

We have shown that $\boldsymbol{v}^*$ is an optimal solution to (3.52) that is in $\mathcal{D}_0$. Now, let us consider two cases. First, if $p(\mathcal{A}_-, \boldsymbol{b}) = +\infty$, then the conclusion follows since $p(\mathcal{A}_0, \boldsymbol{b})$ is always finite. Otherwise, take any $\boldsymbol{c}^* \in \mathrm{BP}(\mathcal{A}_-, \boldsymbol{b})$. We have $\|\boldsymbol{c}^*\|_1 = p(\mathcal{A}_-, \boldsymbol{b})$ and $\boldsymbol{b} = \mathbf{A}_- \boldsymbol{c}^*$. Note that (3.52) is the dual problem to (3.43), we have

$$\begin{aligned} p(\mathcal{A}_0, \boldsymbol{b}) = d(\mathcal{A}_0, \boldsymbol{b}) = \langle \boldsymbol{v}^*, \boldsymbol{b} \rangle = \langle \boldsymbol{v}^*, \mathbf{A}_- \boldsymbol{c}^* \rangle \\ \leq \|\mathbf{A}_-^\top \boldsymbol{v}^*\|_\infty \cdot \|\boldsymbol{c}^*\|_1 < p(\mathcal{A}_-, \boldsymbol{b}), \end{aligned} \tag{3.55}$$

in which we have used $\|\mathbf{A}_-^\top \boldsymbol{v}^*\|_\infty < 1$ from assumption. □

Finally, we note that the converse of the statement in Theorem 16 is not true in general. To see this, we provide the following counter-example. Take $\mathcal{A}_0$ to be any finite subset of $\mathcal{S}_0$ that spans $\mathcal{S}_0$. From Lemma 8, there exist some $\bar{\boldsymbol{v}} \in \mathcal{D}_0$ such that $\|\bar{\boldsymbol{v}}\|_2 = 1/\cos\gamma_0 > 1$ (since it is obvious that $\gamma_0 \neq 0$). Now, take $\mathcal{A}_-$ to be a singleton set containing a data point $\bar{\boldsymbol{a}} \notin \mathcal{S}_0$ that satisfies $|\langle \bar{\boldsymbol{a}}, \bar{\boldsymbol{v}} \rangle| \geq 1$. In this example, we can see that the premise of Theorem 16 does not hold, i.e., $\|\mathbf{A}_-^\top \bar{\boldsymbol{v}}\|_\infty = |\langle \bar{\boldsymbol{a}}, \bar{\boldsymbol{v}} \rangle| \geq 1$, but the conclusion of Theorem 16 is still true, i.e. $p(\mathcal{A}_0, \boldsymbol{b}) < \infty = p(\mathcal{A}_-, \boldsymbol{b})$.

3.2.3.2 Universal recovery by OMP

The lower half of Figure 3.3 summarizes the results for subspace-preserving recovery by OMP. It may be surprising that it is roughly symmetric to that of subspace-preserving recovery by BP. We first identify a sufficient condition for OMP to give subspace-preserving solutions, which is captured by the spherical angle between an arbitrary $\boldsymbol{b} \in \mathcal{S}_0$ and $\mathcal{A}_0$ and $\mathcal{A}_-$. We then provide proofs for the PRC and the DRC results.

A recovery condition. Our recovery condition is that for any point $\boldsymbol{b} \in \mathcal{S}_0 \setminus \{0\}$, the closest point (in terms of spherical distance $\theta(\cdot,\cdot)$) to either $\boldsymbol{b}$ or $-\boldsymbol{b}$ in the entire dictionary $\mathcal{A}$ must be one of those in $\mathcal{A}_0$.

Theorem 17 (A recovery condition for OMP)**.** $\mathrm{OMP}(\mathcal{A}, \boldsymbol{b})$ *is subspace-preserving for all* $\boldsymbol{b} \in \mathcal{S}_0$ *if* $\theta(\mathcal{A}_0, \{\pm\boldsymbol{b}\}) < \theta(\mathcal{A}_-, \{\pm\boldsymbol{b}\})$ *for all* $\boldsymbol{b} \in \mathcal{S}_0 \setminus \{0\}$.

Proof. Note that $\mathrm{OMP}(\mathcal{A}, \boldsymbol{b})$ is subspace-preserving if each iteration of OMP picks a point in $\mathcal{A}_0$. This can be seen in an inductive way: for any given $\boldsymbol{b} \in \mathcal{S}_0$, the first step of $\mathrm{OMP}(\mathcal{A}, \boldsymbol{b})$ chooses an atom from $\mathcal{A}_0$ by the condition that $\theta(\mathcal{A}_0, \{\pm\boldsymbol{b}\}) < \theta(\mathcal{A}_-, \{\pm\boldsymbol{b}\})$. This gives a residual $\boldsymbol{v}_0 = \boldsymbol{b} - \mathbf{A}\boldsymbol{c}_0$ with $\boldsymbol{c}_0$ defined in (2.6), and $\boldsymbol{v}_0$ is still in $\mathcal{S}_0$ since it is a linear combination of points in $\mathcal{S}_0$. If $\boldsymbol{v}_0 = \mathbf{0}$, then OMP is terminated; otherwise, by the condition we will have $\theta(\mathcal{A}_0, \{\pm\boldsymbol{v}_0\}) < \theta(\mathcal{A}_-, \{\pm\boldsymbol{v}_0\})$, which then guarantees that the next step of $\mathrm{OMP}(\mathcal{A}, \boldsymbol{b})$ also chooses an entry from $\mathcal{A}_0$. This procedure will finally be terminated when an exact recovery of $\boldsymbol{b}$ is achieved, and the solution is subspace-preserving as the support of representation coefficients is a subset of the selected entries. □

We note that this *sufficient* condition in Theorem 17 is also "almost" *necessary*, in the sense that it is necessary for guaranteeing that OMP never selects a point in $\mathcal{A}_-$ to the working set throughout the iterations. Indeed, if the sufficient condition in Theorem 17 is not satisfied for some $\boldsymbol{b} \in \mathcal{S}_0 \backslash \{\mathbf{0}\}$, i.e. if $\theta(\mathcal{A}_0, \{\pm\boldsymbol{b}\}) \geq \theta(\mathcal{A}_-, \{\pm\boldsymbol{b}\})$, then for this specific $\boldsymbol{b}$, the OMP procedure will pick a point from $\mathcal{A}_-$ in the first iteration. Meanwhile, there does exist cases in which subspace-preserving recovery is achieved even when the OMP procedure picks points not in subspace $\mathcal{S}_0$ at some iterations prior to termination: this happens when in the final iteration the coefficients to those points are set to zero (i.e., by equation (2.6)). One such example is that the span of points in

$\mathcal{A}_-$ has trivial intersection with the subspace $\mathcal{S}_0$. In this case, any solution in the set $\mathrm{OMP}(\mathcal{A}, \boldsymbol{b})$ is subspace-preserving for all $\boldsymbol{b} \in \mathcal{S}_0$ (see Theorem 29) even when the working set of OMP at termination contains points not in $\mathcal{A}_0$.

The PRC result. Similar to the discussion for BP, the term $\theta(\mathcal{A}_0, \{\pm\boldsymbol{b}\})$ on the LHS of the recovery condition in Theorem 17 depends on $\mathcal{A}_0$ and can be bounded from above by the characterization γ_0, while the term $\theta(\mathcal{A}_-, \{\pm\boldsymbol{b}\})$ depends on relation between $\mathcal{S}_0$ and $\mathcal{A}_-$ and can be bounded from below. Following this idea, we can prove the following theorem which establishes the PRC as a sufficient condition for subspace-preserving representation by OMP.

Theorem 18 (Correctness of OMP for subspace-preserving recovery via PRC)**.** *If the PRC* $\gamma_0 < \theta(\mathcal{A}_-, \mathcal{S}_0)$ *holds then* $\forall \boldsymbol{b} \in \mathcal{S}_0 \setminus \{0\}, \theta(\mathcal{A}_0, \{\pm\boldsymbol{b}\}) < \theta(\mathcal{A}_-, \{\pm\boldsymbol{b}\})$.

Proof. We prove this by bounding each side of the objective inequality separately.

For the left hand side, notice $\gamma_0 := \gamma(\pm\mathcal{A}_0)$, then by definition of covering radius, $\gamma_0 \geq \theta(\mathcal{A}_0, \{\pm\boldsymbol{b}\})$.

For the right hand side, we have $\theta(\mathcal{A}_-, \{\pm\boldsymbol{b}\}) \geq \theta(\mathcal{A}_-, \mathcal{S}_0)$ by definition of the notation $\theta(\cdot, \cdot)$.

The conclusion thus follows by concatenating the bounds for both sides above with the PRC. □

The DRC result. Finally, we prove the result for the DRC. Following the

discussion for BP, we do this by showing that the condition in the rightmost box of Figure 3.3 guarantees the sufficient condition in Theorem 17.

Theorem 19 (Correctness of OMP for subspace-preserving recovery via DRC)**.** *If* $\|\mathbf{A}_{-}^{\top}\boldsymbol{v}\|_{\infty} < 1, \forall \boldsymbol{v} \in \mathcal{D}_0$ *holds then* $\forall \boldsymbol{b} \in \mathcal{S}_0 \setminus \{0\}, \theta(\mathcal{A}_0, \{\pm\boldsymbol{b}\}) < \theta(\mathcal{A}_{-}, \{\pm\boldsymbol{b}\})$.

To prove this theorem, we use the result that the polar set $\mathcal{K}_0^o$ is a symmetric convex body and it induces a norm on the space $\mathcal{S}_0$, by means of the so-called Minkowski functional. The relevant definitions and results are stated as follows.

Definition 15 (Minkowski functional)**.** The Minkowski functional of the set $\mathcal{K}$ is defined on $\operatorname{span}(\mathcal{K})$ as

$$\|\boldsymbol{v}\|_{\mathcal{K}} = \inf\{t > 0 : \frac{\boldsymbol{v}}{t} \in \mathcal{K}\}. \tag{3.56}$$

Lemma 6. *[160] If* $\mathcal{K}$ *is a symmetric convex body, then* $\|\cdot\|_{\mathcal{K}}$ *is a norm on* $span(\mathcal{K})$ *with* $\mathcal{K}$ *being the unit ball.*

Thus, $\|\cdot\|_{\mathcal{K}_0^o}$ is a norm on $\mathcal{S}_0$ with $\mathcal{K}_0^o$ being the unit ball.

Proof of Theorem 19. It suffices to prove the result for every $\boldsymbol{b} \in \mathcal{S}_0 \setminus \{0\}$ that has a unit norm, by using any norm defined on $\mathcal{S}_0$. Here we use the norm of Minkowski functional $\|\cdot\|_{\mathcal{K}_0^o}$, and we need to prove that $\theta(\mathcal{A}_0, \{\pm\boldsymbol{b}\}) < \theta(\mathcal{A}_{-}, \{\pm\boldsymbol{b}\})$ for all $\boldsymbol{b} \in \mathcal{S}_0$ such that $\|\boldsymbol{b}\|_{\mathcal{K}_0^o} = 1$.

Recall that the set of dual points $\mathcal{D}_0$ is the set of extreme points of $\mathcal{K}_0^o$. Therefore, the convex hull of the dual points is $\mathcal{K}_0^o$ (i.e. the convex hull of the set of the extreme points of a convex body is this convex body itself, see e.g. [24]). Since $\|\boldsymbol{b}\|_{\mathcal{K}_0^o} = 1$, it has $\boldsymbol{b} \in \mathcal{K}_0^o$. Therefore, $\boldsymbol{b}$ can be expressed as a convex combination of the dual points, i.e. one can write $\boldsymbol{b} = \sum_i x_i \cdot \boldsymbol{v}_i$ in which $\boldsymbol{v}_i \in \mathcal{D}_0, x_i \in [0, 1]$ for all i and $\sum_i x_i = 1$. Thus,

$$\|\mathbf{A}_-^\top \boldsymbol{b}\|_\infty = \|\mathbf{A}_-^\top \sum_i \boldsymbol{v}_i \cdot x_i\|_\infty \leq \sum_i \|\mathbf{A}_-^\top \boldsymbol{v}_i \cdot x_i\|_\infty < \sum_i x_i = 1 = \|\mathbf{A}_0^\top \boldsymbol{b}\|_\infty, \quad (3.57)$$

in which the last equality follows from $\|\boldsymbol{b}\|_{\mathcal{K}_0^o} = 1$. The proof is completed by dividing both sides of (3.57) by $\|\boldsymbol{b}\|_2$ and then taking $\arccos$. □

Finally, we note that the converse of the statement in Theorem 19 is also true:

Theorem 20. *If* $\forall \boldsymbol{b} \in \mathcal{S}_0 \backslash \{0\}, \theta(\mathcal{A}_0, \{\pm \boldsymbol{b}\}) < \theta(\mathcal{A}_-, \{\pm \boldsymbol{b}\})$ *then* $\forall \boldsymbol{v} \in \mathcal{D}_0, \|\mathbf{A}_-^\top \boldsymbol{v}\|_\infty < 1$.

Proof. Take any $\boldsymbol{v} \in \mathcal{D}_0$. By definition of dual points, we have $\boldsymbol{v} \in \mathcal{S}_0 \backslash \{\mathbf{0}\}$, thus $\theta(\mathcal{A}_0, \{\pm \boldsymbol{v}\}) < \theta(\mathcal{A}_-, \{\pm \boldsymbol{v}\})$. This gives us

$$\begin{aligned} \|\boldsymbol{v}\|_2 \cos \theta(\mathcal{A}_0, \{\pm \boldsymbol{v}\}) &> \|\boldsymbol{v}\|_2 \cos \theta(\mathcal{A}_-, \{\pm \boldsymbol{v}\}) \\ \implies \|\mathbf{A}_0^\top \boldsymbol{v}\|_\infty &> \|\mathbf{A}_-^\top \boldsymbol{v}\|_\infty. \end{aligned} \quad (3.58)$$

Note that $\|\mathbf{A}_0^\top \boldsymbol{v}\|_\infty \leq 1$ since $\boldsymbol{v} \in \mathcal{D}_0 \subseteq \mathcal{K}_0^o$, the conclusion of the theorem follows.

□

3.3 A random analysis

In order to better understand the regime where BP and OMP succeed in recovering subspace-preserving solutions, we will consider a probabilistic data generating model and study the effect of subspace dimension d_0, the ambient space dimension D, the number of data points in the subspace N_0 and outside of the subspace N_-.

Our random data model is defined as follows.

Definition 16 (Random data model)**.** Given any quadruple (D, d_0, N_0, N_-), our random data model is defined as generating a dictionary $\mathcal{A} = \mathcal{A}_0 \cup \mathcal{A}_-$ in which $\mathcal{A}_0$ contains N_0 points drawn independently and uniformly at random on the unit sphere of a randomly generated subspace $\mathcal{S}_0 \subseteq \mathbb{R}^D$ of dimension d_0, and $\mathcal{A}_-$ contains N_- points drawn independently and uniformly at random on the unit sphere $\mathbb{S}^{D-1}$.

To state our results, we will use $c(N_0/d_0)$ (see [139]) to denote a positive numerical value which takes value $c(N_0/d_0) = \frac{1}{\sqrt{8}}$ when N_0/d_0 is greater than a certain constant.

3.3.1 Instance recovery conditions

Our first result for instance recovery under the random data model in Definition 16 is as follows.

Theorem 21 (Instance recovery condition for BP and OMP in random model)**.** *Given any quadruple* (D, d_0, N_0, N_-)*, draw a dictionary* $\mathcal{A} = \mathcal{A}_0 \cup \mathcal{A}_-$ *where* $\mathcal{A}_0 \subset \mathcal{S}_0$ *according to the random data model. Draw a vector* $\boldsymbol{b} \in \mathcal{S}_0$ *that is independent of* $\mathcal{A}$*. Assume that* $N_-/N_0 < \alpha$ *for some* $\alpha > 0$ *and* $d_0 < N_0 < d_0 e^{d_0/2}$*. Under the condition that*

$$\frac{D}{d_0} > \frac{12 \log N_0 + 4 \log \alpha}{c^2 (N_0/d_0) \log(N_0/d_0)}, \tag{3.59}$$

the probability that all elements in $\mathrm{BP}(\mathcal{A}, \boldsymbol{b})$ *are subspace-preserving is at least* $1 - \frac{2}{N_0^2} - e^{-\sqrt{N_0 \cdot d_0}}$*, and the probability that all elements in* $\mathrm{OMP}(\mathcal{A}, \boldsymbol{b})$ *are subspace-preserving is at least* $1 - \frac{2d_0}{N_0^2} - e^{-\sqrt{N_0 \cdot d_0}}$.

The proof of this result is provide in Section 3.3.3. Theorem 21 shows that both BP and OMP succeed in subspace-preserving recovery under the same conditions on the parameters D, d_0, N_0, N_-, but with different probabilities. Specifically, when $d_0 > 1$, the probability that OMP succeeds is lower than that of BP. However, it is easy to see that the difference in probability goes to zero as the number of sample points N_0 goes to infinity. This means that the performance difference vanishes as the scale of the problem increases.

One interpretation of the condition in (3.59) is that the dimension d_0 of the subspace should be small relative to the ambient space dimension D. This is well-expected since BP and OMP are expected to work better if the desired solutions are sparser, and there always exist subspace-preserving representations that have at most d_0 nonzero entries. In terms of the number of data points in the subspace (i.e., N_0), if we consider the regime where N_0/d_0 is large enough so that $c(N_0/d_0) = \frac{1}{\sqrt{8}}$, the right hand side of (3.59) is a monotone decreasing function of N_0. This suggests that the condition in (3.59) is more likely to be satisfied if the number of sample points increases. In terms of N_-, Theorem 21 is derived for N_- that grows proportionally with N_0, and the ratio $\alpha = N_-/N_0$ affects condition (3.59) only through the term $4 \log \alpha$.

A particularly interesting regime where Theorem 21 applies is where the ratio d_0/D converges to a parameter λ (i.e., D grows linearly with d_0), and the ratio $\log N_0 / \log d_0$ converges to a parameter δ (i.e., N_0 grows polynomially with d_0). More formally, we have the following asymptotic result.

Corollary 1. *Given any infinite sequence of quadruples* $\{(D^{(k)}, d_0^{(k)}, N_0^{(k)}, N_-^{(k)})\}_{k=1}^{\infty}$, *draw a sequence of dictionaries* $\mathcal{A}^{(k)} = \mathcal{A}_0^{(k)} \cup \mathcal{A}_-^{(k)}$ *where* $\mathcal{A}_0^{(k)} \subseteq \mathcal{S}_0^{(k)}$*according to the random data model. Then, draw a sequence of vectors* $\boldsymbol{b}^{(k)} \in \mathcal{S}_0^{(k)}$ *such that each* $\boldsymbol{b}^{(k)}$ *is independent of* $\mathcal{A}^{(k)}$. *Assume that* $\lim_{k\to\infty} N_0^{(k)} = \infty$ *and that there exists* $\alpha > 0$ *such that* $N_-^{(k)}/N_0^{(k)} < \alpha$ *for all sufficiently large* k. *If there exist*

$\lambda \in (0, 1/96)$ *and* $\delta \in \mathbb{R}$ *such that*

$$\lim_{k\to\infty} \left(d_0^{(k)}/D^{(k)}\right) = \lambda, \tag{3.60}$$

$$\lim_{k\to\infty} \left((\log N_0^{(k)})/(\log d_0^{(k)})\right) = \delta, \ \textit{and} \tag{3.61}$$

$$\frac{1}{1-96\lambda} < \delta, \tag{3.62}$$

then the probability that all elements in $\mathrm{BP}(\mathcal{A}^{(k)}, \boldsymbol{b}^{(k)})$ *and* $\mathrm{OMP}(\mathcal{A}^{(k)}, \boldsymbol{b}^{(k)})$ *are subspace-preserving tends to* 1 *as* $k \to \infty$.

Corollary 1 is derived directly from Theorem 21 by taking the limit on (3.59) for $k \to \infty$. The proof of this result is provide in Section 3.3.3. It clearly shows how the relative dimension λ and the relative number of samples δ together affect subspace-preserving recovery. Specifically, note that the left hand side of (3.62) is an increasing function of λ in the range $\lambda \in (0, 1/96)$. This suggests that it is easier to achieve subspace-preserving recovery when λ is smaller, or equivalently when the subspace dimension $d_0^{(k)}$ is smaller relative to the ambient dimension $D^{(k)}$. The right hand side of (3.62) increases when $N_0^{(k)}$ becomes larger relative to $d_0^{(k)}$. This suggests that subspace-preserving recovery can be improved with more number of data points.

3.3.2 Universal recovery conditions

The universal recovery condition under the random model is stated as follows.

Theorem 22 (Universal recovery condition in random model)**.** *Given any quadruple* (D, d_0, N_0, N_-)*, draw a dictionary* $\mathcal{A} = \mathcal{A}_0 \cup \mathcal{A}_-$ *where* $\mathcal{A}_0 \subseteq \mathcal{S}_0$ *according to the random data model. Assume that* $N_-/N_0 < \alpha$ *for some* $\alpha > 0$ *and* $d_0 < N_0 < d_0 e^{d_0/2}$*. Under the condition that*

$$\frac{D}{d_0} > 1 + \frac{8 \log N_0 + 4 \log \alpha + 2 d_0 \log(e \frac{D}{d_0})}{c^2 (N_0/d_0) \log(N_0/d_0)}, \tag{3.63}$$

the probability that all elements in $\mathrm{BP}(\mathcal{A}, \boldsymbol{b})$ *and* $\mathrm{OMP}(\mathcal{A}, \boldsymbol{b})$ *are subspace-preserving for all* $\boldsymbol{b} \in \mathcal{S}_0$ *is at least* $1 - \frac{1}{N_0} - e^{-\sqrt{N_0 \cdot d_0}}$.

Theorem 22 shows that both BP and OMP are guaranteed to succeed in subspace-preserving recovery under the same condition and with the same probability. For arbitrarily fixed N_0 and d_0, the condition (3.63) is satisfied if D is large enough. This is consistent with the message from the instance recovery condition in Theorem 21 that BP and OMP can work when the dimension of the subspace is low relative to the ambient dimension.

One of the major differences between the universal recovery condition in (3.63) and the instance recovery condition in (3.59) is that the condition in (3.63) has an additional term $2 d_0 \log(e \frac{D}{d_0})$. Because of this term, the universal

recovery condition is more difficult to be satisfied than that of the instance recovery, which is to be expected as universal recovery also guarantees instance recovery (but not vise versa). In addition, in the derivation of an asymptotic result for universal subspace-preserving recovery we need the parameter N_0 to be exponentially large in d_0. This is in contrast with the asymptotic result for instance subspace-preserving recovery in Corollary 1, which only requires N_0 to be polynomially large in d_0. Formally, we have the following result.

Corollary 2. *Given any infinite sequence of quadruples* $\{(D^{(k)}, d_0^{(k)}, N_0^{(k)}, N_-^{(k)})\}_{k=1}^{\infty}$*, draw a sequence of dictionaries* $\mathcal{A}^{(k)} = \mathcal{A}_0^{(k)} \cup \mathcal{A}_-^{(k)}$ *where* $\mathcal{A}_0^{(k)} \subseteq \mathcal{S}_0^{(k)}$ *according to the random data model. Assume that* $\lim_{k\to\infty} N_0^{(k)} = \infty$ *and that there exists* $\alpha > 0$ *such that* $N_-^{(k)}/N_0^{(k)} < \alpha$ *for all sufficiently large* k*. If there exist* $\lambda \in (0, 1/65)$*,* $\delta \in (0, 1/2)$ *such that*

$$\lim_{k\to\infty} \left(d_0^{(k)}/D^{(k)}\right) = \lambda, \tag{3.64}$$

$$\lim_{k\to\infty} \left((\log N_0^{(k)})/d_0^{(k)}\right) = \delta, \ \textit{and} \tag{3.65}$$

$$\frac{16 \cdot \lambda(1 - \log\lambda)}{1 - 65\lambda} < \delta, \tag{3.66}$$

then the probability that all elements in $\mathrm{BP}(\mathcal{A}^{(k)}, \boldsymbol{b})$ *and* $\mathrm{OMP}(\mathcal{A}^{(k)}, \boldsymbol{b})$ *are subspace-preserving for all* $\boldsymbol{b} \in \mathcal{S}_0^{(k)}$ *tends to* 1 *as* $k \to \infty$*.*

Corollary 2 is derived directly from Theorem 22 by taking the limit on (3.63) for $k \to \infty$. Its interpretation is similar to that of the Corollary 1. Specifically,

the left hand side of (3.66) is an increasing function of λ in the range $\lambda \in (0, 1/65)$. Therefore, it is easier to achieve subspace-preserving recovery when λ is smaller, or equivalently when the subspace dimension $d_0^{(k)}$ is smaller relative to the ambient dimension $D^{(k)}$. This benefit of a high dimensional ambient space can be understood from the geometric interpretation of the PRC and the DRC: points in $\mathcal{A}_-$ become more separated from a fixed subspace $\mathcal{S}_0$ as the ambient dimension increases [34]. Thus, the PRC and the DRC become easier to be satisfied as the ambient dimension increases.

The right hand side of (3.66) increases when $N_0^{(k)}$ becomes larger relative to $d_0^{(k)}$. In particular, (3.65) suggests that $N_0^{(k)}$ is exponentially large in the dimension $d_0^{(k)}$. When compared with the polynomial number of samples in the case of instance recovery, the exponential number of samples for universal recovery may be necessary as the latter task is more difficult than the former task. The exponential complexity may also be explained by the "curse of dimensionality", i.e., the number of points needed to cover well-enough the subspace grows exponentially with the dimension of the subspace.

By using another proof technique (see Section 3.3.3), we can derive the following result which shows that the ranges of $\lambda < 1/65$ and $\delta \leq 1/2$ to which Corollary 2 applies can be extended.

Theorem 23 (Universal recovery condition in random model - asymptotic result)**.** *Given any infinite sequence of quadruples* $\{(D^{(k)}, d_0^{(k)}, N_0^{(k)}, N_-^{(k)})\}_{k=1}^{\infty}$*, draw*

a sequence of dictionaries $\mathcal{A}^{(k)} = \mathcal{A}_0^{(k)} \cup \mathcal{A}_-^{(k)}$ *according to the random data model. Assume that* $\lim_{k\to\infty} N_0^{(k)} = \infty$ *and that there exists* $\alpha > 0$ *such that* $N_-^{(k)}/N_0^{(k)} < \alpha$ *for all sufficiently large* k*. Then, the following two results hold.*

(i) If $D = D^{(k)}$ *and* $d_0 = d_0^{(k)}$ *for all* k*, and*

$$D \geq 2d_0, \tag{3.67}$$

then the probability that all elements in $\mathrm{BP}(\mathcal{A}^{(k)}, \boldsymbol{b})$ *and* $\mathrm{OMP}(\mathcal{A}^{(k)}, \boldsymbol{b})$ *are subspace-preserving for all* $\boldsymbol{b} \in \mathcal{S}_0^{(k)}$ *tends to* 1 *as* $k \to \infty$*.*

(ii) If there exist $\lambda \in (0, 0.5), \delta \in \mathbb{R}$ *such that*

$$\lim_{k\to\infty} \left(d_0^{(k)}/D^{(k)}\right) = \lambda, \tag{3.68}$$

$$\lim_{k\to\infty} \left((\log N_0^{(k)})/d_0^{(k)}\right) = \delta, \quad \textit{and} \tag{3.69}$$

$$\frac{\log 2}{2} + \frac{\log 2}{1-2\lambda} < \delta, \tag{3.70}$$

then the probability that all elements in $\mathrm{BP}(\mathcal{A}^{(k)}, \boldsymbol{b})$ *and* $\mathrm{OMP}(\mathcal{A}^{(k)}, \boldsymbol{b})$ *are subspace-preserving for all* $\boldsymbol{b} \in \mathcal{S}_0^{(k)}$ *tends to* 1 *as* $k \to \infty$*.*

Condition (3.67) states that if the number of sample points in the subspace tends to infinity, then subspace-preserving recovery can be achieved as long as $d_0/D \leq 0.5$. Furthermore, if the parameters D and d_0 also tend to infinity, then subspace-preserving recovery can be guaranteed as long as the parameters λ

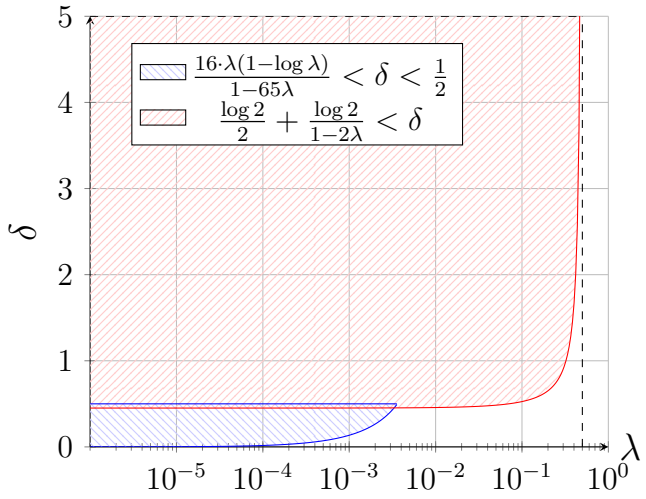


Figure 3.4: A comparison of the condition in (3.66) and the condition in (3.70).

and δ defined in (3.68) and (3.69) satisfy the condition in (3.70). In particular, the valid range of λ in (3.67) is extended from $(0, 1/65)$ as in Corollary 2 to $(0, 1/2)$. This significantly increases the range of problems where subspace-preserving recovery is applicable. On the other hand, the condition in (3.66) is tighter than the condition in (3.70) for values of λ that are close to zero, making it more suitable in that range. This can be seen from Figure 3.4 which gives a pictorial comparison of the conditions (3.66) and (3.70). In particular, as λ decreases to zero, the left hand side of (3.66) tends to zero while the left hand side of (3.70) tends to $1.5 \log 2 \approx 0.45$.

3.3.3 Proofs

In this section, we prove the results presented in the random analysis above. Our proof for instance recovery conditions in Theorem 21 and Corollary 1 are

based on deriving probabilistic bounds on the geometric conditions for instance recovery by BP (i.e., (3.29)) and by OMP (i.e., (3.33)). Our proof for universal recovery conditions in Theorem 22, Corollary 2 and Theorem 23 are based on deriving probabilistic bounds on the PRC condition in (3.38), that is, the geometric conditions for universal recovery by BP and OMP. In all the three geometric conditions (3.29), (3.33) and (3.38), the left hand side is the inradius of the set $\mathcal{K}(\pm\mathcal{A}_0)$, while the right hand side is the maximum inner product (in absolute value) between a subset of $\mathcal{S}_0$ (i.e., a point in $\mathcal{D}(\mathcal{A}_0, \boldsymbol{b})$ for the case of (3.29), all points in $\mathcal{R}(\mathcal{A}_0, \boldsymbol{b})$ for the case of (3.29) and all points in $\mathcal{S}_0$ for the case of (3.38)) and all points in $\mathcal{A}_{-}$. Therefore, we need to derive probabilistic bound on the inradius of $\mathcal{K}(\pm\mathcal{A}_0)$, the inner product between pairs of data points (for (3.29) and (3.33)) as well as inner product between data points and all points in a subspace (for (3.38)).

3.3.3.1 Volume of high-dimensional balls

We start with some background. Let $B^p(r) := \{\boldsymbol{v} \in \mathbb{R}^p : \|\boldsymbol{v}\|_2 \leq r\}$ be a ball of radius r in space $\mathbb{R}^p$. It is well known that its volume is computed in closed form, i.e.,

$$\text{vol}(B^p(r)) = v_p \cdot r^p, \;\; \text{in which} \;\; v_p = \pi^{\frac{p}{2}}/\Gamma(\frac{p}{2}+1). \tag{3.71}$$

In above, $\Gamma(\cdot)$ is the Gamma function, i.e.,

$$\Gamma(x) = \int_0^\infty e^{-t} t^{x-1} dt, x > 0. \tag{3.72}$$

We list the following properties of the Gamma function which will be used in our proof.

Lemma 7 ([68, Proposition 8.1])**.**

$$\frac{p}{\sqrt{p+1}} \leq \sqrt{2}\frac{\Gamma(\frac{p+1}{2})}{\Gamma(\frac{p}{2})} \leq \sqrt{p}. \tag{3.73}$$

Lemma 8 ([75])**.**

$$\frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)} \geq \frac{x^{x-1}y^{y-1}}{(x+y)^{x+y-1}}, \forall x, y > 0. \tag{3.74}$$

3.3.3.2 A bound on the area of spherical caps

Given any $\boldsymbol{w} \in \mathbb{S}^{p-1}$ and $\beta \in [0, \pi]$, a *spherical cap* is defined to be

$$\mathbb{S}_\beta^{p-1}(\boldsymbol{w}) := \{\boldsymbol{v} \in \mathbb{S}^{p-1}, \theta(\boldsymbol{w}, \boldsymbol{v}) \leq \beta\}, \tag{3.75}$$

that is, it is the set of points on $\mathbb{S}^{p-1}$ whose spherical distance to $\boldsymbol{w}$ is no more than β.

The following result gives upper and lower bounds for the area of a spherical cap.

Lemma 9. *For any $\beta \in [0, \pi/2]$ and any $p \geq 2$,*

$$\frac{v_{p-1}}{p v_p} \sin^{p-1} \beta \leq \frac{\sigma_{p-1}(\mathbb{S}_\beta^{p-1}(\boldsymbol{w}))}{\sigma_{p-1}(\mathbb{S}^{p-1})} \leq \frac{v_{p-1}}{v_p} \sin^{p-1} \beta, \tag{3.76}$$

in which v_p is defined in (3.71)*, and σ_{p-1} is the uniform area measure on $\mathbb{S}^{p-1}$.*

Proof. We prove by using geometry. We are motivated by the proof of a similar result in [144].

We first prove the upper bound. In Figure 3.5 we show a projection of $\mathbb{R}^p$ onto any two-dimensional space that contains the origin and $\boldsymbol{w}$. The ratio of the area of the spherical cap $\mathbb{S}_\beta^{p-1}(\boldsymbol{w})$ to the area of the entire unit sphere $\mathbb{S}^{p-1}$, is the same as the ratio of the volume of the red solid cone (i.e. $\text{cone}(\mathbb{S}_\beta^{p-1}(\boldsymbol{w}))$ where $\text{cone}(\cdot)$ is the conic hull of a set) intersecting with $B^p(1)$ to the volume of $B^p(1)$. Also note that the part of the red solid cone in the $B^p(1)$ lie completely in the green dotted cylinder, i.e., the set $\{\boldsymbol{v} \in \mathbb{R}^D : 0 \leq \langle \boldsymbol{w}, \boldsymbol{v} \rangle \leq 1, \|\mathbb{P}_{\boldsymbol{w}}^{\perp}(\boldsymbol{v})\|_2 \leq \sin\beta\}$, where $\mathbb{P}_{\boldsymbol{w}}^{\perp}(\cdot)$ is the operator of projecting the point onto the hyperplane whose normal vector is $\boldsymbol{w}$. Therefore,

$$\begin{aligned}
\frac{\sigma_{p-1}(\mathbb{S}_\beta^{p-1}(\boldsymbol{w}))}{\sigma_{p-1}(\mathbb{S}^{p-1})} &= \frac{\text{vol}(\text{cone}(\mathbb{S}_\beta^{p-1}(\boldsymbol{w})) \cap B^p(1))}{\text{vol}(B^p(1))} && (3.77)\\
&\leq \frac{\text{vol}(\{\boldsymbol{v} \in \mathbb{R}^D : 0 \leq \langle \boldsymbol{w}, \boldsymbol{v} \rangle \leq 1, \|\mathbb{P}_{\boldsymbol{w}}^{\perp}(\boldsymbol{v})\|_2 \leq \sin\beta\})}{\text{vol}(B^p(1))} && (3.78)\\
&= \frac{\sin^{p-1}\beta \cdot v_{p-1} \cdot 1}{1^p \cdot v_p} = \sin^{p-1}\beta \frac{v_{p-1}}{v_p}, && (3.79)
\end{aligned}$$

in which we have used (3.71). This gives the upper bound.

To prove the lower bound, consider again the part of the red solid cone in the $B^p(r)$. Its volume is bounded below by the intersection of the red solid cone and the cylean dashed cone. It is known that the volume of a p-dimensional cone (i.e. a cone with a $p-1$ dimensional base) is the product of the $p-1$ dimensional area of its base and its height divided by p. Therefore, the volume of the intersection of these two cones is $v_{p-1}\sin^{p-1}\beta \cdot 1/p$. This gives the lower bound in (3.76). □

There are other existing closed formulas and other lower and upper bounds for the area of spherical caps, see, e.g. [15, 93, 94]. Among them, we will also use the following well-known upper bound.

Lemma 10 ([15]). *For any $\beta \in [0, \pi/2]$ and any $p \geq 2$,*

$$\frac{\sigma_{p-1}(\mathbb{S}^{p-1}_{\beta}(\boldsymbol{w}))}{\sigma_{p-1}(\mathbb{S}^{p-1})} \leq \exp\left(-\frac{p\cos^2\beta}{2}\right). \tag{3.80}$$

3.3.3.3 A bound on the area near the subspace

Lemma 11 below provides an upper bound on the area of the region $\{\boldsymbol{w} \in \mathbb{S}^{D-1} : \theta(\boldsymbol{w}, \mathcal{S}_0) \leq \beta\}$. Geometrically, this region contains points on the unit sphere $\mathbb{S}^{D-1}$ that lie close to the subspace $\mathcal{S}_0$ (see Figure 3.2).

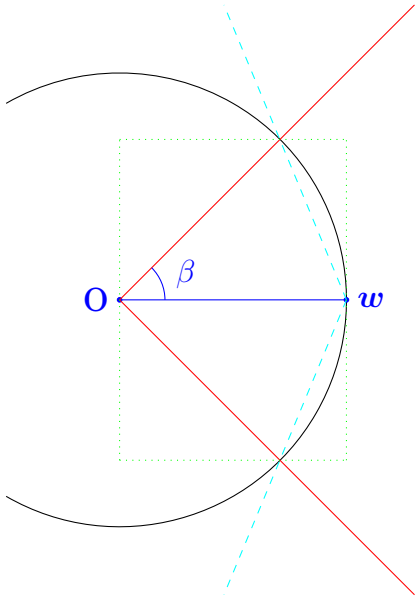


Figure 3.5: Illustration for proving bounds for area of spherical cap.

Lemma 11. *For any* $D > d_0 > 0$ *and any* $\bar{\theta} \in [0, \pi/2]$*,*

$$\frac{\sigma_{D-1}(\{\boldsymbol{w} \in \mathbb{S}^{D-1} : \theta(\boldsymbol{w}, \mathcal{S}_0) \leq \bar{\theta}\})}{\sigma_{D-1}(\mathbb{S}^{D-1})} \leq \min(\sqrt{2}^{D}, \sqrt{e\frac{D}{d_0}}^{d_0}) \sin^{D-d_0}\bar{\theta}, \tag{3.81}$$

in which σ_{D-1} *is the uniform area measure on* $\mathbb{S}^{D-1}$*.*

Proof. Our proof technique is similar to that of Lemma 9 for bounding the area of spherical caps. Consider the set

$$\mathcal{R} := \{\boldsymbol{w} \in \mathbb{R}^D : \|\mathbb{P}_{\mathcal{S}_0}(\boldsymbol{w})\|_2 \leq 1, \|\mathbb{P}^{\perp}_{\mathcal{S}_0}(\boldsymbol{w})\|_2 \leq \sin\bar{\theta}\}, \tag{3.82}$$

where $\mathbb{P}_{\mathcal{S}_0}(\cdot)$ is the operator of projecting the point onto the subspace $\mathcal{S}_0$ and

$\mathbb{P}_{\mathcal{S}_0}^{\perp}(\cdot)$ is the orthogonal complement. By geometry, it can be seen that

$$\begin{aligned}&\frac{\sigma_{D-1}(\{\boldsymbol{w}\in\mathbb{S}^{D-1}:\theta(\boldsymbol{w},\mathcal{S}_0)\leq\bar{\theta}\})}{\sigma_{D-1}(\mathbb{S}^{D-1})}\\&\quad=\frac{\mathrm{vol}(\mathrm{cone}(\{\boldsymbol{w}\in\mathbb{S}^{D-1}:\theta(\boldsymbol{w},\mathcal{S}_0)\leq\bar{\theta}\})\cap B^D(1))}{\mathrm{vol}(B^D(1))}\leq\frac{\mathrm{vol}(\mathcal{R})}{\mathrm{vol}(B^D(1))}.\end{aligned}\tag{3.83}$$

We can calculate the volume of $\mathcal{R}$ as

$$\mathrm{vol}(\mathcal{R})=(v_{d_0}\cdot 1^{d_0})\cdot(v_{D-d_0}\cdot\sin^{D-d_0}\bar{\theta}).\tag{3.84}$$

Therefore, we get

$$\frac{\mathrm{vol}(\mathcal{R})}{\mathrm{vol}(B^D(1))}=\frac{v_{d_0}v_{D-d_0}}{v_D}\cdot\sin^{D-d_0}\bar{\theta}.\tag{3.85}$$

It remains to prove that $\frac{v_{d_0}v_{D-d_0}}{v_D}\leq\min(\sqrt{2}^D,\sqrt{e\frac{D}{d_0}}^{d_0})$. By using (3.71) and (3.74) we have

$$\begin{aligned}\frac{v_{d_0}v_{D-d_0}}{v_D}&=\frac{\Gamma(\frac{D}{2}+1)}{\Gamma(\frac{d_0}{2}+1)\Gamma(\frac{D-d_0}{2}+1)}=\frac{\frac{D}{2}\cdot\Gamma(\frac{D}{2})}{\frac{d_0}{2}\frac{D-d_0}{2}\cdot\Gamma(\frac{d_0}{2})\cdot\Gamma(\frac{D-d_0}{2})}\\&\leq\frac{\left(\frac{D}{2}\right)^{\frac{D}{2}}}{\left(\frac{d_0}{2}\right)^{\frac{d_0}{2}}\left(\frac{D-d_0}{2}\right)^{\frac{D-d_0}{2}}}=\sqrt{\frac{D^D}{d_0^{d_0}(D-d_0)^{D-d_0}}}.\end{aligned}\tag{3.86}$$

We now give two upper bounds for the last term in (3.86). First, note that the term $d_0^{d_0}(D-d_0)^{D-d_0}$ in the denominator is a decreasing function of d_0 when $d_0<D/2$ and is an increasing function when $d_0>D/2$. Thus we have $d_0^{d_0}(D-$

$d_0)^{D-d_0} \geq \left(\frac{D}{2}\right)^D$, which further implies that

$$\sqrt{\frac{D^D}{d_0^{d_0}(D-d_0)^{D-d_0}}} \leq \sqrt{\frac{D^D}{\left(\frac{D}{2}\right)^D}} = \sqrt{2}^D. \tag{3.87}$$

We now derive the second bound. Let $\Lambda = D/d_0$. We have

$$\begin{aligned}
\sqrt{\frac{D^D}{d_0^{d_0}(D-d_0)^{D-d_0}}} &= \sqrt{\frac{(d_0\Lambda)^{d_0\Lambda}}{d_0^{d_0}\cdot(d_0(\Lambda-1))^{d_0(\Lambda-1)}}} \\
&= \sqrt{\frac{(d_0\Lambda)^{d_0\Lambda}}{(d_0(\Lambda-1))^{d_0\Lambda}}\cdot\frac{(d_0(\Lambda-1))^{d_0}}{d_0^{d_0}}} \\
&= \sqrt{\frac{1}{(1-\frac{1}{\Lambda})^{d_0\Lambda}}\cdot(\Lambda-1)^{d_0}} \\
&= \sqrt{\Lambda(1-\frac{1}{\Lambda})^{1-\Lambda}}^{d_0} \leq \sqrt{\Lambda e}^{d_0}.
\end{aligned} \tag{3.88}$$

The inequality in the last line follows from the fact that $(1-\frac{1}{\Lambda})^{1-\Lambda} < e$, which we prove in the rest of this proof.

We first show that $(1-\frac{1}{\Lambda})^{1-\Lambda}$ is monotonically increasing in the range $\Lambda \in (1,\infty)$. This is equivalent to showing that the function $f(\Lambda) := (1-\Lambda)\log(1-\frac{1}{\Lambda})$ is increasing. By taking the first and second order derivatives of $f(\Lambda)$ we get

$$\begin{aligned}
\frac{df}{d\Lambda} &= -\log(1-\frac{1}{\Lambda}) + (1-\Lambda)\frac{\frac{1}{\Lambda^2}}{1-\frac{1}{\Lambda}} = -\log(1-\frac{1}{\Lambda}) - \frac{1}{\Lambda}, \\
\frac{d^2f}{d\Lambda^2} &= \frac{\frac{1}{\Lambda^2}}{\frac{1}{\Lambda}-1} + \frac{1}{\Lambda^2} = \frac{1}{\Lambda(1-\Lambda)} + \frac{1}{\Lambda^2} = \frac{1}{\Lambda^2(1-\Lambda)}.
\end{aligned} \tag{3.89}$$

Note that $\frac{d^2f}{d\Lambda^2} < 0$ for all $\Lambda \in (1,\infty)$, therefore $\frac{df}{d\Lambda}$ is monotonically decreasing.

In addition, we can check that $\lim_{\Lambda_0\to\infty} \frac{df}{d\Lambda}\Big|_{\Lambda=\Lambda_0} = \lim_{\Lambda_0\to\infty} \big(-\log(1-\frac{1}{\Lambda_0}) - \frac{1}{\Lambda_0}\big) = 0$. Therefore, we have $\frac{df}{d\Lambda} > 0$ for all $\Lambda \in (1,\infty)$, which implies that $f(\Lambda)$ is monotonically increasing.

We now proceed to compute the limit of $(1-\frac{1}{\Lambda})^{1-\Lambda}$ as Λ tends to infinity. Take $x = -1/\Lambda$. We have

$$\begin{aligned} &\lim_{\Lambda\to\infty}(1-\frac{1}{\Lambda})^{1-\Lambda} = \lim_{x\to 0^-}(1+x)^{1+\frac{1}{x}} \\ &= \lim_{x\to 0^-}(1+x)\cdot\lim_{x\to 0^-}(1+x)^{\frac{1}{x}} = 1\cdot e = e. \end{aligned} \tag{3.90}$$

Combine this fact with the monotonicity of $(1-\frac{1}{\Lambda})^{1-\Lambda}$, we get that $(1-\frac{1}{\Lambda})^{1-\Lambda} < e$ for all $\Lambda \in (1,\infty)$. This finishes the proof. □

3.3.3.4 A bound on covering radius

The covering radius measures the property of dictionary atoms $\mathcal{A}_0$. Specifically, given N_0 points independently and uniformly sampled from the unit sphere $\mathbb{S}^{d_0-1}$ of the subspace $\mathcal{S}_0$, we want to measure how well-spread out they are in terms of having small covering radius. Intuitively, as N_0 increases, the unit sphere is expected to be better covered by $\mathcal{A}_0$ and the covering radius is expected to be smaller.

We start by introducing a previous result which is formulated in [139] and has been used extensively in the study of subspace clustering [153, 171, 172].

Theorem 24 ([6, 139]). *Let $\mathcal{P} \subseteq \mathbb{S}^{d_0-1}$ be a set of N_0 points that are drawn independently and uniformly at random on $\mathbb{S}^{d_0-1}$. If $N_0 < d_0 e^{d_0/2}$, then*

$$P\Big(\cos\gamma_0 < c(N_0/d_0)\sqrt{\frac{\log N_0/d_0}{2d_0}}\Big) \leq e^{-\sqrt{N_0 d_0}}. \tag{3.91}$$

This result will be used to prove our result in Theorem 21 and in Theorem 22. On the other hand, this result imposes the assumption that N_0 is bounded from above, i.e., $N_0 < d_0 e^{d_0/2}$. To prove our asymptotic results, we need a novel bound on the covering radius that allows us to send N_0 to infinity while fixing d_0. We will prove the following result.

Theorem 25. *Let $\mathcal{P} \subseteq \mathbb{S}^{d_0-1}$ be a set of N_0 points that are drawn independently and uniformly at random on $\mathbb{S}^{d_0-1}$.*

- *If $d_0 = 1$ and $N_0 > 0$, then it has $\gamma(\pm\mathcal{P}) = 0$ surely.*
- *If $d_0 \geq 2$, then for any $\bar{\gamma} \leq \pi/2$, it has $\gamma(\pm\mathcal{P}) < \bar{\gamma}$ with probability at least*

$$1 - \frac{\sqrt{2\pi d_0}\cdot 4^{d_0-1}}{\sin^{d_0-1}\bar{\gamma}} \cdot \exp\big(-\frac{2N_0}{\sqrt{2\pi d_0}}\frac{\sin^{d_0-1}\bar{\gamma}}{2^{d_0-1}}\big) \tag{3.92}$$

In Theorem 25, the result for the case $d_0 = 1$ can be see directly from the definition of covering radius. In the following, we provide a proof for the case

$d_0 \geq 2$. The idea of the proof is taken from [80]. Assume that there is a set of spherical caps of radius ϵ on $\mathbb{S}^{d_0-1}$ that can cover the entire unit sphere (i.e., an ϵ-covering as in Definition 17). If the N_0 sample points are distributed on $\mathbb{S}^{d_0-1}$ in a way that every spherical cap contains at least one sample point, then the covering radius of this set of N_0 sample points is upper-bounded by $2 \times \epsilon$. In the following, we first give an upper bound on the cardinality of an ϵ-covering of a sphere $\mathbb{S}^{d_0-1}$, then a lower bound on the probability that each spherical cap contains at least one point if N_0 points are are drawn at random on $\mathbb{S}^{d_0-1}$.

A bound on the ϵ-covering of spheres. We first formally present the concept of ϵ-covering.

Definition 17 (ϵ-covering)**.** A set $\mathcal{V} \subseteq \mathbb{S}^{d_0-1}$ is called an ϵ-covering of $\mathbb{S}^{d_0-1}$ if the covering radius of $\mathcal{V}$ is no more than ϵ.

As part of the proof to Theorem 25, we need an estimation on the cardinality of an ϵ-covering of sphere. While one can always take infinitely many points for an ϵ-covering, we want to find a small ϵ-covering. The concept of covering number captures the smallest ϵ-covering.

Definition 18 (Covering number)**.** Given $\epsilon > 0$, the covering number of $\mathbb{S}^{d_0-1}$

is defined as

$$\mathcal{C}(\mathbb{S}^{d_0-1}, \epsilon) := \min\{\text{card}(\mathcal{V}) : \mathcal{V} \text{ is an } \epsilon\text{-covering of } \mathbb{S}^{d_0-1}\}, \tag{3.93}$$

i.e., the cardinality of the smallest ϵ-covering of $\mathbb{S}^{d_0-1}$.

We have the following upper bound on the covering number.

Lemma 12. *The covering number of* $\mathbb{S}^{d_0-1}, d_0 \geq 2$ *is bounded by*

$$\mathcal{C}(\mathbb{S}^{d_0-1}, \epsilon) \leq \frac{d_0}{\frac{v_{d_0-1}}{v_{d_0}} \sin^{d_0-1}\frac{\epsilon}{2}}, \forall \epsilon \leq \frac{\pi}{2}. \tag{3.94}$$

Proof. The proof follows from the standard volume packing argument, e.g. [142, 158]. We can construct a specific ϵ-covering $\mathcal{V}$ iteratively. Initially, $\mathcal{V}$ is set to be empty. In the first step, an arbitrary point in $\mathbb{S}^{d_0-1}$ is added into $\mathcal{V}$. In the following steps, we will find any point $\boldsymbol{w}$ in $\mathbb{S}^{d_0-1}$ which satisfies $\theta(\boldsymbol{w}, \mathcal{V}) > \epsilon$ and then add this $\boldsymbol{w}$ into $\mathcal{V}$. This procedure is terminated when no such point exists.

It is easy to see that this procedure must terminate in finite number of iterations. In fact, we will provide an upper bound on the number of iterations.

Before that, we first point out that $\mathcal{V}$ constructed in this way is an ϵ-covering of $\mathbb{S}^{d_0-1}$, or equivalently, $\gamma(\mathcal{V}) \leq \epsilon$. Otherwise, there would be a $\boldsymbol{w}$ such that $\theta(\boldsymbol{w}, \mathcal{V}) > \epsilon$, and by the procedure above, this $\boldsymbol{b}$ should be added to $\mathcal{V}$. Thus,

we can bound the covering number $\mathcal{C}(\mathbb{S}^{d_0-1}, \epsilon)$ by the cardinality of $\mathcal{V}$ that we constructed above.

We now give an upper bound on $\mathrm{card}(\mathcal{V})$. Imagine that centered at each point in $\mathcal{V}$ we draw a spherical cap (i.e. a ball in the space $\mathbb{S}^{d_0-1}$ with distance metric $\theta(\cdot,\cdot)$)) with radius $\epsilon/2$. By the construction of $\mathcal{V}$, any two points in $\mathcal{V}$ are at least ϵ away, so the balls do not intersect with each other. Therefore, the sum of the area of these balls is strictly less than the area of the entire unit sphere. By using (3.76), we can bound the area of these balls, i.e., for any $\boldsymbol{w} \in \mathcal{V}$,

$$\frac{\sigma_{d_0-1}(\mathbb{S}^{d_0-1}_{\epsilon/2}(\boldsymbol{w}))}{\sigma_{d_0-1}(\mathbb{S}^{d_0-1})} \geq \frac{v_{d_0-1}}{d_0 v_{d_0}} \sin^{d_0-1}\frac{\epsilon}{2}.$$

Therefore, the cardinality of $\mathcal{V}$ is bounded by

$$\mathrm{card}(\mathcal{V}) \leq \frac{\sigma_{d_0-1}(\mathbb{S}^{d_0-1})}{\sigma_{d_0-1}(\mathbb{S}^{d_0-1}_{\epsilon/2}(\boldsymbol{w}))} \leq \frac{d_0}{\frac{v_{d_0-1}}{v_{d_0}} \sin^{d_0-1}\frac{\epsilon}{2}}.$$

As we have constructed a specific ϵ-covering, the covering number is bounded by the cardinality of $\mathcal{V}$. This finishes the proof. □

Proof of Theorem 25. Given the bound on the covering number in the previous part, we further provide a lower bound on the probability that every circle in the ϵ-covering contains at least one sample point. This will give a bound on covering radius as stated in Theorem 25.

Proof. Let $\epsilon = \bar{\gamma}/2$, and let $\mathcal{V}$ be any ϵ-covering of $\mathbb{S}^{d_0-1}$ such that $\mathrm{card}(\mathcal{V}) = \mathcal{C}(\mathbb{S}^{d_0-1}, \epsilon)$. Centered at each point of $\mathcal{V}$ we draw a spherical ball of radius ϵ, then the union of these balls covers the entire sphere. The idea of the proof is that if each of the balls contain at least one point from the set $\pm\mathcal{P}$, then the covering radius $\gamma(\pm\mathcal{P})$ is bounded by 2ϵ. This is because that for any $\boldsymbol{w} \in \mathbb{S}^{d_0-1}$, it lies in at least one of the balls, and when this ball contains at least one point in $\pm\mathcal{P}$, then the distance $\theta(\boldsymbol{w}, \pm\mathcal{P})$ is bounded above by 2ϵ. Concretely, denote $M := \mathrm{card}(\mathcal{V})$ and let $B_0, \cdots, B_M$ be the balls illustrated above, then

$$\begin{aligned} P(\gamma(\pm\mathcal{P}) > 2\epsilon) &\leq P(\exists i \in \{1, \cdots, M\} \text{ s.t. } B_i \cap \pm\mathcal{P} = \emptyset) \\ &\leq \sum_{i=1}^{M} P(B_i \cap \pm\mathcal{P} = \emptyset) \\ &= \sum_{i=1}^{M} (1 - 2\frac{\sigma_{d_0-1}(B_i)}{\sigma_{d_0-1}(\mathbb{S}^{d_0-1})})^{N_0}, \end{aligned}$$

where the factor of 2 appears in the last line because we are considering covering radius of symmetrized points $\pm\mathcal{P}$. Notice that each B_i is a spherical cap of radius ϵ, we can apply (3.76) and get

$$P(\gamma(\pm\mathcal{P}) > 2\epsilon) \leq \sum_{i=1}^{M} (1 - \frac{2v_{d_0-1}}{d_0 v_{d_0}} \sin^{d_0-1}\epsilon)^{N_0} \leq M \exp(-N_0 \frac{2v_{d_0-1}}{d_0 v_{d_0}} \sin^{d_0-1}\epsilon). \quad (3.95)$$

We can further bound M by (3.94), so

$$P(\gamma(\pm\mathcal{P}) > 2\epsilon) \leq \frac{d_0}{\frac{v_{d_0-1}}{v_{d_0}} \sin^{d_0-1}\frac{\epsilon}{2}} \exp(-N_0 \frac{2v_{d_0-1}}{d_0 v_{d_0}} \sin^{d_0-1}\epsilon). \tag{3.96}$$

By using the result (3.73), we get

$$P(\gamma(\pm\mathcal{P}) > 2\epsilon) \leq \frac{\sqrt{2\pi d_0}}{\sin^{d_0-1}\frac{\epsilon}{2}} \exp(-\frac{2N_0}{\sqrt{2\pi d_0}} \sin^{d_0-1}\epsilon). \tag{3.97}$$

By replacing ϵ with $\bar{\gamma}/2$ and using the fact that $\sin(x) \leq 2\sin(x/2)$ for any $x \in [0, \pi]$ we get (3.92). □

3.3.3.5 Proof of Theorem 21 and Corollary 1

The proof follows by providing probabilistic bounds on each side of the deterministic conditions for BP in (3.29) and for OMP in (3.33).

We start with the proof for BP. Let $\boldsymbol{v}$ be any point in $\mathcal{D}(\mathcal{A}_0, \boldsymbol{b})$. From Theorem 10, BP achieves subspace-preserving recovery if

$$\cos\gamma_0 > |\langle \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}, \boldsymbol{a}\rangle|, \forall \boldsymbol{a} \in \mathcal{A}_{-}. \tag{3.98}$$

A lower bound on the left hand side of (3.98) is given by Theorem 24, i.e., we have

$$P\Big(\cos\gamma_0 \geq c(N_0/d_0)\sqrt{\frac{\log N_0/d_0}{2d_0}}\Big) \geq 1 - e^{-\sqrt{N_0 d_0}}, \tag{3.99}$$

To obtain an upper bound on the right hand side of (3.98), note that the vector $\boldsymbol{v}$ is independent with any vector $\boldsymbol{a} \in \mathcal{A}_{-}$, and that any vector $\boldsymbol{a} \in \mathcal{A}_{-}$ is uniformly distributed on the unit sphere of the ambient space. Therefore, the distribution of the inner product $\langle \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}, \boldsymbol{a} \rangle$ is as if one is fixed, and the other is uniformly distributed on $\mathbb{S}^{D-1}$. We can then obtain an upper bound on the inner product $\langle \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}, \boldsymbol{a} \rangle$ from the upper bound on the area of spherical cap in Lemma 10:

$$P(|\langle \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}, \boldsymbol{a} \rangle| \leq \cos\beta) \geq 1 - 2 \cdot \exp\left(-\frac{D\cos^2\beta}{2}\right). \tag{3.100}$$

By taking $\cos\beta = \sqrt{\frac{6\log N_0 + 2\log\alpha}{D}}$ in (3.100), we get

$$P\big(|\langle \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}, \boldsymbol{a} \rangle| \leq \sqrt{\frac{6\log N_0 + 2\log\alpha}{D}}\big) \geq 1 - \frac{2}{\alpha N_0^3}. \tag{3.101}$$

By applying a union bound to at most αN_0 points in $\mathcal{A}_{-}$, we get

$$P\big(|\langle \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}, \boldsymbol{a} \rangle| \leq \sqrt{\frac{6\log N_0 + 2\log\alpha}{D}} \ \text{ for all } \ \boldsymbol{a} \in \mathcal{A}_{-}\big) \geq 1 - \frac{2}{N_0^2}. \tag{3.102}$$

The statement in Theorem 21 for BP follows by applying union bound on (3.99) and (3.102).

We now provide the proof for OMP. From Theorem 11, OMP achieves subspace-

preserving recovery if

$$\cos\gamma_0 > |\langle \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}, \boldsymbol{a}\rangle|, \forall \boldsymbol{a} \in \mathcal{A}_{-}, \tag{3.103}$$

for all $\boldsymbol{v} \in \mathcal{R}(\mathcal{A}_0, \boldsymbol{b})$. From Definition 13, the set $\mathcal{R}(\mathcal{A}_0, \boldsymbol{b})$ contains all the residual points obtained in the procedure of OMP applied to $\mathcal{A}_0$ and $\boldsymbol{b}$. Since $\boldsymbol{b} \in \mathcal{S}_0 = \text{span}(\mathcal{A}_0)$, the residual vector in step 4 (see Algorithm 1) will be a zero vector after d_0 iterations (i.e., when the working set $\mathcal{W}$ contains d_0 data points). Therefore, the OMP procedure will terminate after at most d_0 iterations. This implies that the set $\mathcal{R}(\mathcal{A}_0, \boldsymbol{b})$ contains at most d_0 number of data points. From this fact, we have

$$P(|\langle \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}, \boldsymbol{a}\rangle| < \sqrt{\frac{6\log N_0 + 2\log\alpha}{D}} \text{ for all } \boldsymbol{v} \in \mathcal{R}(\mathcal{A}_0, \boldsymbol{b})) \geq 1 - \frac{2d_0}{\alpha N_0^3}. \tag{3.104}$$

By applying a union bound to at most αN_0 points in $\mathcal{A}_{-}$, we get

$$P\big(|\langle \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}, \boldsymbol{a}\rangle| \leq \sqrt{\frac{6\log N_0 + 2\log\alpha}{D}} \text{ for all } \boldsymbol{a} \in \mathcal{A}_{-} \text{ and } \boldsymbol{v} \in \mathcal{R}(\mathcal{A}_0, \boldsymbol{b})\big) \geq 1 - \frac{2d_0}{N_0^2}. \tag{3.105}$$

The statement in Theorem 21 for OMP follows by applying union bound on (3.99) and (3.105).

We now proceed to the proof of Corollary 1. The proof is based on showing that for all k large enough, the quadruple $(D^{(k)}, d_0^{(k)}, N_0^{(k)}, N_{-}^{(k)})$ satisfies the

conditions in Theorem 21 under the assumptions (3.60), (3.61) and (3.62).

We first show that $d^{(k)} < N_0^{(k)} < d_0^{(k)} e^{d_0^{(k)}/2}$ for all k large enough. Since $\lim_{k\to\infty} \frac{\log N_0^{(k)}}{\log d_0^{(k)}} = \delta$ (which follows from (3.61)) and $\delta > 1$ (which follows from (3.62)), we have $\log d^{(k)} < \log N_0^{(k)}$ for all k large enough, which further implies that $d^{(k)} < N_0^{(k)}$ for all k large enough. Moreover, we have

$$\begin{aligned}\lim_{k\to\infty} \frac{\log N_0^{(k)}}{\log(d_0^{(k)} e^{d_0^{(k)}/2})} &= \lim_{k\to\infty} \frac{\log N_0^{(k)}}{(\log d_0^{(k)}) + d_0^{(k)}/2} \\ &= \lim_{k\to\infty} \frac{\log N_0^{(k)}}{\log d_0^{(k)}} \cdot \lim_{k\to\infty} \frac{\log d_0^{(k)}}{(\log d_0^{(k)}) + d_0^{(k)}/2} = \delta \cdot 0 = 0, \quad (3.106)\end{aligned}$$

which implies that $N_0^{(k)} < d_0^{(k)} e^{d_0^{(k)}/2}$ for all k large enough.

We now show that the condition in (3.59) is satisfied for all k large enough. This can be seen by taking the limit on the right hand side of (3.59), which gives us

$$\begin{aligned}\lim_{k\to\infty} \frac{12\log N_0^{(k)} + 4\log\alpha}{c^2(N_0^{(k)}/d_0^{(k)})\log(N_0^{(k)}/d_0^{(k)})} &= \frac{\lim_{k\to\infty}\left(12\frac{\log N_0^{(k)}}{\log d_0^{(k)}} + 4\frac{\log\alpha}{\log d_0^{(k)}}\right)}{\lim_{k\to\infty}\left(c^2(N_0^{(k)}/d_0^{(k)}) \cdot \frac{\log N_0^{(k)} - \log d_0^{(k)}}{\log d_0^{(k)}}\right)} \\ &= \frac{12\delta + 0}{\frac{1}{8}\cdot(\delta - 1)} = \frac{96}{1 - \frac{1}{\delta}} < \frac{1}{\lambda} = \lim_{k\to\infty} \frac{D^{(k)}}{d^{(k)}}, \quad (3.107)\end{aligned}$$

where we have used the fact that $c^2(N_0^{(k)}/d_0^{(k)}) = \frac{1}{8}$ if $N_0^{(k)}/d_0^{(k)}$ is larger than a certain threshold. Therefore, we have $\frac{12\log N_0^{(k)} + 4\log\alpha}{c^2(N_0^{(k)}/d_0^{(k)})\log(N_0^{(k)}/d_0^{(k)})} < \frac{D^{(k)}}{d^{(k)}}$ for all k large enough.

Finally, it is easy to see that the probability of success in Theorem 21 goes to 1 as $k \to \infty$. This establishes our claim that subspace-preserving recovery is achieved with the probability that tends to 1.

3.3.3.6 Proof of Theorem 22 and Corollary 2

We prove this theorem by deriving probabilistic bounds on each side of the PRC condition in Theorem 12, which can be rewritten as follows:

$$\cos \gamma_0 > |\langle \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}, \boldsymbol{a} \rangle|, \quad \forall \boldsymbol{a} \in \mathcal{A}_-, \forall \boldsymbol{v} \in \mathcal{S}_0 - \{\boldsymbol{0}\}. \tag{3.108}$$

Note that on the right hand side of (3.108), each vector $\boldsymbol{a} \in \mathcal{A}_-$ is uniformly distributed on the unit sphere $\mathbb{S}^{D-1}$. Therefore, we can apply Lemma 11 and get

$$\begin{aligned} P(|\langle \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}, \boldsymbol{a} \rangle| \leq \cos \beta \ \text{ for all } \ \boldsymbol{v} \in \mathcal{S}_0 - \{\boldsymbol{0}\}) &\geq 1 - \sqrt{e \frac{D}{d_0}}^{d_0} \sin^{D-d_0} \beta \\ &\geq 1 - \sqrt{e \frac{D}{d_0}}^{d_0} \exp(-\frac{D-d_0}{2} \cos^2 \beta). \end{aligned} \tag{3.109}$$

Applying a union bound to all $\boldsymbol{a} \in \mathcal{A}_-$, we get

$$\begin{aligned} P(|\langle \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}, \boldsymbol{a} \rangle| \leq \cos \beta \ \text{ for all } \ \boldsymbol{v} \in \mathcal{S}_0 - \{\boldsymbol{0}\} \ \text{ and for all } \ \boldsymbol{a} \in \mathcal{A}_-) \\ \geq 1 - N_- \cdot \sqrt{e \frac{D}{d_0}}^{d_0} \exp(-\frac{D-d_0}{2} \cos^2 \beta). \end{aligned} \tag{3.110}$$

By taking $\cos\beta = \min\left(1, \sqrt{\frac{4\log N_0 + 2\log\alpha + d_0\log(e\frac{D}{d_0})}{D-d_0}}\right)$, we further get

$$P(|\langle \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}, \boldsymbol{a}\rangle| \leq \sqrt{\frac{4\log N_0 + 2\log\alpha + d_0\log(e\frac{D}{d_0})}{D-d_0}} \text{ for all } \boldsymbol{v} \in \mathcal{S}_0 - \{\boldsymbol{0}\} \text{ and all } \boldsymbol{a} \in \mathcal{A}_-) \geq 1 - \frac{1}{N_0}. \quad (3.111)$$

Applying union bound to (3.111) and (3.99) we get

$$P(\cos\gamma_0 > |\langle \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}, \boldsymbol{a}\rangle| \text{ for all } \boldsymbol{v} \in \mathcal{S}_0 - \{\boldsymbol{0}\} \text{ and all } \boldsymbol{a} \in \mathcal{A}_-) \geq 1 - \exp(-\sqrt{N_0 d_0}) - \frac{1}{N_0}, \quad (3.112)$$

provided that the following condition holds:

$$\sqrt{\frac{4\log N_0 + 2\log\alpha + d_0\log(e\frac{D}{d_0})}{D-d_0}} < c(N_0/d_0)\sqrt{\frac{\log N_0/d_0}{2d_0}}. \quad (3.113)$$

By taking squares on both sides of (3.113) and rearranging the terms, we can get the condition in (3.63).

We proceed to present the proof of Corollary 2. The proof is based on showing that the condition (3.63) in Theorem 22 is satisfied for all k large enough

under the conditions (3.64), (3.65) and (3.66). Concretely, we have

$$\begin{aligned}
&\lim_{k\to\infty} 1 + \frac{8\log N_0^{(k)} + 4\log\alpha + 2d_0^{(k)}\log(e\frac{D^{(k)}}{d_0^{(k)}})}{c^2(N_0^{(k)}/d_0^{(k)})\log(N_0^{(k)}/d_0^{(k)})} \\
&= 1 + \frac{\lim_{k\to\infty}\left(\frac{8\log N_0^{(k)}}{d_0^{(k)}} + \frac{4\log\alpha}{d_0^{(k)}} + 2\log(e\frac{D^{(k)}}{d_0^{(k)}})\right)}{\lim_{k\to\infty}\left(c^2(N_0^{(k)}/d_0^{(k)})\cdot\frac{\log(N_0^{(k)})-\log(d_0^{(k)})}{d_0^{(k)}}\right)} \\
&= 1 + \frac{8\delta + 0 + 2(1-\log\lambda)}{\frac{1}{8}\cdot(\delta - 0)} \\
&= 65 + 16\cdot\frac{1-\log\lambda}{\delta} < 65 + 16\cdot\frac{1-\log\lambda}{16\cdot\frac{\lambda(1-\log\lambda)}{1-65\lambda}} \\
&= \frac{1}{\lambda} = \lim_{k\to\infty}\frac{d_0^{(k)}}{D_0^{(k)}},
\end{aligned} \tag{3.114}$$

where in going from line 2 to line 3 we have used the assumptions that $\log N_0^{(k)}/d_0^{(k)} \to \delta$, $d_0^{(k)}/D^{(k)} \to \lambda$, and the fact that $c^2(N_0^{(k)}/d_0^{(k)}) = \frac{1}{8}$ when $N_0^{(k)}/d_0^{(k)}$ is large enough (as stated in [139]); in line 4 we have used the assumption in (3.66). This shows that the condition (3.63) in Theorem 22 is satisfied for all k large enough. Meanwhile, it is easy to see that the probability of success in Theorem 22 goes to 1 as $k \to \infty$. This establishes our claim that subspace-preserving recovery is achieved with the probability that tends to 1.

We now proceed to the proof of Corollary 2. The proof is based on showing that for all k large enough, the quadruple $(D^{(k)}, d_0^{(k)}, N_0^{(k)}, N_-^{(k)})$ satisfies the conditions in Theorem 22 under the assumptions (3.64), (3.65) and (3.66).

We first show that $d^{(k)} < N_0^{(k)} < d_0^{(k)} e^{d_0^{(k)}/2}$ for all k large enough. Note that

$$\lim_{k\to\infty} \frac{N_0^{(k)}}{d_0^{(k)}} = \lim_{k\to\infty} \frac{N_0^{(k)}}{\log N_0^{(k)}} \cdot \lim_{k\to\infty} \frac{\log N_0^{(k)}}{d_0^{(k)}} = \infty \cdot \delta = \infty, \tag{3.115}$$

where we have used the fact that $\lim_{k\to\infty} \frac{N_0^{(k)}}{(d_0^{(k)})^\delta} = 1$ which follows from (3.61). This implies that $d^{(k)} < N_0^{(k)}$ for all k large enough. Moreover, we ahve

$$\begin{aligned} \lim_{k\to\infty} \frac{\log N_0^{(k)}}{\log(d_0^{(k)} e^{d_0^{(k)}/2})} &= \lim_{k\to\infty} \frac{\log N_0^{(k)}}{(\log d_0^{(k)}) + d_0^{(k)}/2} \\ &= \lim_{k\to\infty} \frac{\log N_0^{(k)}}{d_0^{(k)}} \cdot \lim_{k\to\infty} \frac{d_0^{(k)}}{\log(d_0^{(k)} + d_0^{(k)}/2)} = \delta \cdot 2 < 1, \end{aligned} \tag{3.116}$$

where we have used the assumption that $\delta < 1/2$. This implies that $N_0^{(k)} < d_0^{(k)} e^{d_0^{(k)}/2}$ for all k large enough.

We now show that the condition in (3.63) is satisfied for all k large enough. This can be seen by taking the limit on the right hand side of (3.63), which

gives us

$$\begin{aligned}
&\lim_{k\to\infty} 1 + \frac{8\log N_0^{(k)} + 4\log\alpha + 2d_0^{(k)}\log(e\frac{D^{(k)}}{d_0^{(k)}})}{c^2(N_0^{(k)}/d_0^{(k)})\log(N_0^{(k)}/d_0^{(k)})} \\
&= 1 + \frac{\lim_{k\to\infty}\left(\frac{8\log N_0^{(k)}}{d_0^{(k)}} + \frac{4\log\alpha}{d_0^{(k)}} + 2\log(e\frac{D^{(k)}}{d_0^{(k)}})\right)}{\lim_{k\to\infty}\left(c^2(N_0^{(k)}/d_0^{(k)})\cdot\frac{\log(N_0^{(k)})-\log(d_0^{(k)})}{d_0^{(k)}}\right)} \\
&= 1 + \frac{8\delta + 0 + 2(1-\log\lambda)}{\frac{1}{8}\cdot(\delta - 0)} \\
&= 65 + 16\cdot\frac{1-\log\lambda}{\delta} < 65 + 16\cdot\frac{1-\log\lambda}{16\cdot\frac{\lambda(1-\log\lambda)}{1-65\lambda}} \\
&= \frac{1}{\lambda} = \lim_{k\to\infty}\frac{d_0^{(k)}}{D_0^{(k)}},
\end{aligned} \tag{3.117}$$

where we have used the fact that $c^2(N_0^{(k)}/d_0^{(k)}) = \frac{1}{8}$ when $N_0^{(k)}/d_0^{(k)}$ is large enough. This shows that the condition (3.63) in Theorem 22 is satisfied for all k large enough. Finally, it is easy to see that the probability of success in Theorem 22 goes to 1 as $k \to \infty$. This finishes the proof.

3.3.3.7 Proof of Theorem 23

The theorem is derived from providing probabilistic bounds on each side of the PRC condition in Theorem 12, which could be rewritten as

$$\gamma(\pm\mathcal{A}_0) < \theta(\mathcal{S}_0, \mathcal{A}_-). \tag{3.118}$$

We will use the upper bound on the left hand side of (3.118) derived in Theorem 25. We can also derive the following lower bound on the right hand side of (3.118) from Lemma 11:

$$P(\theta(\mathcal{A}_-, \mathcal{S}_0) > \bar{\theta}) \geq 1 - N_- \min(\sqrt{2}^D, \sqrt{e\frac{D}{d_0}}^{d_0}) \cdot \sin^{D-d_0} \bar{\theta}. \tag{3.119}$$

We first prove the result stated in Theorem 23 (i) where D, d_0 are fixed and both $N_0^{(k)}, N_-^{(k)} \to \infty$. Let $\gamma_0^{(k)} := \gamma(\pm\mathcal{A}_0^{(k)})$.

Consider the case where $d_0 = 1$. From Theorem 25, we have $P(\gamma_0^{(k)} = 0) = 1$ as long as $N_0^{(k)} > 0$. Moreover, if condition (3.67) is satisfied, i.e., if $D \geq 2d_0$ then it has $P(\theta(\mathcal{A}_-^{(k)}, \mathcal{S}_0^{(k)}) > 0) = 1$ from (3.119). Therefore, we have $P(\gamma_0^{(k)} < \theta(\mathcal{A}_-^{(k)}, \mathcal{S}_0^{(k)})) = 1$ for all k large enough (i.e., for all k such that $N_0^{(k)} > 0$). We therefore conclude that $\lim_{k\to\infty} P(\gamma_0^{(k)} < \theta(\mathcal{A}_-^{(k)}, \mathcal{S}_0^{(k)})) \to 1$.

We now consider the general case where $d_0 > 1$. For each k, we set both $\bar{\gamma}$ in (3.92) and $\bar{\theta}$ in (3.119) to be such that

$$\sin \bar{\gamma}^{(k)} = \sin \bar{\theta}^{(k)} = {N_0^{(k)}}^{-\frac{D-d_0+1}{d_0(D-d_0)}}. \tag{3.120}$$

Plugging it into (3.92) and (3.119) we get

$$P(\gamma_0^{(k)} < \bar{\gamma}^{(k)}) \geq 1 - \sqrt{2\pi d_0} \cdot 4^{d_0 - 1} {N_0^{(k)}}^{\frac{(D-d_0+1)(d_0-1)}{d_0(D-d_0)}} \cdot \exp(-\frac{2}{\sqrt{2\pi d_0} 2^{d_0-1}} {N_0^{(k)}}^{\frac{D-2d_0+1}{d(D-d_0)}}), \quad (3.121)$$

$$P(\theta(\mathcal{A}_-^{(k)}, \mathcal{S}_0^{(k)}) > \bar{\theta}^{(k)}) \geq 1 - \alpha \cdot \sqrt{2}^D \cdot {N_0^{(k)}}^{-\frac{D-2d_0+1}{d_0}}, \quad (3.122)$$

in which we have used the assumption that $N_-^{(k)} < \alpha N_0^{(k)}$ for all k large enough. Note that when condition (3.67) is satisfied, i.e., when $D \geq 2d_0$, the probabilities of $\gamma_0^{(k)} < \bar{\gamma}^{(k)}$ and $\theta(\mathcal{A}_-^{(k)}, \mathcal{S}_0^{(k)}) > \bar{\theta}^{(k)}$ all converge to one as $N_0^{(k)} \to \infty$. By applying a union bound, we get

$$P(\gamma_0^{(k)} < \theta(\mathcal{A}_-^{(k)}, \mathcal{S}_0^{(k)})) \to 1 \text{ as } k \to \infty. \quad (3.123)$$

Now we turn to proving Theorem 23 (ii) in which all elements in the quadruple $(D^{(k)}, d_0^{(k)}, N_0^{(k)}, N_-^{(k)})$ goes to infinity as $k \to \infty$. Set $\delta' = \frac{\delta}{\log 2}$, and let $\epsilon = \frac{1}{2}(\delta' - \frac{1}{2} - \frac{1}{1-2\lambda})(1-2\lambda)$. Note that ϵ is positive when the condition in (3.70) is satisfied. Then we set

$$\sin \bar{\theta} = 2^{-\frac{\frac{1}{2}+(\delta'+\epsilon)\lambda}{1-\lambda}}, \sin \bar{\gamma} = 2^{1-\delta'+\epsilon}. \quad (3.124)$$

Notice that

$$\begin{aligned}
&\sin\bar{\theta} \geq \sin\bar{\gamma} \\
\Leftrightarrow\ & -\frac{\frac{1}{2}+(\delta'+\epsilon)\lambda}{1-\lambda} \geq 1-\delta'+\epsilon \\
\Leftrightarrow\ & -\frac{1}{2}-(\delta'+\epsilon)\lambda \geq (1-\delta'+\epsilon)(1-\lambda) \\
\Leftrightarrow\ & (1-2\lambda)\delta' \geq \frac{3}{2}+\epsilon-\lambda \\
\Leftrightarrow\ & \delta' \geq \frac{\frac{3}{2}+\epsilon-\lambda}{1-2\lambda} \\
\Leftrightarrow\ & \delta'-\frac{1}{2}-\frac{1}{1-2\lambda} \geq \frac{\epsilon}{1-2\lambda} \\
\Leftrightarrow\ & 2\epsilon \geq \epsilon,
\end{aligned} \tag{3.125}$$

in which we have plugged in the definition of ϵ in the last step. This shows that $\bar{\theta} \geq \bar{\gamma}$. By substituting $\sin\bar{\theta}$ and $\sin\bar{\gamma}$ into (3.119) and (3.92), respectively, we get

$$P(\gamma_0^{(k)} < \bar{\gamma}) \geq 1-\sqrt{2\pi d_0^{(k)}}\cdot 2^{(d_0^{(k)}-1)(1+\delta'-\epsilon)}\cdot\exp\big(-\frac{2N_0^{(k)}}{\sqrt{2\pi d_0^{(k)}}}2^{(\epsilon-\delta')(d_0^{(k)}-1)}\big), \tag{3.126}$$

and

$$\begin{aligned}
&P(\theta(\mathcal{A}_-^{(k)},\mathcal{S}_0^{(k)}) > \bar{\theta}) \geq\ 1-\alpha N_0^{(k)}\cdot 2^{\frac{D^{(k)}}{2}-\frac{\frac{1}{2}+(\delta'+\epsilon)\lambda}{1-\lambda}(D^{(k)}-d_0^{(k)})} \\
=\ & 1-\alpha N_0^{(k)}\cdot 2^{\left(-\frac{\lambda}{1-\lambda}(\frac{1}{2}+\delta'+\epsilon)D^{(k)}+\frac{\frac{1}{2}+(\delta'+\epsilon)\lambda}{1-\lambda}d_0^{(k)}\right)}.
\end{aligned} \tag{3.127}$$

Now, from the assumption that $\frac{\log N_0^{(k)}}{d_0^{(k)}} \to \delta$, we have $\frac{\log_2 N_0^{(k)}}{d_0^{(k)}} \to \delta'$. Therefore,

$N_0^{(k)} > 2^{d_0^{(k)}(\delta'-\epsilon/2)}$ for all k large enough. Plugging this bound on $N_0^{(k)}$ into (3.126) we get

$$P(\gamma_0^{(k)} < \bar{\gamma}) \geq 1 - \sqrt{2\pi d_0^{(k)}} \cdot 2^{(d_0^{(k)}-1)(1+\delta'-\epsilon)} \cdot \exp\big(-\frac{2 \cdot 2^{d_0^{(k)}\epsilon/2}}{\sqrt{2\pi d_0^{(k)}}} 2^{\delta'-\epsilon}\big). \tag{3.128}$$

By taking $k \to \infty$, we have

$$\lim_{k\to\infty} P(\gamma_0^{(k)} < \bar{\gamma}) = 1. \tag{3.129}$$

Furthermore, take any $\epsilon_\lambda \in \big(0, \frac{\lambda\epsilon(1-\lambda)}{\frac{1}{2}+\epsilon\lambda}\big)$ and any $\epsilon_\delta \in \big(0, \frac{\epsilon_\lambda}{(1-\lambda)(\lambda+\epsilon_\lambda)}\delta'\big)$. Since $\frac{\log_2 N_0^{(k)}}{d_0^{(k)}} \to \delta'$ and $\frac{d_0^{(k)}}{D^{(k)}} \to \lambda$, we have $D^{(k)} > \frac{d_0^{(k)}}{\lambda+\epsilon_\lambda}$ and $N_0^{(k)} < 2^{d_0^{(k)}(\delta'+\epsilon_\delta)}$ for k large enough. Plugging these bounds on $D^{(k)}$ and $N_0^{(k)}$ into (3.127) we can get

$$\begin{aligned} & P(\theta(\mathcal{A}_-^{(k)}, \mathcal{S}_0^{(k)}) > \bar{\theta}) \\ \geq & 1 - \alpha \cdot 2^{d_0^{(k)}(\delta'+\epsilon_\delta)} \cdot 2^{\big(-\frac{\lambda}{1-\lambda}(\frac{1}{2}+\delta'+\epsilon)\frac{d_0^{(k)}}{\lambda+\epsilon_\lambda} + \frac{\frac{1}{2}+(\delta'+\epsilon)\lambda}{1-\lambda} d_0^{(k)}\big)} \\ = & 1 - \alpha \cdot 2^{d_0^{(k)} \cdot C(\delta', \epsilon_\delta, \lambda, \epsilon_\lambda, \epsilon)}, \end{aligned} \tag{3.130}$$

where

$$\begin{aligned}
&C(\delta', \epsilon_\delta, \lambda, \epsilon_\lambda, \epsilon) \\
=&\delta' + \epsilon_\delta - \frac{\lambda}{1-\lambda}\frac{\frac{1}{2}+\delta'+\epsilon}{\lambda+\epsilon_\lambda} + \frac{\frac{1}{2}+(\delta'+\epsilon)\lambda}{1-\lambda} \\
=&\big(\delta' + \epsilon_\delta - \frac{\lambda}{1-\lambda}\frac{1}{\lambda+\epsilon_\lambda}\delta' + \frac{\lambda}{1-\lambda}\delta'\big) \\
&\qquad + \big(-\frac{\lambda}{1-\lambda}\frac{\epsilon+\frac{1}{2}}{\lambda+\epsilon_\lambda} + \frac{\frac{1}{2}+\epsilon\lambda}{1-\lambda}\big) \\
=&\big(\epsilon_\delta - \delta'\frac{\epsilon_\lambda}{(1-\lambda)(\lambda+\epsilon_\lambda)}\big) \\
&\quad + \big(\frac{\frac{1}{2}+\epsilon\lambda}{(1-\lambda)(\lambda+\epsilon_\lambda)}(\epsilon_\lambda - \frac{\epsilon\lambda(1-\lambda)}{\frac{1}{2}+\epsilon\lambda})\big) < 0.
\end{aligned} \tag{3.131}$$

Therefore, we can see that

$$\lim_{k\to\infty} P(\theta(\mathcal{A}_-^{(k)}, \mathcal{S}_0^{(k)}) > \bar{\theta}) = 1. \tag{3.132}$$

By applying union bound on (3.132) and (3.129), we get

$$P(\gamma_0^{(k)} < \theta(\mathcal{A}_-^{(k)}, \mathcal{S}_0^{(k)})) \to 1 \ \text{ as } \ k \to \infty. \tag{3.133}$$

3.4 Relation with sparse recovery

Sparse recovery is the problem of recovering a sparse signal $\boldsymbol{c}$ from linear measurements $\boldsymbol{b} = \mathrm{A}\boldsymbol{c}$. As we have seen in Chapter 2, sparse recovery can be achieved by BP and OMP if the dictionary A satisfies the incoherence condition

or has the restricted isometry properties.

The problem of sparse recovery can be considered as a particular case of the problem of subspace-preserving recovery. Let us take $\mathcal{A} = \mathcal{A}_0 \cup \mathcal{A}_-$ in which $\mathcal{A}_0$ contains the columns of $\mathbf{A}$ corresponding to the nonzero entries of $\boldsymbol{c}$, and $\mathcal{A}_-$ correspond to the rest of the columns of $\mathbf{A}$. If the atoms in $\mathcal{A}_0$ are linearly independent, then there is only one subspace-preserving solution, which is the sparse signal $\boldsymbol{c}$. In this case, $\boldsymbol{c}$ can be recovered if subspace-preserving recovery can be achieved. From this observation, we can formulate the following result which can be derived by directly applying the PRC in Theorem 12 and the DRC in Theorem 13.

Theorem 26 (Guaranteed sparse recovery via PRC and DRC)**.** *Given a dictionary $\mathcal{A}$, any s_0-sparse vector $\boldsymbol{c}$ can be recovered from the observation $\boldsymbol{b} = \mathbf{A}\boldsymbol{c}$ by BP and OMP if for any s_0 atoms in $\mathcal{A}$, denoted by $\mathcal{A}_0$, it has 1) atoms in $\mathcal{A}_0$ are linearly independent, and 2) the PRC (respectively, the DRC) holds for $\mathcal{A} = \mathcal{A}_0 \cup \mathcal{A}_-$.*

This result gives new conditions for guaranteeing correct recovery of sparse signals which does not use the incoherence or restricted isometry properties. Note that the requirement that any s_0 columns are linearly independent is necessary for the uniqueness of s_0 sparse solutions. The requirement that the PRC/DRC is satisfied has the same geometric interpretation as that of the PRC and the DRC for the subspace-preserving recovery, i.e., any s_0 atoms of the

dictionary should be well separated and all other atoms should be sufficiently away from the span of these s_0 atoms (by the PRC) or from a subset of the span of them (by the DRC).

For practical purposes, Theorem 26 also has the benefit that the conditions can be checked, as explained below. First, the set of dual points $\mathcal{D}_0$ can be written out explicitly when $\mathcal{A}_0$ contains linearly independent point.

Lemma 13. *Assume that $\mathcal{A}_0$ contains s_0 linearly independent atoms. The set of dual points, $\mathcal{D}_0$, contains exactly 2^{s_0} points specified by $\{\mathbf{A}_0(\mathbf{A}_0^\top\mathbf{A}_0)^{-1}\cdot\boldsymbol{u}, \boldsymbol{u}\in U_{s_0}\}$, where $U_{s_0}:=\{[u_1,\cdots,u_{s_0}], u_i=\pm 1, i=1,\cdots,s_0\}$.*

Proof. From Lemma 4, there are possibly at most 2^{s_0} dual points in the case where $\mathbf{A}_0$ is of full column rank. So in order to prove the result, it is enough to show that the set $\{\mathbf{A}_0(\mathbf{A}_0^\top\mathbf{A}_0)^{-1}\cdot\boldsymbol{u}, \boldsymbol{u}\in U_{s_0}\}$ contains 2^{s_0} points, and each of them is a dual point.

To show that there are 2^{s_0} different points, notice that U_{s_0} has 2^{s_0} points, so we are left to show that for any $\boldsymbol{u}_1, \boldsymbol{u}_2\in U_{s_0}$ with $\boldsymbol{u}_1\neq\boldsymbol{u}_2$, it has $\mathbf{A}_0(\mathbf{A}_0^\top\mathbf{A}_0)^{-1}\boldsymbol{u}_1\neq \mathbf{A}_0(\mathbf{A}_0^\top\mathbf{A}_0)^{-1}\boldsymbol{u}_2$. This can be easily established by noticing that $\mathrm{rank}(\mathbf{A}_0(\mathbf{A}_0^\top\mathbf{A}_0)^{-1}) = \mathrm{rank}(\mathbf{A}_0)=s_0$, i.e., $\mathbf{A}_0(\mathbf{A}_0^\top\mathbf{A}_0)^{-1}$ is also of full column rank, so its null space contains only the origin. Consequently, if $\mathbf{A}_0(\mathbf{A}_0^\top\mathbf{A}_0)^{-1}\boldsymbol{u}_1 = \mathbf{A}_0(\mathbf{A}_0^\top\mathbf{A}_0)^{-1}\boldsymbol{u}_2$, then $\boldsymbol{u}_1=\boldsymbol{u}_2$, which is a contradiction.

Now we show that $\mathbf{A}_0(\mathbf{A}_0^\top\mathbf{A}_0)^{-1}\boldsymbol{u}_0$ is a dual point for any $\boldsymbol{u}_0\in U_{s_0}$. Denote $\boldsymbol{v}_0 = \mathbf{A}_0(\mathbf{A}_0^\top\mathbf{A}_0)^{-1}\boldsymbol{u}_0$. By definition, we need to show that $\boldsymbol{v}_0$ is an extreme

point of the set $\mathcal{K}_0^o = \{\boldsymbol{v} \in \mathcal{S}_0 : \|\mathbf{A}_0^\top \boldsymbol{v}\|_\infty \leq 1\}$. First, $\boldsymbol{v}_0$ is in $\mathcal{K}_0^o$ because $\|\mathbf{A}_0^\top \boldsymbol{v}_0\|_\infty = \|\boldsymbol{u}_0\|_\infty = 1$. Second, suppose there are two points, $\boldsymbol{v}_1, \boldsymbol{v}_2 \in \mathcal{K}_0^o$, such that

$$\boldsymbol{v}_0 = (1-\lambda)\boldsymbol{v}_1 + \lambda \boldsymbol{v}_2 \tag{3.134}$$

for some $\lambda \in (0,1)$, we need to show that it must be the case that $\boldsymbol{v}_1 = \boldsymbol{v}_2$. Notice that the columns of $\mathbf{A}_0(\mathbf{A}_0^\top \mathbf{A}_0)^{-1}$ span the space $\mathcal{S}_0$ and that $\boldsymbol{v}_1, \boldsymbol{v}_2 \in \mathcal{K}_0^o \subseteq \mathcal{S}_0$, there exist $\boldsymbol{c}_1, \boldsymbol{c}_2$ such that $\boldsymbol{v}_i = \mathbf{A}_0(\mathbf{A}_0^\top \mathbf{A}_0)^{-1}\boldsymbol{c}_i, i = 1, 2$. Then by using (3.134), it has

$$\mathbf{A}_0(\mathbf{A}_0^\top \mathbf{A}_0)^{-1}\boldsymbol{u}_0 = (1-\lambda)\mathbf{A}_0(\mathbf{A}_0^\top \mathbf{A}_0)^{-1}\boldsymbol{c}_1 + \lambda \mathbf{A}_0(\mathbf{A}_0^\top \mathbf{A}_0)^{-1}\boldsymbol{c}_2, \tag{3.135}$$

and by left multiplying $\mathbf{A}_0^\top$, we have

$$\boldsymbol{u}_0 = (1-\lambda)\boldsymbol{c}_1 + \lambda \boldsymbol{c}_2. \tag{3.136}$$

Now, consider equation (3.136) for each entry separately, i.e., $[\boldsymbol{u}_0]_i = (1-\lambda)[\boldsymbol{c}_1]_i + \lambda[\boldsymbol{c}_2]_i$, where i indexes an entry in the vector. The left hand side, being ± 1, is a extreme point of the set $[-1, 1]$, while the right hand side is the convex combination of two points in $[-1, 1]$, so it necessarily has that $[\boldsymbol{c}_1]_i = [\boldsymbol{c}_2]_i$. This is true for all entries i, so $\boldsymbol{c}_1 = \boldsymbol{c}_2$, thus $\boldsymbol{v}_1 = \boldsymbol{v}_2$, which shows that $\boldsymbol{v}_0$ is indeed an extreme point. $\square$

Given the dual points, one can then compute $\theta(\mathcal{A}_-, \mathcal{D}_0)$ on the RHS of the DRC. Moreover, the covering radius γ_0 can be computed by the relation in Lemma 8, i.e.

$$\cos\gamma_0 = 1/\max\{\|\boldsymbol{v}\|_2 : \boldsymbol{v} \in \mathcal{K}_0^o\} = 1/\max\{\|\boldsymbol{v}\|_2 : \boldsymbol{v} \in \mathcal{D}_0\}, \tag{3.137}$$

where the last equality follows from the fact that $\mathcal{D}_0$ is the set of extreme points of $\mathcal{K}_0^o$. Thus, all terms in the PRC/DRC can be computed and the conditions in Theorem 26 can be checked.

To finish the discussion of Theorem 26, we compare it with canonical results for sparse signal recovery. Specifically, from Theorem 3 we see that $\mu(\mathcal{A}) < \frac{1}{2s_0-1}$ is a sufficient condition for BP and OMP to recover any s_0-sparse signals. The next theorem states that this is a stronger requirement than that of Theorem 26.

Theorem 27. *If a dictionary $\mathcal{A}$ satisfies $\mu(\mathcal{A}) < \frac{1}{2s_0-1}$, then for any partition of $\mathcal{A}$ into $\mathcal{A}_0$ and $\mathcal{A}_-$ where $\operatorname{card}(\mathcal{A}_0) = s_0$, it has that the atoms in $\mathcal{A}_0$ are linearly independent and that both PRC and DRC hold.*

Proof. Suppose $\mu(\mathcal{A}) < 1/(2s_0 - 1)$. It is well-known in the study of sparse recovery (e.g. [53]) that the columns of $\mathbf{A}_0$ are linearly independent. In the following, we only need to show that the PRC is true, as the DRC is implied by the PRC.

We start by giving an upper bound on $1/\cos\gamma_0$. From Lemma 13, given any $\boldsymbol{v} \in \mathcal{K}_0^o$ where $\boldsymbol{v} \neq 0$, it can be written as $\boldsymbol{v} = \mathbf{A}_0(\mathbf{A}_0^\top \mathbf{A}_0)^{-1}\boldsymbol{u}$ for some $\boldsymbol{u} \neq 0$ with $\|\boldsymbol{u}\|_\infty \leq 1$. Thus,

$$\|\boldsymbol{v}\|_2^2 = \boldsymbol{v}^\top \boldsymbol{v} = \boldsymbol{u}^\top (\mathbf{A}_0^\top \mathbf{A}_0)^{-1}\boldsymbol{u} \leq s_0 \cdot \frac{\boldsymbol{u}^\top (\mathbf{A}_0^\top \mathbf{A}_0)^{-1}\boldsymbol{u}}{\boldsymbol{u}^\top \boldsymbol{u}}.$$

Denote $\lambda_{\max}(\cdot), \lambda_{\min}(\cdot)$ to be the maximum and minimum eigenvalue of a symmetric matrix, respectively. We get

$$\begin{aligned}\|\boldsymbol{v}\|_2^2 &\leq s_0 \cdot \max_{\boldsymbol{u}\neq 0} \frac{\boldsymbol{u}^\top (\mathbf{A}_0^\top \mathbf{A}_0)^{-1}\boldsymbol{u}}{\boldsymbol{u}^\top \boldsymbol{u}} \\ &= s_0 \cdot \lambda_{\max}(\mathbf{A}_0^\top \mathbf{A}_0)^{-1} = \frac{s_0}{\lambda_{\min}(\mathbf{A}_0^\top \mathbf{A}_0)}.\end{aligned}$$

Notice that $\mathbf{A}_0^\top \mathbf{A}_0$ is close to an identity matrix, i.e., its diagonals are 1 and the magnitude of each off-diagonal entry is bounded above by $\mu(\mathcal{A})$. By using Gersgorin's disc theorem, $\lambda_{\min}(\mathbf{A}_0^\top \mathbf{A}_0) \geq 1 - (s_0 - 1)\mu(\mathcal{A})$, so

$$\|\boldsymbol{v}\|_2^2 \leq \frac{s_0}{1 - (s_0 - 1)\mu(\mathcal{A})}.$$

As a consequence, $1/\cos\gamma_0 \leq \sqrt{\frac{s_0}{1-(s_0-1)\mu(\mathcal{A})}}$ by Lemma 8.

We now give an upper bound for the right hand side of the PRC. By definition,

$$\cos\theta(\mathcal{A}_-, \mathcal{S}_0) = \max_{\substack{\boldsymbol{v}\in\mathcal{S}_0, \\ \|\boldsymbol{v}\|_2=1}} \|\mathbf{A}_-^\top \boldsymbol{v}\|_\infty.$$

We thus need to bound $\|\mathbf{A}_{-}^{\top}\boldsymbol{v}\|_{\infty}$ for any $\boldsymbol{v} \in \mathcal{S}_0$ with $\|\boldsymbol{v}\|_2 = 1$. Consider the optimization program

$$\boldsymbol{c}^* = \arg\min_{\boldsymbol{c}} \|\boldsymbol{c}\|_1 \quad \text{s.t.} \;\; \boldsymbol{v} = \mathbf{A}_0\boldsymbol{c}.$$

and its dual program

$$\max_{\omega} \langle \omega, \boldsymbol{v} \rangle \quad \text{s.t.} \;\; \|\mathbf{A}_0^{\top}\omega\|_{\infty} \leq 1.$$

The strong duality holds since the primal problem is feasible, and the objective of the dual is bounded by $\|\omega\|_2\|\boldsymbol{v}\|_2 \leq 1/\cos\gamma_0$. Consequently, it has $\|\boldsymbol{c}^*\|_1 \leq 1/\cos\gamma_0$. This leads to

$$\|\mathbf{A}_{-}^{\top}\boldsymbol{v}\|_{\infty} = \|\mathbf{A}_{-}^{\top}\mathbf{A}_0\boldsymbol{c}^*\|_{\infty} \leq \|\mathbf{A}_{-}^{\top}\mathbf{A}_0\|_{\infty}\|\boldsymbol{c}^*\|_1 \leq \mu(\mathcal{A})/\cos\gamma_0, \tag{3.138}$$

in which $\|\cdot\|_{\infty}$ for matrix treats the matrix as a vector.

Now we combine the results from the above two parts.

$$\cos\theta(\mathcal{A}_{-}, \mathcal{S}_0) \leq \mu(\mathcal{A})/\cos\gamma_0 = \cos\gamma_0 \cdot (\mu(\mathcal{A})/\cos\gamma_0^2) \leq \cos\gamma_0 \frac{s_0\mu(\mathcal{A})}{1-(s_0-1)\mu(\mathcal{A})}, \tag{3.139}$$

in which

$$\frac{s_0\mu(\mathcal{A})}{1-(s_0-1)\mu(\mathcal{A})} = 1 + \frac{\mu(\mathcal{A})(2s_0-1)-1}{1-(s_0-1)\mu} < 1,$$

thus $\cos\theta(\mathcal{A}_-, \mathcal{S}_0) < \cos\gamma_0$, which is the PRC. □

This result shows that the PRC/DRC conditions in Theorem 26 are implied by the condition of mutual coherence. While the mutual coherence condition requires all atoms of $\mathcal{A}$ to be incoherent from each other, the PRC and the DRC provide more detailed requirements, in terms of the distribution of points $\mathcal{A}_0$ as well as the relation of $\mathcal{A}_0$ and $\mathcal{A}_-$.

3.5 Applications to multi-subspace learning

In this section, we derive conditions for subspace-preserving recovery in the sparse representation based classification (SRC) and the sparse subspace clustering (SSC) methods. We will start our analysis with the case where the subspaces are independent.

Definition 19 (Independent subspaces)**.** A collection of subspaces $\{\mathcal{S}_\ell\}_{\ell=1}^n$ is called independent if $\dim\big(\sum_\ell \mathcal{S}_\ell\big) = \sum_\ell \dim(\mathcal{S}_\ell)$, where $\sum_\ell \mathcal{S}_\ell$ is defined as the subspace $\{\sum_\ell \boldsymbol{x}_\ell : \boldsymbol{x}_\ell \in \mathcal{S}_\ell\}$.

Notice that two subspaces are *independent* if and only if they are *disjoint*, i.e., if they intersect only at the origin. However, pairwise disjoint subspaces

need not be independent, e.g., three lines in $\mathbb{R}^2$ are disjoint but not independent. Notice also that any subset of a set of independent subspaces is also independent. Therefore, any two subspaces in a set of independent subspaces are independent and hence disjoint. In particular, this implies that if $\{\mathcal{S}_\ell\}_{\ell=1}^n$ are independent, then $\mathcal{S}_\ell$ and $\mathcal{S}_{(-\ell)} := \sum_{\kappa \neq \ell} \mathcal{S}_\kappa$ are independent.

Under the independent subspace model, we will show that both the SRC and the SSC provably produce subspace-preserving representations regardless of how the data points are arranged on the subspaces (other than the requirement that there are *enough* data points from each subspace). Furthermore, by considering both the arrangement of the subspaces as well as the arrangement of data points on the subspaces, we extend the analysis to the case where the subspaces need not be independent and subspace-preserving recovery can still be guaranteed. Finally, we will show that such conditions on the arrangement of subspaces and data points can be satisfied with high probability if both the subspaces and the data points are drawn according to a probabilistic model.

3.5.1 Theoretical analysis of SRC

In the subspace classification problem (see Definition 4), we are given a training data set $\mathcal{X} := \{\boldsymbol{x}_j\}_{j=1}^N$ that contains data from a union of n subspaces. That is, there exists a partition of $\mathcal{X}$ into $\mathcal{X}^1, \cdots, \mathcal{X}^n$, such that $\mathcal{X}^\ell$ contains points from a low dimensional subspace $\mathcal{S}_\ell$. Te goal is to classify any test data

$\boldsymbol{x}$, which is known to lie in the union of subspaces $\cup_{\ell=1}^{n}\mathcal{S}_\ell$, to the subspace it belongs to. The SRC solves the classification problem by finding a sparse representation vector $\boldsymbol{c}$ such that $\boldsymbol{x} = \mathbf{X}\boldsymbol{c}$ by using BP or OMP. It is then expected that the representation vector $\boldsymbol{c}$ is subspace-preserving, i.e., the nonzero entries of $\boldsymbol{c}$ correspond to columns of $\mathcal{X}$ that are in the same subspace of $\boldsymbol{x}$. Therefore, $\boldsymbol{x}$ can be correctly classified by assigning it to the class to which points in the training data $\mathcal{X}$ corresponding to nonzero entries of $\boldsymbol{c}$ belongs (see Algorithm 2).

Independent subspace model. We first consider the case where the collection of subspaces $\cup_{\ell=1}^{n}\mathcal{S}_\ell$ is independent. If the sparse recovery in SRC is solved by BP, then subspace-preserving recovery can be achieved for any test point in the union of subspaces. More specifically, we have the following result which is reformulated from [60].

Theorem 28 (Subspace-preserving recovery for SRC-BP: independent subspace model)**.** *Consider training data $\mathcal{X} = \cup_{\ell=1}^{n}\mathcal{X}^\ell$ that lie in a union of* independent *subspaces $\{\mathcal{S}_\ell\}_{\ell=1}^{n}$. Assume that $\mathrm{span}(\mathcal{X}^\ell) = \mathcal{S}_\ell$ for each $\ell = 1, \cdots, n$. Then, all elements in $\mathrm{BP}(\mathcal{X}, \boldsymbol{x})$ are subspace-preserving for any point $\boldsymbol{x} \in \cup_{\ell=1}^{n}\mathcal{S}_\ell$.*

We refer the reader to [60] for a proof of Theorem 28. In the following, we proceed to the analysis of SRC-OMP. Note that when computing sparse representation of a test data $\boldsymbol{x}$ using the training data $\mathcal{X}$, the goal is to select

points in the same subspace as $\boldsymbol{x}$. The process for selecting these points occurs in step 3 of Algorithm 1, where the dot products between all points $\boldsymbol{x}_j$ and the current residual $\boldsymbol{v}^{(k)}$ are computed and the point with the highest product (in absolute value) is chosen. Since in the first iteration the residual is $\boldsymbol{v}^{(0)} = \boldsymbol{x}$, we could immediately choose a point $\boldsymbol{x}_j$ in the wrong subspace whenever the dot product of $\boldsymbol{x}_j$ with a point in a wrong subspace is larger than the dot product of $\boldsymbol{x}_j$ with points in the subspace it lies in. What the following theorem shows is that, even though OMP may select points in the wrong subspaces as the iterations proceed, the coefficients associated to points in other subspaces will be zero at the end. Therefore, OMP is guaranteed to find a subspace-preserving representation.

Theorem 29 (Subspace-preserving recovery for SRC-OMP: independent subspace model)**.** *Consider training data* $\mathcal{X} = \cup_{\ell=1}^{n} \mathcal{X}^{\ell}$ *that lie in a union of* independent *subspaces* $\{\mathcal{S}_\ell\}_{\ell=1}^{n}$. *Assume that* $\operatorname{span}(\mathcal{X}^{\ell}) = \mathcal{S}_\ell$ *for each* $\ell = 1, \cdots, n$. *Then, all elements in* $\operatorname{OMP}(\mathcal{X}, \boldsymbol{x})$ *are subspace-preserving for any point* $\boldsymbol{x} \in \cup_{\ell=1}^{n} \mathcal{S}_\ell$.

Proof. Without the loss of generality we take a test point $\boldsymbol{x} \in \mathcal{S}_\ell$. We need to show that the output of OMP is subspace-preserving. As an assumption, the termination parameters in OMP are set to be $\epsilon = 0$ and $k_{\max} = N$ (i.e., the total number of points in the dictionary $\mathcal{X}$). This means, in particular, that OMP always terminates with some iteration $k^* \leq N$ with $\boldsymbol{v}^{(k^*)} = 0$, which can be seen to hold as follows. If the OMP algorithm computes $\boldsymbol{v}^{(k)} = 0$ for

some $k \leq N-1$, then there is nothing to prove. Thus, to complete the proof, we suppose that $\boldsymbol{v}^{(k)} \neq 0$ for all $0 \leq k \leq N-1$, and proceed to prove that $\boldsymbol{v}^{(N)} = 0$. In the OMP algorithm, the data points in $\mathcal{X}$ that are indexed by $\mathcal{W}^{(k)}$ for any k are always linearly independent. This is evident from step 4 of Algorithm 1, as the residual vector $\boldsymbol{v}^{(k)}$ is orthogonal to every data points in $\mathcal{X}$ that are indexed by $\mathcal{W}^{(k)}$, thus when choosing a new entry to be added to $\mathcal{W}^{(k)}$ in step 3 of Algorithm 1, points that are linearly dependent with the points indexed by $\mathcal{W}^{(k)}$ would have zero inner product with $\boldsymbol{v}^{(k)}$, so would not be picked. Since all of the data points in $\mathcal{X}$ have been added by iteration N, we know that data points in $\mathcal{X}$ are linearly independent and must contain at least d_ℓ linearly independent vectors from $\mathcal{S}_\ell$. We conclude that $\boldsymbol{v}^{(k^*)} = \boldsymbol{v}^{(N)} = 0$ with $k^* = N$, as claimed. In light of this result and denoting $\mathcal{W}^* := \mathcal{W}^{(k^*)}$, it follows from $\boldsymbol{v}^{(k^*)} = \mathbf{0}$ that $P_{\mathcal{W}^*} \cdot \boldsymbol{x} = \boldsymbol{x}$ by step 4 of Algorithm 1, so that $\boldsymbol{x}$ is in the span of $\{\boldsymbol{x}_j \in \mathcal{X} : j \in \mathcal{W}^*\}$.

As a consequence of the previous paragraph, the final output of OMP, given by

$$\boldsymbol{c}^* = \underset{\boldsymbol{c}:\text{Supp}(\boldsymbol{c}) \subseteq \mathcal{W}^*}{\arg\min} \|\boldsymbol{x} - \mathbf{X}\boldsymbol{c}\|_2,$$

will satisfy $\boldsymbol{x} = \mathbf{X} \cdot \boldsymbol{c}^*$. We rewrite it as

$$\boldsymbol{x} - \sum_{\substack{j:\boldsymbol{x}_j \in \mathcal{S}_\ell \\ j \in \mathcal{W}^{(k^*)}}} \boldsymbol{x}_j \cdot c_j^* = \sum_{\substack{j:\boldsymbol{x}_j \notin \mathcal{S}_\ell \\ j \in \mathcal{W}^{(k^*)}}} \boldsymbol{x}_j \cdot c_j^*. \tag{3.140}$$

Observe that the left hand side of (3.140) is in subspace $\mathcal{S}_\ell$ while the right hand side is in subspace $\mathcal{S}_{-\ell}$. By the assumption that the set of all subspaces is independent, we know $\mathcal{S}_\ell$ and $\mathcal{S}_{-\ell}$ are also independent, so they intersect only at the origin. As a consequence, we have

$$0 = \sum_{\substack{j:\boldsymbol{x}_j \notin \mathcal{S}_\ell \\ j \in \mathcal{W}^{(k^*)}}} \boldsymbol{x}_j \cdot c_j^* = \sum_{j:\boldsymbol{x}_j \notin \mathcal{S}_\ell} \boldsymbol{x}_j \cdot c_j^*, \tag{3.141}$$

where we also used the fact that $c_j^* = 0$ for all $j \notin \mathcal{W}^{(k^*)}$. Combining (3.141) with the early fact that the points $\boldsymbol{x}_j : j \in \mathcal{W}^{(k)}$ are linearly independent for all k (this includes $k = k^*$), we know that

$$c_j^* = 0 \text{ if } \boldsymbol{x}_j \notin \mathcal{S}_\ell \text{ and } j \in \mathcal{W}^{(k^*)}. \tag{3.142}$$

Finally, we use this to prove that $\boldsymbol{c}^*$ is subspace-preserving. To this end, suppose that $c_j^* \neq 0$, which from the definition of $\boldsymbol{c}^*$ means that $j \in \mathcal{W}^{(k^*)}$. Using this fact, $c_j^* \neq 0$, and (3.142) allows us to conclude that $c_j^* \in \mathcal{S}_\ell$. Thus the solution c^* is subspace-preserving. □

Arbitrary subspace model. We now proceed to the analysis where the subspaces are not necessarily independent. Assume that all the points in $\mathcal{X}$ are normalized to have unit ℓ_2 norm. By applying the universal recovery condition in Theorem 13 to each of the subspaces, we can get the following result which

establishes the correctness of SRC.

Theorem 30 (Subspace-preserving recovery for SRC)**.** *Given training data* $\mathcal{X} = \cup_{\ell=1}^{n} \mathcal{X}^{\ell}$ *that lie in a union of subspaces* $\{\mathcal{S}_\ell\}_{\ell=1}^{n}$*, Then, all elements in* $\mathrm{BP}(\mathcal{X}, \boldsymbol{x})$ *and* $\mathrm{OMP}(\mathcal{X}, \boldsymbol{x})$ *are subspace-preserving for any point* $\boldsymbol{x} \in \cup_{\ell=1}^{n} \mathcal{S}_\ell$ *if*

$$r_\ell > \max_{\boldsymbol{v} \in \mathcal{D}_\ell} \max_{\boldsymbol{x} \in \mathcal{X}^{-\ell}} |\langle \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}, \boldsymbol{x} \rangle|, \quad \forall \ell = 1, \cdots, n, \tag{3.143}$$

where r_ℓ *is the inradius of the convex hull of* $\pm \mathcal{X}^{\ell}$*,* $\mathcal{D}_\ell$ *is the set of dual points of* $\mathcal{X}^{\ell}$*, and* $\mathcal{X}^{-\ell} := \mathcal{X} - \mathcal{X}^{\ell}$ *is the set of all points not in* $\mathcal{S}_\ell$*.*

This theorem asserts that SRC is correct if the DRC is satisfied for each of the subspaces, i.e., if the training data in each of the subspaces are well-distributed, and for each subspace the dual points $\mathcal{D}_\ell$ are sufficiently separated from training data in all other subspaces.

By extending Theorem 22, we have the following randomized result which reveals the effect of subspace dimension and number of subspaces on the correctness of SRC. For simplicity, we consider the case where all subspaces $\{\mathcal{S}_\ell\}_{\ell=1}^{n}$ have the same dimension d and all $\mathcal{X}^{\ell}$ contain the same number of points $\rho \cdot d$ ($\rho > 1$ is the "density" of points in each subspace).

Theorem 31 (Subspace-preserving recovery for SRC in random model)**.** *Given the training data* $\mathcal{X} = \cup_{\ell=1}^{n} \mathcal{X}^{\ell}$ *in which each* $\mathcal{X}^{\ell}$ *contains* $\rho \cdot d$ *points drawn independently and uniformly at random on the unit sphere of a randomly gen-*

erated subspace $\mathcal{S}_\ell \subseteq \mathbb{R}^D$ *of dimension* d*. Assume that* $1 < \rho < e^{d/2}$*. Let* $N(\rho, d, n) = \rho \cdot d \cdot n$ *be the total number of data points in* $\mathcal{X}$*. Then, under the condition that*

$$\frac{D}{d} > 1 + \frac{8 \log N(\rho, d, n) + 2d \log \frac{D}{d}}{c^2(\rho) \log(\rho)}, \tag{3.144}$$

the probability that all elements in $\mathrm{BP}(\mathcal{X}, \boldsymbol{x})$ *and* $\mathrm{OMP}(\mathcal{X}, \boldsymbol{x})$ *are subspace-preserving for all* $\boldsymbol{x} \in \cup_{\ell=1}^{n} \mathcal{S}_\ell$ *is at least* $1 - \frac{1}{\rho \cdot d} - n \cdot e^{-\sqrt{\rho} \cdot d}$*.*

Proof. Fix any $\ell \in \{1, \cdots, n\}$. We can apply Theorem 24 which gives us

$$P\Big(\cos \gamma_\ell \geq c(\rho) \sqrt{\frac{\log \rho}{2d}} \Big) \geq 1 - e^{-\sqrt{\rho} d}. \tag{3.145}$$

On the other hand, from Lemma 11 we get

$$P(\max_{\boldsymbol{v} \in \mathcal{S}_\ell} |\langle \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}, \boldsymbol{x} \rangle| \leq \cos \beta) \geq 1 - \sqrt{e \frac{D}{d_0}}^{d_0} \exp(-\frac{D - d_0}{2} \cos^2 \beta), \tag{3.146}$$

for any fixed $\boldsymbol{x} \in \mathcal{X}^{-\ell}$. Applying a union bound we get

$$P(\max_{\boldsymbol{v} \in \mathcal{S}_\ell} \max_{\boldsymbol{x} \in \mathcal{X}^{-\ell}} |\langle \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}, \boldsymbol{x} \rangle| \leq \cos \beta) \geq 1 - (n-1)\rho d \cdot \sqrt{e \frac{D}{d_0}}^{d_0} \exp(-\frac{D - d_0}{2} \cos^2 \beta). \tag{3.147}$$

By taking $\cos\beta = \min\left(1, \sqrt{\frac{4\log N(\rho,d,n)+d_0\log(e\frac{D}{d_0})}{D-d_0}}\right)$, we further get

$$P(\max_{\boldsymbol{v}\in\mathcal{S}_\ell}\max_{\boldsymbol{x}\in\mathcal{X}^{-\ell}}|\langle\frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2},\boldsymbol{x}\rangle| \le \sqrt{\frac{4\log N(\rho,d,n)+d_0\log(e\frac{D}{d_0})}{D-d_0}}) \ge 1-\frac{(n-1)\rho d}{N(\rho,d,n)^2}$$
$$\ge 1-\frac{1}{N(\rho,d,n)} \quad (3.148)$$

Applying union bound to (3.148) and (3.145) we get

$$P(\cos\gamma_\ell > \max_{\boldsymbol{v}\in\mathcal{S}_\ell}\max_{\boldsymbol{x}\in\mathcal{X}^{-\ell}}|\langle\frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2},\boldsymbol{x}\rangle|) \ge 1-\exp(-\sqrt{\rho}d)-\frac{1}{N(\rho,d,n)}, \quad (3.149)$$

provided that the following condition holds:

$$\sqrt{\frac{4\log N(\rho,d,n)+d\log(e\frac{D}{d})}{D-d}} < c(\rho)\sqrt{\frac{\log\rho}{2d}}. \quad (3.150)$$

One can check that this condition is equivalent to the condition in (3.144). By applying a union bound, we get

$$P(\cos\gamma_\ell > \max_{\boldsymbol{v}\in\mathcal{S}_\ell}\max_{\boldsymbol{x}\in\mathcal{X}^{-\ell}}|\langle\frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2},\boldsymbol{x}\rangle| \text{ for all } \ell\in\{1,\cdots,n\})$$
$$\ge 1-n\exp(-\sqrt{\rho}d)-\frac{n}{N(\rho,d,n)} = 1-n\exp(-\sqrt{\rho}d)-\frac{1}{\rho\cdot d}, \quad (3.151)$$

provided that (3.144) is satisfied. Therefore, the conclusion of the theorem follows from Theorem 30. □

Theorem 31 shows that SRC is correct if the subspace dimension d is low enough relative to the ambient dimension D. In addition, it reflects the fact that as the number of subspaces n increases, subspace-preserving recovery becomes more difficult as the right hand side of condition (3.144) is increasing in n, making the condition harder to be satisfied. Moreover, the probability of success decreases with n.

3.5.2 Theoretical analysis of SSC

Let $\mathcal{X} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\}$ be a set of points drawn from a union of unknown subspaces $\{\mathcal{S}_\ell\}_{\ell=1}^n$. Subspace clustering addresses the problem of clustering these points into their respective subspaces, without knowing their membership *a priori* (see Definition 5).

The SSC solves the subspace clustering task by expressing each data point as a sparse linear combination of all other data points. Specifically, for each $\boldsymbol{x}_j \in \mathcal{X}$ we let $\mathcal{X}_{-j} = \mathcal{X} \setminus \{\boldsymbol{x}_j\}$ be the set of all other data points in the dataset. Assume that $\boldsymbol{x}_j$ lies in the subspace $\mathcal{S}_\ell$. Then, the set $\mathcal{X}_{-j}$ can be decomposed into two disjoint sets $\mathcal{X}_{-j} := \mathcal{X}_{-j}^\ell \cup \mathcal{X}^{-\ell}$, where $\mathcal{X}_{-j}^\ell$ contains all data points in $\mathcal{S}_\ell$ except $\boldsymbol{x}_j$ itself, and $\mathcal{X}^{-\ell}$ contains all data points in all subspaces except $\mathcal{S}_\ell$. If the elements in $\mathrm{BP}(\boldsymbol{x}_j, \mathcal{X}_{-j})$ and $\mathrm{OMP}(\boldsymbol{x}_j, \mathcal{X}_{-j})$ are subspace-preserving, then the nonzero entries of such sparse solutions correspond to data points in $\mathcal{X}_{-j}^\ell$, which are also in the subspace $\mathcal{S}_\ell$. One can compute such sparse solutions for

all points $\boldsymbol{x}_j \in \mathcal{X}$, and clusters can be obtained by extracting the connected components from the similarity graph constructed from such sparse representations (see Algorithm 3).

Independent subspace model. If the collection of subspaces $\cup_{\ell=1}^{n} \mathcal{S}_\ell$ is independent, then both SSC-BP and SSC-OMP produce subspace-preserving representations. More specifically, we have the following result.

Theorem 32 (Subspace-preserving recovery for SSC: independent subspace model)**.** *Consider data points $\mathcal{X} = \cup_{\ell=1}^{n} \mathcal{X}^\ell$ that lie in a union of* independent *subspaces $\{\mathcal{S}_\ell\}_{\ell=1}^{n}$. Assume that $\operatorname{span}(\mathcal{X}_{-j}^{\ell}) = \mathcal{S}_\ell$ for each $j : \boldsymbol{x}_j \in \mathcal{S}_\ell$ and each $\ell = 1, \cdots, n$. Then, all elements in $\operatorname{BP}(\mathcal{X}_{-j}, \boldsymbol{x}_j)$ and $\operatorname{OMP}(\mathcal{X}_{-j}, \boldsymbol{x}_j)$ are subspace-preserving for all points $\boldsymbol{x}_j \in \mathcal{X}$.*

The proof of this theorem follows directly from the results in Theorem 28 and Theorem 29.

Arbitrary subspace model. Assume that all the points in $\mathcal{X}$ are normalized to have unit ℓ_2 norm. We can apply Theorem 10 with $\mathcal{A}_0$, $\mathcal{A}_-$ and $\boldsymbol{b}$ set to be $\mathcal{X}_{-j}^{\ell}$, $\mathcal{X}^{-\ell}$, and $\boldsymbol{x}_j$, respectively. Then, the condition in (3.29) guarantees the subspace-preserving recovery of $\boldsymbol{x}_j$ by BP. We can rephrase this result in terms of the properties of the subspaces to make it more interpretable. For each $\mathcal{S}_\ell$, we denote the minimum leave-one-out inradius of sample points in $\mathcal{S}_\ell$ as

$$r_\ell := \min_{j:\boldsymbol{x}_j \in \mathcal{S}_\ell} r(\mathcal{K}(\pm \mathcal{X}_{-j}^{\ell})). \tag{3.152}$$

Theorem 33 (Subspace-preserving recovery for SSC-BP)**.** *Given data* $\mathcal{X} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\}$ *that lie in a union of subspaces* $\{\mathcal{S}_\ell\}_{\ell=1}^n$*, all elements in* $\mathrm{BP}(\mathcal{X}_{-j}, \boldsymbol{x}_j)$ *are subspace-preserving for all points* $\boldsymbol{x}_j \in \mathcal{X}$ *if*

$$r_\ell > \max_{\boldsymbol{v} \in \mathcal{D}_\ell} \max_{\boldsymbol{x} \in \mathcal{X}^{-\ell}} |\langle \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}, \boldsymbol{x} \rangle|, \quad \forall \ell = 1, \cdots, n, \tag{3.153}$$

where r_ℓ *is defined in* (3.152)*.* $\mathcal{D}_\ell$ *is any set such that for all* $j : \boldsymbol{x}_j \in \mathcal{S}_\ell$*, it has* $\mathcal{D}_\ell \cap \mathcal{D}(\mathcal{X}_{-j}^\ell, \boldsymbol{x}_j) \neq \emptyset$*, i.e., it contains at least one point from each of* $\mathcal{D}(\mathcal{X}_{-j}^\ell, \boldsymbol{x}_j)$ *for all* $\boldsymbol{x}_j$ *in* $\mathcal{S}_\ell$*.*

Similarly, from Theorem 11 we can derive the following result which provides guarantee for subspace-preserving recovery in SSC-OMP.

Theorem 34 (Subspace-preserving recovery for SSC-OMP)**.** *Given data* $\mathcal{X} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\}$ *that lie in a union of subspaces* $\{\mathcal{S}_\ell\}_{\ell=1}^n$*, all elements in* $\mathrm{OMP}(\mathcal{X}_{-j}, \boldsymbol{x}_j)$ *are subspace-preserving for all points* $\boldsymbol{x}_j \in \mathcal{X}$ *if*

$$r_\ell > \max_{\boldsymbol{v} \in \mathcal{R}_\ell} \max_{\boldsymbol{x} \in \mathcal{X}^{-\ell}} |\langle \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}, \boldsymbol{x} \rangle|, \quad \forall \ell = 1, \cdots, n, \tag{3.154}$$

where r_ℓ *is defined in* (3.152)*, and* $\mathcal{R}_\ell := \cup_{j:\boldsymbol{x}_j \in \mathcal{S}_\ell} \mathcal{R}(\mathcal{X}_{-j}^\ell, \boldsymbol{x}_j)$ *is the union of the sets of residual points for all* $\boldsymbol{x}_j$ *in* $\mathcal{S}_\ell$*.*

Theorem 33 and Theorem 34 assert that SSC-BP and SSC-OMP produce subspace-preserving representations for all data points if the points in each

of the subspaces are well-distributed so that r_ℓ is large. Meanwhile, the set of dual points $\mathcal{D}_\ell$ in SSC-BP and the set of residual points $\mathcal{R}_\ell$ in SSC-OMP for each subspace $\mathcal{S}_\ell$ should be well separated from all data points in other subspaces.

We now consider the fully random union of subspaces model in [139], where the basis elements of each subspace are chosen uniformly at random from the unit sphere of the ambient space and the data points from each subspace are uniformly distributed on the unit sphere of that subspace. Theorem 35 shows that the sufficient conditions in Theorem 33 and Theorem 34 hold true with high probability (i.e. the probability goes to 1 as the density of points grows to infinity) given some conditions on the subspace dimension d, the ambient space dimension D, the number of subspaces n and the number of data points per subspace.

Theorem 35 (Subspace-preserving recovery for SSC in random model)**.** *Given the dataset $\mathcal{X} = \cup_{\ell=1}^{n} \mathcal{X}^\ell$ in which each $\mathcal{X}^\ell$ contains $\rho \cdot d + 1$ points drawn independently and uniformly at random on the unit sphere of a randomly generated subspace $\mathcal{S}_\ell \subseteq \mathbb{R}^D$ of dimension d. Assume that $1 < \rho < e^{d/2}$. Let $N(\rho, d, n) = (\rho \cdot d + 1) \cdot n$ be the total number of data points in $\mathcal{X}$. Then, under the condition that*

$$\frac{D}{d} > \frac{12 \log N(\rho, d, n)}{c^2(\rho) \log(\rho)}, \tag{3.155}$$

the probability that all elements in $\mathrm{BP}(\mathcal{X}_{-j}, \boldsymbol{x}_j)$ are subspace-preserving for all

points $\boldsymbol{x}_j \in \mathcal{X}$ *is at least* $1 - \frac{2}{N(\rho,d,n)} - N(\rho,d,n) \cdot e^{-\sqrt{\rho} \cdot d}$*, and the probability that all elements in* $\mathrm{OMP}(\mathcal{X}_{-j}, \boldsymbol{x}_j)$ *are subspace-preserving for all points* $\boldsymbol{x}_j \in \mathcal{X}$ *is at least* $1 - \frac{2d}{N(\rho,d,n)} - N(\rho,d,n) \cdot e^{-\sqrt{\rho} \cdot d}$.

3.5.2.1 Comparison with other work

We now compare our analysis of SSC-BP and SSC-OMP with prior results.

In [139], it is shown that SSC-BP produces subspace-preserving representations if the condition in (3.153) holds, but with a different definition of $\mathcal{D}_\ell$. In particular, [139] defines $\mathcal{D}_\ell$ as $\cup_{j:\boldsymbol{x}_j \in \mathcal{S}_\ell} \{\boldsymbol{v}_j : \boldsymbol{v}_j$ is a point in $\mathcal{D}(\mathcal{X}^\ell_{-j}, \boldsymbol{x}_j)$ that has the minimum Euclidean norm$\}$. Note that this definition of $\mathcal{D}_\ell$ is a particular case of the $\mathcal{D}_\ell$ defined in Theorem 33. Therefore, this result in [139] is a particular case of Theorem 33.

Finally, we compare our results with those in [55] for SSC-OMP. Define the principal angle between two subspaces $\mathcal{S}_\ell$ and $\mathcal{S}_k$ as:

$$\theta^*_{\ell,m} = \min_{\substack{\boldsymbol{v} \in \mathcal{S}_\ell \\ \|\boldsymbol{v}\|_2 = 1}} \min_{\substack{\boldsymbol{w} \in \mathcal{S}_m \\ \|\boldsymbol{w}\|_2 = 1}} \arccos \langle \boldsymbol{v}, \boldsymbol{w} \rangle. \tag{3.156}$$

It is shown in [55] that the output of SSC-OMP is subspace-preserving if for all $\ell = 1, \ldots, n$,

$$\max_{j:\boldsymbol{x}_j \in \mathcal{X}^\ell} \max_{k:\boldsymbol{x}_k \in \mathcal{X}^{-\ell}} |\langle \boldsymbol{x}_j, \boldsymbol{x}_k \rangle| < r_\ell - \frac{2\sqrt{1-(r_\ell)^2}}{\sqrt[4]{12}} \max_{m:m \neq \ell} \cos \theta^*_{\ell,m}. \tag{3.157}$$

The merit of this result is that it introduces the subspace angles in the condition, and satisfies the intuition that the algorithm is more likely to work if the subspaces are far apart from each other. However, the right hand side of the condition shows an intricate relationship between the intra-class property r_ℓ and the inter-class property $\theta^*_{\ell,m}$, which greatly complicates the interpretation of the condition. More importantly, as we show below, the condition is more restrictive than (3.154), which makes Theorem 34 a stronger result.

Notice that the inequality in (3.157) implies that $\forall m \neq \ell$,

$$\max_{j:\boldsymbol{x}_j \in \mathcal{X}^\ell} \max_{k:\boldsymbol{x}_k \in \mathcal{X}^m} |\langle \boldsymbol{x}_j, \boldsymbol{x}_k \rangle| < r_\ell - \sqrt{2 - 2r_\ell} \cos \theta^*_{\ell,m}, \tag{3.158}$$

see Lemma 1 in their paper. We consider the following two case.

Case 1. If $r_\ell \leq 1/2$, then $\sqrt{2 - 2r_\ell} \geq 1$, thus

$$\begin{aligned} (3.158) &\Rightarrow \max_{j:\boldsymbol{x}_j \in \mathcal{X}^\ell} \max_{k:\boldsymbol{x}_k \in \mathcal{X}^m} |\langle \boldsymbol{x}_j, \boldsymbol{x}_k \rangle| < r_\ell - \cos \theta^*_{\ell,m} \Rightarrow \cos \theta^*_{\ell,m} < r_\ell \\ &\Rightarrow \max_{\boldsymbol{v} \in \mathcal{R}_\ell} \max_{k:\boldsymbol{x}_k \in \mathcal{X}^m} |\langle \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}, \boldsymbol{x}_k \rangle| < r_\ell \Leftrightarrow (3.154). \end{aligned}$$

Case 2. If $r_\ell > 1/2$, then

$$
\begin{aligned}
(3.158) &\Rightarrow \max_{j:\boldsymbol{x}_j \in \mathcal{X}^\ell} \max_{k:\boldsymbol{x}_k \in \mathcal{X}^m} |\langle \boldsymbol{x}_j, \boldsymbol{x}_k \rangle| < r_\ell - \sqrt{2 - 2r_\ell} \max_{j:\boldsymbol{x}_j \in \mathcal{X}^\ell} \max_{k:\boldsymbol{x}_k \in \mathcal{X}^m} |\langle \boldsymbol{x}_j, \boldsymbol{x}_k \rangle| \\
&\Rightarrow \max_{j:\boldsymbol{x}_j \in \mathcal{X}^\ell} \max_{k:\boldsymbol{x}_k \in \mathcal{X}^m} |\langle \boldsymbol{x}_j, \boldsymbol{x}_k \rangle| < r_\ell / (1 + \sqrt{2 - 2r_\ell}) \\
&\Rightarrow \max_{j:\boldsymbol{x}_j \in \mathcal{X}^\ell} \max_{k:\boldsymbol{x}_k \in \mathcal{X}^m} |\langle \boldsymbol{x}_j, \boldsymbol{x}_k \rangle| < r_\ell / (1 + (2 - 2r_\ell)) \\
&\Rightarrow \max_{j:\boldsymbol{x}_j \in \mathcal{X}^\ell} \max_{k:\boldsymbol{x}_k \in \mathcal{X}^m} |\langle \boldsymbol{x}_j, \boldsymbol{x}_k \rangle| < (r_\ell)^2 \Rightarrow (3.154),
\end{aligned}
$$

see [193] for a proof of the last step. Therefore, we have shown that the condition in (3.157) is implied by (3.154).

3.5.3 Theoretical analysis of EnSC

From the theoretical analysis in the previous section, SSC is an appropriate method for the task of subspace clustering as it is guaranteed to produce subspace-preserving similarity matrices. However, the practical performance of SSC can be limited by the connectivity issue, which states that the data points from the same cluster may not form a connected component of the similarity graph due to the sparseness of the representation, causing over-segmentation in the clustering result.

In this section, we provide a theoretical analysis of elastic-net subspace clustering (EnSC) which uses a mixture of ℓ_1 and ℓ_2 norms to balance the subspace-preserving and connectedness properties of data similarity graph.

Recall that in EnSC, the self-representation coefficients are computed from solving the following optimization problem:

$$\min_{\boldsymbol{c}_j} \lambda\|\boldsymbol{c}\|_1 + \frac{1-\lambda}{2}\|\boldsymbol{c}\|_2^2 \quad \text{s.t. } \boldsymbol{x}_j = \mathbf{X}\boldsymbol{c}_j, c_{jj} = 0, \tag{3.159}$$

where $\lambda \in [0,1]$ controls the trade-off between the ℓ_1 and ℓ_2 regularizations. In particular, if $\lambda = 1$, then EnSC reduces to SSC-BP. Since the elastic net problem is more general than BP, we cannot directly apply previous results on the subspace-preserving recovery by BP. In the following, we will develop novel concepts and results for the analysis of EnSC.

While the data points in a subspace clustering setup satisfy the self-expressive model (i.e., $\boldsymbol{x}_j = \mathbf{X}\boldsymbol{c}_j$) in principle, practical data is often corrupted by noise. Therefore, instead of imposing self-expressiveness as an equality constraint, we hereafter consider the following optimization problem in which the self-representation is imposed in a penalty form, i.e.,

$$\min_{\boldsymbol{c}_j} \lambda\|\boldsymbol{c}_j\|_1 + \frac{1-\lambda}{2}\|\boldsymbol{c}_j\|_2^2 + \frac{\gamma}{2}\|\boldsymbol{x}_j - \mathbf{X}\boldsymbol{c}_j\|_2^2 \quad \text{s.t. } c_{jj} = 0 \tag{3.160}$$

where $\gamma > 0$ is a trade-off parameter between the elastic net regularization and the self-representation residual $\boldsymbol{x}_j - \mathbf{X}\boldsymbol{c}_j$.

We will provide theoretical conditions under which the similarity graph generated by EnSC via solving (3.160) is subspace-preserving, as well as a clear

geometric interpretation for the balance between the subspace-preserving and connectedness properties.

3.5.3.1 Geometry of the elastic-net solution

We first present a geometric analysis of the elastic net solution. Consider the objective function

$$f_{EN}(\boldsymbol{c};\ \boldsymbol{b}, \mathbf{A}) := \lambda\|\boldsymbol{c}\|_1 + \frac{1-\lambda}{2}\|\boldsymbol{c}\|_2^2 + \frac{\gamma}{2}\|\boldsymbol{b} - \mathbf{A}\boldsymbol{c}\|_2^2, \tag{3.161}$$

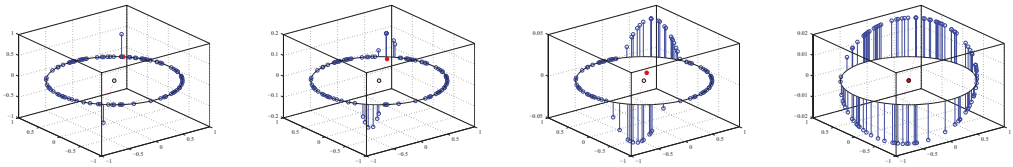
where $\boldsymbol{b} \in \mathbb{R}^D$, $\mathbf{A} = [\boldsymbol{a}_1, \cdots, \boldsymbol{a}_N] \in \mathbb{R}^{D\times N}$, $\gamma > 0$, and $\lambda \in [0, 1)$ (the reader is referred to the appendix of this chapter for a study of the case $\lambda = 1$). Without loss of generality, we assume that $\boldsymbol{b}$ and $\{\boldsymbol{a}_j\}_{j=1}^N$ are normalized to be of unit ℓ_2 norm in our analysis. The elastic net model then computes

$$\boldsymbol{c}^*(\boldsymbol{b}, \mathbf{A}) := \arg\min_{\boldsymbol{c}} f_{EN}(\boldsymbol{c};\ \boldsymbol{b}, \mathbf{A}). \tag{3.162}$$

We note that $\boldsymbol{c}^*(\boldsymbol{b}, \mathbf{A})$ is unique since $f_{EN}(\boldsymbol{c};\ \boldsymbol{b}, \mathbf{A})$ is a strongly convex function; we use the notation $\boldsymbol{c}^*$ in place of $\boldsymbol{c}^*(\boldsymbol{b}, \mathbf{A})$ when the meaning is clear.

A fundamental result that serves as the basis for the analysis of the elastic net solution in this section is the next lemma.

Lemma 14 ([51,86]). *The vector $\hat{\boldsymbol{c}} \in \mathbb{R}^N$ is the unique solution to* (3.162) *if and*



(a) $\lambda = 1$ **(b)** $\lambda = 0.9$ **(c)** $\lambda = 0.3$ **(d)** $\lambda = 0$

Figure 3.6: Illustration of the structure of the solution $\boldsymbol{c}^*$ for a data matrix $\mathbf{A}$ containing 100 randomly generated points in $\mathbb{R}^2$, which are shown as blue dots in the x-y plane. The z direction shows the magnitude for each coefficient $\boldsymbol{c}_j^*$. The red dot represents the oracle point $\boldsymbol{\delta}(\boldsymbol{b}, \mathbf{A})$, with its direction denoted by the red dashed line. The value for γ is fixed at 50, but the value for λ varies as depicted.

only if it satisfies

$$(1-\lambda)\hat{\boldsymbol{c}} = \mathcal{T}_\lambda\big(A^\top \cdot \gamma(\boldsymbol{b} - A\hat{\boldsymbol{c}})\big). \tag{3.163}$$

Proof. We provide a sketch of the proof for completeness. Since problem (3.162) is strongly convex, $\hat{\boldsymbol{c}}$ is the unique optimal solution if and only if it satisfies the following optimality condition:

$$A^\top \cdot \gamma(\boldsymbol{b} - A\hat{\boldsymbol{c}}) = (1-\lambda)\hat{\boldsymbol{c}} + \lambda z. \tag{3.164}$$

for some $z \in \partial\|\hat{\boldsymbol{c}}\|_1$. Then, by taking the soft-thresholding $\mathcal{T}_\lambda(\cdot)$ on both sides of (3.164) we get (3.163). For a proof of the reverse implication, suppose $\hat{\boldsymbol{c}}$ satisfies (3.163). For each $j = 1, \cdots, N$, by considering the three cases $\hat{\boldsymbol{c}}_j > 0$, $\hat{\boldsymbol{c}}_j = 0$, and $\hat{\boldsymbol{c}}_j < 0$ separately, one can establish that the j-th row of (3.164) is satisfied when the corresponding row of (3.163) holds. □

We first introduce the concept of an oracle point.

Definition 20 (Oracle Point)**.** The oracle point associated with the optimization problem (3.162) is defined to be

$$\boldsymbol{\delta}(\boldsymbol{b}, \mathbf{A}) := \gamma \cdot \big(\boldsymbol{b} - \mathbf{A}\boldsymbol{c}^*(\boldsymbol{b}, \mathbf{A})\big). \tag{3.165}$$

When there is no risk of confusion, we omit the dependency of the oracle point on $\boldsymbol{b}$ and $\mathbf{A}$ and write $\boldsymbol{\delta}(\boldsymbol{b}, \mathbf{A})$ as $\boldsymbol{\delta}$.

Notice that the oracle point is unique since $\boldsymbol{c}^*$ is unique, and that the oracle point cannot be computed until the optimal solution $\boldsymbol{c}^*$ has been computed.

The next result gives a critical relationship involving the oracle point that is exploited by our subsequent analysis. It follows directly from Lemma 14.

Theorem 36. *The solution $\boldsymbol{c}^*$ to problem* (3.162) *satisfies*

$$(1 - \lambda)\boldsymbol{c}^* = \mathcal{T}_\lambda(\mathbf{A}^\top \boldsymbol{\delta}), \tag{3.166}$$

where $\mathcal{T}_\lambda(\cdot)$ is the soft-thresholding operator (applied componentwise to $\mathbf{A}^\top\boldsymbol{\delta}$) defined as $\mathcal{T}_\lambda(v) = sgn(v)(|v| - \lambda)$ if $|v| > \lambda$ and 0 otherwise.

Theorem 36 shows that if the oracle point $\boldsymbol{\delta}$ is known, the solution $\boldsymbol{c}^*$ can be written out directly. Moreover, it follows from (3.165) and (3.166) that $\boldsymbol{\delta} = 0$ if and only if $\boldsymbol{b} = 0$.

In Figure 3.6, we depict a two dimensional example of the solution to the elastic net problem (3.162) for different values of the tradeoff parameter λ. As expected, the solution $\boldsymbol{c}^*$ becomes denser as λ decreases. Moreover, as predicted by Theorem 36, the magnitude of the coefficient c_j^* is a decaying function of the angle between the corresponding dictionary atom $\boldsymbol{a}_j$ and the oracle point $\boldsymbol{\delta}$ (shown in red). If $\boldsymbol{a}_j$ is far enough from $\boldsymbol{\delta}$ such that $|\langle \boldsymbol{a}_j, \boldsymbol{\delta} \rangle| \leq \lambda$ holds true, then the corresponding coefficient c_j^* is zero. We call the region containing the nonzero coefficients the *oracle region*. We can formally define the oracle region by using the quantity $\mu(\cdot, \cdot)$ to denote the coherence of two vectors, i.e.,

$$\mu(\boldsymbol{v}, \boldsymbol{w}) := \frac{|\langle \boldsymbol{v}, \boldsymbol{w} \rangle|}{\|\boldsymbol{v}\|_2 \|\boldsymbol{w}\|_2}. \tag{3.167}$$

Definition 21 (Oracle Region)**.** The oracle region associated with the optimization problem (3.162) is defined as

$$\boldsymbol{\Delta}(\boldsymbol{b}, \mathbf{A}) := \Big\{ \boldsymbol{v} \in \mathbb{R}^D : \|\boldsymbol{v}\|_2 = 1,\, \mu(\boldsymbol{v}, \boldsymbol{\delta}) > \frac{\lambda}{\|\boldsymbol{\delta}\|_2} \Big\}. \tag{3.168}$$

The oracle region is composed of an antipodal pair of spherical caps of the unit ball of $\mathbb{R}^D$ that are located at the symmetric locations $\pm \boldsymbol{\delta}/\|\boldsymbol{\delta}\|_2$, both with an angular radius of $\theta = \arccos(\lambda/\|\boldsymbol{\delta}\|_2)$ (see Figure 3.7). From the definition of the oracle region and Theorem 36, it follows that $c_j^* \neq 0$ if and only if $\boldsymbol{a}_j \in$

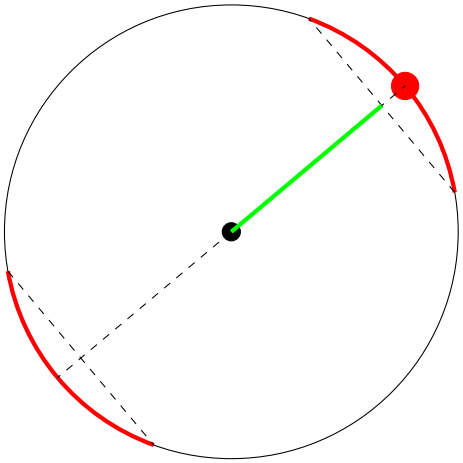


Figure 3.7: The oracle region $\boldsymbol{\Delta}(\boldsymbol{b}, \mathbf{A})$ is illustrated in red. Note that the size of the oracle region increases as the quantity $\lambda / \|\boldsymbol{\delta}\|_2$ decreases, and vice versa.

$\boldsymbol{\Delta}(\boldsymbol{b}, \mathbf{A})$. In other words, the support of the solution $\boldsymbol{c}^*$ are those vectors $\boldsymbol{a}_j$ in the oracle region.

The oracle region also captures the behavior of the solution when columns from the matrix $\mathbf{A}$ are removed or new columns are added. This provides the key insight into analyzing the subspace-preserving property of EnSC in the next section.

Proposition 1. *For any* $\boldsymbol{b} \in \mathbb{R}^D$, $\mathbf{A} \in \mathbb{R}^{D \times N}$ *and* $\mathbf{A}' \in \mathbb{R}^{D \times N'}$, *if no column of* $\mathbf{A}'$ *is contained in* $\boldsymbol{\Delta}(\boldsymbol{b}, \mathbf{A})$, *then* $\boldsymbol{c}^*(\boldsymbol{b}, [\mathbf{A}, \mathbf{A}']) = [\boldsymbol{c}^*(\boldsymbol{b}, \mathbf{A})^\top, \mathbf{0}_{N' \times 1}^\top]^\top$.

Proof. Notice that $\boldsymbol{c}^*(\boldsymbol{b}, A)$ satisfies

$$(1-\lambda)\boldsymbol{c}^*(\boldsymbol{b}, A) = \mathcal{T}_\lambda\big(A^\top \gamma(\boldsymbol{b} - A\boldsymbol{c}^*(\boldsymbol{b}, A))\big) \\ = \mathcal{T}_\lambda\left(A^\top \gamma\Big(\boldsymbol{b} - [A, A']\begin{bmatrix} \boldsymbol{c}^*(\boldsymbol{b}, A) \\ \\ \mathbf{0}_{N'\times 1} \end{bmatrix}\Big)\right). \quad (3.169)$$

Using the assumption that no column of A' is contained in $\boldsymbol{\Delta}(\boldsymbol{b}, A)$, it follows that

$$(1-\lambda)\mathbf{0}_{N'\times 1} = \mathcal{T}_\lambda\big(A'^\top \boldsymbol{\delta}(\boldsymbol{b}, A)\big) \\ = \mathcal{T}_\lambda\big(A'^\top \gamma(\boldsymbol{b} - A\boldsymbol{c}^*(\boldsymbol{b}, A))\big) \\ = \mathcal{T}_\lambda\left(A'^\top \gamma\Big(\boldsymbol{b} - [A, A']\begin{bmatrix} \boldsymbol{c}^*(\boldsymbol{b}, A) \\ \\ \mathbf{0}_{N'\times 1} \end{bmatrix}\Big)\right).$$

We may then combine this equality with (3.169) and define the vector $\hat{\boldsymbol{c}} := [\boldsymbol{c}^*(\boldsymbol{b}, A)^\top, \mathbf{0}_{N'\times 1}^\top]^\top$ to obtain

$$(1-\lambda)\hat{\boldsymbol{c}} = \mathcal{T}_\lambda\Big([A, A']^\top \gamma(\boldsymbol{b} - [A, A']\hat{\boldsymbol{c}})\Big), \quad (3.170)$$

thus by Lemma 14, $\hat{\boldsymbol{c}}$ must equal $\boldsymbol{c}^*(\boldsymbol{b}, [A, A'])$. □

The interpretation for Proposition 1 is that the solution $\boldsymbol{c}^*(\boldsymbol{b}, \mathbf{A})$ does not change (modulo padding with additional zeros) when new columns are added

to the dictionary $\mathbf{A}$, as long as the new columns are not inside the oracle region $\boldsymbol{\Delta}(\boldsymbol{b}, \mathbf{A})$. From another perspective, $\boldsymbol{c}^*(\boldsymbol{b}, [\mathbf{A}, \mathbf{A}'])$ does not change if one removes columns from the dictionary $[\mathbf{A}, \mathbf{A}']$ that are not in the oracle region $\boldsymbol{\Delta}(\boldsymbol{b}, [\mathbf{A}, \mathbf{A}'])$.

Proposition 2. *For any $\boldsymbol{b} \in \mathbb{R}^D$, $\mathbf{A} \in \mathbb{R}^{D \times N}$ and $\mathbf{A}' \in \mathbb{R}^{D \times N'}$, denote $\boldsymbol{c}^*(\boldsymbol{b}, [\mathbf{A}, \mathbf{A}']) = [\boldsymbol{c}_{\mathbf{A}}^\top, \boldsymbol{c}_{\mathbf{A}'}^\top]^\top$. If any column of $\mathbf{A}'$ lies within $\boldsymbol{\Delta}(\boldsymbol{b}, \mathbf{A})$, then $\boldsymbol{c}_{\mathbf{A}'}^\top \neq \mathbf{0}$.*

Proof. We prove the contrapositive; let $\boldsymbol{c}_{A'} = \mathbf{0}$. It then follows from $\boldsymbol{c}_A = \boldsymbol{c}^*(\boldsymbol{b}, A)$ that $\boldsymbol{c}^*(\boldsymbol{b}, [A, A']) = [\boldsymbol{c}^*(\boldsymbol{b}, A)^\top, \mathbf{0}^\top]$, and by definition of the oracle point that $\boldsymbol{\delta}(\boldsymbol{b}, A) = \boldsymbol{\delta}(\boldsymbol{b}, [A, A'])$. Now by Theorem 36, we have

$$(1 - \lambda) \begin{bmatrix} \boldsymbol{c}^*(\boldsymbol{b}, A) \\ \\ \mathbf{0} \end{bmatrix} = \mathcal{T}_\lambda \left(\begin{bmatrix} A^\top \\ \\ A'^\top \end{bmatrix} \cdot \boldsymbol{\delta}(\boldsymbol{b}, A) \right). \tag{3.171}$$

From the second block of equations and the definition of $\boldsymbol{\Delta}(\boldsymbol{b}, A)$, we have that no column of A' lies in the oracle region $\boldsymbol{\Delta}(\boldsymbol{b}, A)$, which completes the contrapositive proof. □

This result means that the solution to the elastic net problem will certainly be changed by adding new columns that lie within the oracle region to the dictionary.

3.5.3.2 Subspace-preserving vs. connected solutions

Although the elastic-net has been recently introduced for subspace clustering in [63, 128], these works do not provide conditions under which the affinity is guaranteed to be subspace-preserving or potential improvements in connectivity. In this section, we give conditions for the affinity to be subspace-preserving and for the balance between the subspace-preserving and connectedness properties. To the best of our knowledge, this is the first time that such theoretical guarantees have been established.

Let $\mathbf{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N] \in \mathbb{R}^{D \times N}$ be a real-valued matrix whose columns are drawn from a union of n subspaces of $\mathbb{R}^D$, say $\bigcup_{\ell=1}^{n} \mathcal{S}_\ell$. The goal of subspace clustering is to segment the columns of $\mathbf{X}$ into their representative subspaces (see Definition 5). In our analysis, we assume that each $\boldsymbol{x}_j$ is of unit norm.

Using the same notation as for (3.162), the proposed EnSC computes $\boldsymbol{c}^*(\boldsymbol{x}_j, \mathbf{X}_{-j})$ for each $\{\boldsymbol{x}_j\}_{j=1}^{N}$, i.e.,

$$\boldsymbol{c}^*(\boldsymbol{x}_j, \mathbf{X}_{-j}) = \arg\min_{\boldsymbol{c}} f_{EN}(\boldsymbol{c}; \boldsymbol{x}_j, \mathbf{X}_{-j}), \tag{3.172}$$

where $\mathbf{X}_{-j}$ is $\mathbf{X}$ with the j-th column removed. In this section, we focus on a given vector, say $\boldsymbol{x}_j$. We suppose that $\boldsymbol{x}_j \in \mathcal{S}_\ell$ for some ℓ, and use $\mathbf{X}_{-j}^{\ell}$ to denote the submatrix of $\mathbf{X}$ with columns from $\mathcal{S}_\ell$ except that $\boldsymbol{x}_j$ is removed. Since our goal is to use the entries of $\boldsymbol{c}^*(\boldsymbol{x}_j, \mathbf{X}_{-j})$ to construct an affinity graph in

which only points in the same subspace are connected, we desire the nonzero entries of $\boldsymbol{c}^*(\boldsymbol{x}_j, \mathbf{X}_{-j})$ to be a subset of the columns $\mathbf{X}_{-j}^{\ell}$ so that no connections are built between points from different subspaces. If this is the case, we say that such a solution $\boldsymbol{c}^*(\boldsymbol{x}_j, \mathbf{X}_{-j})$ is *subspace-preserving*. On the other hand, we also want the nonzero entries of $\boldsymbol{c}^*(\boldsymbol{x}_j, \mathbf{X}_{-j})$ to be as dense as possible in $\mathbf{X}_{-j}^{\ell}$ so that within each cluster the affinity graph is well-connected[2]. To some extent, these are conflicting goals: if the connections are few, it is more likely that the solution is subspace-preserving, but the affinity graph of each cluster is not well connected. Conversely, as one builds more connections, it is more likely that some of them will be false, but the connectivity is improved. In the following, we give a geometric interpretation of the tradeoff between the subspace preserving and connectedness properties.

Our analysis is built upon the optimization problem $\min_{\boldsymbol{c}} f_{EN}(\boldsymbol{c}; \boldsymbol{x}_j, \mathbf{X}_{-j}^{\ell})$. Note that its solution is trivially subspace preserving since the dictionary $\mathbf{X}_{-j}^{\ell}$ is contained in $\mathcal{S}_\ell$. We then treat all points from other subspaces as newly added columns to $\mathbf{X}_{-j}^{\ell}$ and apply Propositions 1 and 2. We get the following geometric result.

Lemma 15. *Suppose that $\boldsymbol{x}_j \in \mathcal{S}_\ell$. Then, the vector $\boldsymbol{c}^*(\boldsymbol{x}_j, \mathbf{X}_{-j})$ is subspace preserving if and only if $\boldsymbol{x}_k \notin \boldsymbol{\Delta}(\boldsymbol{x}_j, \mathbf{X}_{-j}^{\ell})$ for all $\boldsymbol{x}_k \notin \mathcal{S}_\ell$.*

[2]In fact, even when each cluster is well-connected, further improving connectivity within clusters is still beneficial since it enhances the ability of the subsequent step of spectral clustering in correcting any erroneous connections in the affinity graph [164, 166].

Proof. Consider the problem

$$\boldsymbol{c}^*(\boldsymbol{x}_j, \mathbf{X}_{-j}^\ell) = \arg\min_{\boldsymbol{c}} f_{EN}(\boldsymbol{c}; \boldsymbol{x}_j, \mathbf{X}_{-j}^\ell) \tag{3.173}$$

and by our notation, let $\boldsymbol{\Delta}(\boldsymbol{x}_j, \mathbf{X}_{-j}^\ell)$ be its oracle region.

For the "if" part, we know from Proposition 1 that adding more points that are outside of the oracle region $\boldsymbol{\Delta}(\boldsymbol{x}_j, \mathbf{X}_{-j}^\ell)$ to the dictionary of (3.173) does not affect its solution. To be more specific, if it holds that $\boldsymbol{x}_k \notin \boldsymbol{\Delta}(\boldsymbol{x}_j, \mathbf{X}_{-j}^\ell)$ for all $\boldsymbol{x}_k \notin \mathcal{S}_\ell$, then by Proposition 1 we have $\boldsymbol{c}^*(\boldsymbol{b}, \mathbf{X}_{-j}) = P \cdot [\boldsymbol{c}^*(\boldsymbol{x}_j, \mathbf{X}_{-j}^\ell)^\top, \mathbf{0}^\top]^\top$, where P is some permutation matrix.

For the "only if" part, if any $\boldsymbol{x}_k \notin \mathcal{S}_\ell$ is in the oracle region $\boldsymbol{\Delta}(\boldsymbol{x}_j, \mathbf{X}_{-j}^\ell)$, then Proposition 2 shows that the coefficient vector of $\boldsymbol{c}^*(\boldsymbol{b}, \mathbf{X}_{-j})$ that corresponds to points outside of $\mathcal{S}_\ell$ is nonzero. Therefore, the solution is not correct in identifying the l-th subspace. □

We illustrate the geometry implied by Lemma 15 in Figure 3.8, where we assume $\mathcal{S}_\ell$ is a two dimensional subspace in $\mathbb{R}^3$. The dictionary $\mathbf{X}_{-j}^\ell$ is represented by the blue dots in the plane and the oracle region $\boldsymbol{\Delta}(\boldsymbol{x}_j, \mathbf{X}_{-j}^\ell)$ is denoted as the two red circles. The green dots are all other points in the dictionary. Lemma 15 says that $\boldsymbol{c}^*(\boldsymbol{x}_j, \mathbf{X}_{-j})$ is subspace preserving if and only if all green dots lie outside of the red region.

To ensure that a solution is subspace preserving one desires a small oracle

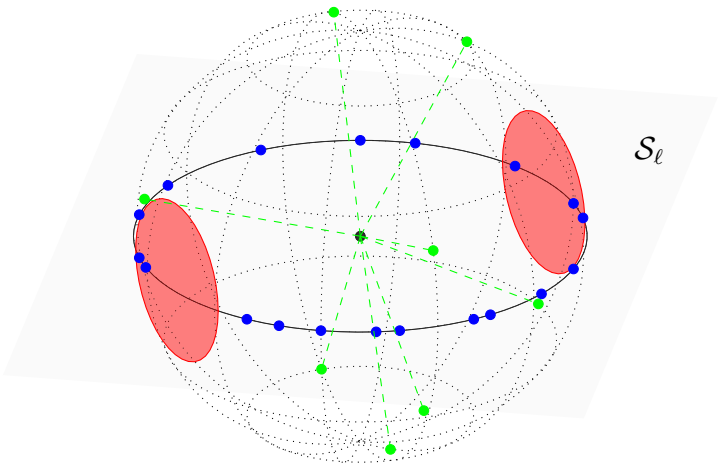


Figure 3.8: The structure of the solution for an example in $\mathbb{R}^3$ associated with a point $\boldsymbol{x}_j$ (not shown) that lies in the 2-dimensional subspace $\mathcal{S}_\ell$. The blue dots illustrate the columns of $\mathbf{X}^\ell_{-j}$, the union of the two red regions is the oracle region $\boldsymbol{\Delta}(\boldsymbol{x}_j, \mathbf{X}^\ell_{-j})$, and the green points are vectors from other subspaces.

region, while to ensure connectedness one desires a large oracle region. These facts again highlight the trade-off between these two properties. Recall that the elastic net balances ℓ_1 regularization (promotes sparse solutions) and ℓ_2 regularization (promotes dense solutions). Thus, one should expect that the oracle region will decrease in size as λ is increased from 0 towards 1. Theorem 37 formalizes this claim. To understand it, we comment that the size of the oracle region $\boldsymbol{\Delta}(\boldsymbol{x}_j, \mathbf{X}^\ell_{-j})$ is controlled by the quantity $\lambda / \|\boldsymbol{\delta}(\boldsymbol{x}_j, \mathbf{X}^\ell_{-j})\|_2$ as depicted in Figure 3.7.

Theorem 37. *If $\boldsymbol{x}_j \in \mathcal{S}_\ell$, then*

$$\frac{\lambda}{\|\boldsymbol{\delta}(\boldsymbol{x}_j, \mathbf{X}^\ell_{-j})\|_2} \geq \frac{r_j^2}{r_j + \frac{1-\lambda}{\lambda}}, \tag{3.174}$$

where r_j is the inradius of the convex hull of the symmetrized points in $\mathbf{X}^{\ell}_{-j}$, i.e.,

$$r_j := r(\text{conv}\{\pm \boldsymbol{x}_k : \boldsymbol{x}_k \in \mathcal{S}_\ell \textit{ and } k \neq j\}). \tag{3.175}$$

We define the right-hand-side of (3.174) *to be zero when $\lambda = 0$.*

The above theorem allows us to determine an upper bound for the size of the oracle region. This follows since a lower bound on the size of $\lambda/\|\boldsymbol{\delta}(\boldsymbol{x}_j, \mathbf{X}^{\ell}_{-j})\|_2$ implies an upper bound on the size of the oracle region (see (3.168) and Figure 3.7). Also notice that the right hand side of (3.174) is in the range $[0, r_j)$ and is monotonically increasing with λ. Thus, it provides an upper bound on the area of the oracle region, which decreases as λ increases. This highlights that the trade-off between the subspace-preserving and connectedness properties is controlled by λ.

Remark. It would be nice if $\lambda/\|\boldsymbol{\delta}(\boldsymbol{x}_j, \mathbf{X}^{\ell}_{-j})\|_2$ was increasing as a function of λ (we already know that its lower bound given in Theorem 37 is increasing in λ). However, one can show using the data $\boldsymbol{x}_j = [0.22, 0.72, 0.66]^\top$,

$$\mathbf{X}^{\ell}_{-j} = \begin{bmatrix} -0.55 & -0.82 & -0.05 & 0.22 \\ 0.22 & 0.57 & 0.84 & 0.78 \\ -0.80 & 0.00 & 0.55 & 0.58 \end{bmatrix}, \tag{3.176}$$

and parameter choice $\gamma = 10$, that $\lambda/\|\boldsymbol{\delta}\|$ (with $\lambda = 0.88$) is larger than $\lambda/\|\boldsymbol{\delta}\|$

(with $\lambda = 0.95$).

To prove Theorem 37, we first state the next lemma which can be interpreted as giving an equivalent definition of inradius for certain convex sets.

Lemma 16. *If $\{\boldsymbol{a}_j\}_{j=1}^N$ are points with unit ℓ_2 norm, then*

$$r\big(\operatorname{conv}\{\pm\boldsymbol{a}_j\}_{j=1}^N\big) = \min_{\boldsymbol{v}\neq 0} \max_{j=1,\cdots,N} \mu(\boldsymbol{a}_j, \boldsymbol{v}). \tag{3.177}$$

The proof follows directly from Theorem 9. The interpretation of Lemma 16 is as follows: one searches for a vector $\boldsymbol{v}$ that is furthest away from all points $\{\pm\boldsymbol{a}_j\}_{j=1}^N$, and the inradius is the coherence of this $\boldsymbol{v}$ with the closest neighbor in $\{\boldsymbol{a}_j\}_{j=1}^N$. In other words, it characterizes the covering property of the points $\{\pm\boldsymbol{a}_j\}_{j=1}^N$. If inradius is large then for any point in the space there exists an $\boldsymbol{a}_j$ that is close to it.

Result Theorem 37 follows from the bound on the norm of the oracle point given below in Lemma 17 and the relation $\kappa \geq r$ as revealed by Lemma 16.

Lemma 17. *Consider problem* (3.162). *If we define $\kappa = \max_j \mu(\boldsymbol{a}_j, \boldsymbol{\delta})$ as the coherence between the oracle point $\boldsymbol{\delta}$ and its closest neighbor among the columns of A, then*

$$\|\boldsymbol{\delta}\|_2 \leq \frac{\lambda\kappa + 1 - \lambda}{\kappa^2}. \tag{3.178}$$

Proof. If $\boldsymbol{c}^* = 0$, then the optimality condition (3.164) shows that $\|A^\top\boldsymbol{\delta}\|_\infty \leq \lambda$,

hence $\kappa\|\boldsymbol{\delta}\|_2 \leq \lambda$. From this it is easy to see that (3.178) holds.

Next, we suppose that $\boldsymbol{c}^* \neq 0$, and assume without loss of generality that every entry in $\boldsymbol{c}^*$ is positive. (If an entry of $\boldsymbol{c}^*$ is zero then we can remove the corresponding column from A without affecting the quantities $\boldsymbol{\delta}$ and κ. Also, if $\boldsymbol{c}^*_j < 0$ for some j, we can change $\boldsymbol{a}_j$ to $-\boldsymbol{a}_j$ so that the solution will simply have $\boldsymbol{c}^*_j$ changed to $-\boldsymbol{c}^*_j$, which is then positive.) Since all entries of $\boldsymbol{c}^*$ are positive, we may conclude that $\boldsymbol{a}_j^\top \boldsymbol{\delta} > \lambda$ for all j.

We now multiply both sides of the optimality condition (3.164) by $\boldsymbol{c}^{*\top}$ to obtain

$$\langle \boldsymbol{c}^*, A^\top \boldsymbol{\delta} \rangle = (1-\lambda)\|\boldsymbol{c}^*\|_2^2 + \lambda\|\boldsymbol{c}^*\|_1. \tag{3.179}$$

Also, by the definition of the oracle point, we have

$$\langle A\boldsymbol{c}^*, \boldsymbol{\delta} \rangle = \langle \boldsymbol{b} - \boldsymbol{\delta}/\gamma, \boldsymbol{\delta} \rangle = \langle \boldsymbol{b}, \boldsymbol{\delta} \rangle - \|\boldsymbol{\delta}\|_2^2/\gamma. \tag{3.180}$$

Notice that since the left-hand-side of (3.179) and (3.180) are the same, we can equate the right-hand-sides to get

$$(1-\lambda)\|\boldsymbol{c}^*\|_2^2 + \lambda\|\boldsymbol{c}^*\|_1 = \langle \boldsymbol{b}, \boldsymbol{\delta} \rangle - \|\boldsymbol{\delta}\|_2^2/\gamma \leq \|\boldsymbol{\delta}\|_2 - \|\boldsymbol{\delta}\|_2^2/\gamma.$$

We now prove a lower bound on the left-hand-side of (3.181) in terms of $\|\boldsymbol{\delta}\|_2$.

From (3.166) and $\boldsymbol{a}_j^\top \boldsymbol{\delta} > \lambda$ for all j, we have

$$\begin{aligned}(1-\lambda)\|\boldsymbol{c}^*\|_2^2 + \lambda\|\boldsymbol{c}^*\|_1 &\geq (1-\lambda)c_j^2 + \lambda c_j \\ &= \frac{\mathcal{T}_\lambda(\boldsymbol{a}_j^\top \boldsymbol{\delta})^2}{1-\lambda} + \frac{\lambda \mathcal{T}_\lambda(\boldsymbol{a}_j^\top \boldsymbol{\delta})}{1-\lambda} = \frac{(\boldsymbol{a}_j^\top \boldsymbol{\delta} - \lambda)\cdot \boldsymbol{a}_j^\top \boldsymbol{\delta}}{1-\lambda} \end{aligned} \tag{3.181}$$

for all $1 \leq j \leq N$. If we now take j to be the index that maximizes $\langle \boldsymbol{a}_j, \boldsymbol{\delta}/\|\boldsymbol{\delta}\|_2 \rangle$ and use the definition of κ, then

$$(1-\lambda)\|\boldsymbol{c}^*\|_2^2 + \lambda\|\boldsymbol{c}^*\|_1 \geq \frac{(\kappa\|\boldsymbol{\delta}\|_2 - \lambda)\cdot \kappa\|\boldsymbol{\delta}\|_2}{1-\lambda}. \tag{3.182}$$

Combining (3.181) with (3.182), we get an inequality on $\|\boldsymbol{\delta}\|_2$:

$$\frac{(\kappa\|\boldsymbol{\delta}\|_2 - \lambda)\cdot \kappa\|\boldsymbol{\delta}\|_2}{1-\lambda} \leq \|\boldsymbol{\delta}\|_2 - \|\boldsymbol{\delta}\|_2^2/\gamma. \tag{3.183}$$

This inequality gives a bound on $\|\boldsymbol{\delta}\|_2$ of

$$\|\boldsymbol{\delta}\|_2 \leq \frac{\lambda\kappa + 1 - \lambda}{\kappa^2 + (1-\lambda)/\gamma} \leq \frac{\lambda\kappa + 1 - \lambda}{\kappa^2}, \tag{3.184}$$

which completes the proof. □

3.5.3.3 Conditions for a subspace-preserving solution

A sufficient condition for a solution to be subspace preserving is obtained by combining the geometry in Lemma 15 with the bound on the size of the oracle region implied by Theorem 37.

Theorem 38. *Let $\boldsymbol{x}_j \in \mathcal{S}_\ell$, $\boldsymbol{\delta}_j = \boldsymbol{\delta}(\boldsymbol{x}_j, \mathbf{X}_{-j}^\ell)$ be the oracle point, and r_j be the inradius characterization of $\mathbf{X}_{-j}^\ell$ as given by* (3.175)*. Then, $\boldsymbol{c}^*(\boldsymbol{x}_j, \mathbf{X}_{-j})$ is subspace preserving if*

$$\max_{k:\boldsymbol{x}_k \notin \mathcal{S}_\ell} \mu(\boldsymbol{x}_k, \boldsymbol{\delta}_j) \leq \frac{r_j^2}{r_j + \frac{1-\lambda}{\lambda}}. \tag{3.185}$$

Theorem 38 follows from Theorem 39 and the fact that $\kappa_j \geq r_j$ as revealed by Lemma 16. Notice that in Theorem 38 the quantity $\boldsymbol{\delta}_j$ is determined from $\mathbf{X}_{-j}^\ell$ and that it lies within the subspace $\mathcal{S}_\ell$ by definition of $\boldsymbol{\delta}(\boldsymbol{x}_j, \mathbf{X}_{-j}^\ell)$. Thus the left-hand-side of (3.185) characterizes the separation between the oracle point—which is in $\mathcal{S}_\ell$—and the set of points outside of $\mathcal{S}_\ell$. On the right-hand-side, r_j characterizes the distribution of points in $\mathbf{X}_{-j}^\ell$. In particular, r_j is large when points are well spread within $\mathcal{S}_\ell$ and not skewed toward any direction. Finally, note that the right-hand-side of (3.185) is an increasing function of λ, showing that the solution is more likely to be subspace preserving if more weight is placed on the ℓ_1 regularizer relative to the ℓ_2 regularizer.

Theorem 38 has a close relationship to the sufficient condition for SSC to give a subspace preserving solution (the case $\lambda = 1$) [139]. Specifically, [139]

shows that if $\max_{k:\boldsymbol{x}_k \notin \mathcal{S}_\ell} \mu(\boldsymbol{x}_k, \boldsymbol{\delta}_j) < r_j$, then SSC gives a subspace preserving solution. We can observe that condition (3.185) approaches the condition for SSC as $\lambda \to 1$.

The result stated in Theorem 38 is a special case of the following more general result.

Theorem 39. *Suppose $\boldsymbol{x}_j \in \mathcal{S}_\ell$. Let $\boldsymbol{\delta}_j = \boldsymbol{\delta}(\boldsymbol{x}_j, \mathbf{X}^\ell_{-j})$ be the oracle point, and let $\kappa_j = \max_{k \neq j, \boldsymbol{x}_k \in \mathcal{S}_\ell} \mu(\boldsymbol{x}_k, \boldsymbol{\delta}_j)$ be the coherence of $\boldsymbol{\delta}_j$ with its nearest neighbor in $\mathbf{X}^\ell_{-j}$. Then, the solution $\boldsymbol{c}^*(\boldsymbol{x}_j, \mathbf{X}_{-j})$ is subspace preserving if*

$$\max_{k:\boldsymbol{x}_k \notin \mathcal{S}_\ell} \mu(\boldsymbol{x}_k, \boldsymbol{\delta}_j) \leq \frac{\kappa_j^2}{\kappa_j + \frac{1-\lambda}{\lambda}}. \tag{3.186}$$

Theorem 39 can be obtained by combining Lemma 15 and Theorem 37. The only difference between this result and that in Theorem 38 is that κ_j is used instead of r_j for characterizing the distribution of points in $\mathbf{X}^\ell_{-j}$. We show in Lemma 16 that $r_j \leq \kappa_j$, which makes Theorem 39 more general than Theorem 38. Geometrically, r_j is large if the subspace $\mathcal{S}_\ell$ is well-covered by $\mathbf{X}^\ell_j$, while κ_j is large if the neighborhood of the oracle closest to $\boldsymbol{\delta}_j$ is well-covered, i.e., there is a point in $\mathbf{X}^\ell_{-j}$ that is close to $\boldsymbol{\delta}_j$. Thus, while the condition in Theorem 38 requires each subspace to have global coverage by the data, the condition in Theorem 39 allows the data to be biased, and only requires a local region to be well-covered. In addition, condition (3.186) can be checked when the

membership of the data points is known. This advantage allows us to check the tightness of the condition (3.186), which is studied in more details in the appendix. In contrast, condition (3.185) and previous work on SSC [139, 170] use the inradius r_j, which is generally NP-hard to calculate [139, 172].

3.5.3.4 Discussion for the Case $\lambda = 1$

As the analyses and results of the previous sections are for $\lambda \in [0, 1)$, in this section we discuss the case $\lambda = 1$. It turns out that the geometric structure of the elastic net solution for $\lambda = 1$ is slightly different. As a result, many of the theorems and discussions do not apply for $\lambda = 1$, so that we need a separate discussion for most of the results.

The oracle point and oracle region. We use the same definitions of the oracle point and oracle region as before. While for $\lambda \in [0, 1)$ the oracle point $\boldsymbol{\delta}$ is unique since $\boldsymbol{c}^*$ is unique due to the strong convexity of the problem, the same argument does not apply to the case $\lambda = 1$. However, we can sill establish the uniqueness of the oracle point.

Theorem 40. *The oracle point $\boldsymbol{\delta}(\boldsymbol{b}, A)$ is unique for each choice of $\lambda \in [0, 1]$.*

Proof. For $\lambda < 1$, the optimization problem (3.162) is strongly convex, thus $\boldsymbol{c}^*$ is unique. Then, by (3.165), $\boldsymbol{\delta}(\boldsymbol{b}, A)$ is unique.

For $\lambda = 1$, we rewrite problem (3.162) equivalently as

$$\min_{\boldsymbol{c},\boldsymbol{e}} \|\boldsymbol{c}\|_1 + \frac{\gamma}{2}\|\boldsymbol{e}\|_2^2 \quad \text{s.t. } \boldsymbol{b} = A\boldsymbol{c} + \boldsymbol{e}. \tag{3.187}$$

Introducing the dual vector $\boldsymbol{v}$, the Lagragian function is

$$L(\boldsymbol{c}, \boldsymbol{e}, \boldsymbol{v}) = \|\boldsymbol{c}\|_1 + \frac{\gamma}{2}\|\boldsymbol{e}\|_2^2 + \langle \boldsymbol{v}, \boldsymbol{b} - A\boldsymbol{c} - \boldsymbol{e}\rangle, \tag{3.188}$$

and the corresponding dual problem is

$$\max_{\boldsymbol{v}} \langle \boldsymbol{b}, \boldsymbol{v}\rangle - \frac{1}{2\gamma}\boldsymbol{v}^\top \boldsymbol{v} \quad \text{s.t. } \|A^\top \boldsymbol{v}\|_\infty \leq 1, \tag{3.189}$$

whose objective function is strongly concave with a unique solution $\boldsymbol{v}^*$. Also, from the optimality conditions we have $\boldsymbol{v}^* = \gamma \boldsymbol{e}^* = \gamma(\boldsymbol{b} - A\boldsymbol{c}^*(\boldsymbol{b}, A)) = \boldsymbol{\delta}(\boldsymbol{b}, A)$, so that $\boldsymbol{\delta}(\boldsymbol{b}, A)$ is unique. □

The geometric structure of the solution. Recall that from Theorem 36 we know that the oracle region contains points whose corresponding coefficients are nonzero, i.e., $\boldsymbol{c}_j^* \neq 0$ if and only if $\boldsymbol{a}_j \in \boldsymbol{\Delta}(\boldsymbol{b}, A)$. For the case $\lambda = 1$, this argument no longer holds. Actually, Theorem 36 still holds for $\lambda = 1$, but the left-hand-side of (3.163) becomes zero, and it means that no column of A is in the oracle region $\boldsymbol{\Delta}(\boldsymbol{b}, A)$. To further understand the structure of the solution, we need the following result.

Theorem 41. *The solution $\boldsymbol{c}^* = \boldsymbol{c}^*(\boldsymbol{b}, A)$ to problem* (3.162) *with $\lambda = 1$ satisfies that if $\boldsymbol{c}_j^* \neq 0$, then $|\boldsymbol{a}_j^\top \boldsymbol{\delta}| = 1$.*

This result follows from the optimality condition. It means that a coefficient $\boldsymbol{c}_j^*$ is nonzero only if $\boldsymbol{a}_j$ is on the boundary of the oracle region $\boldsymbol{\Delta}(\boldsymbol{b}, A)$, which we denote as $\partial\boldsymbol{\Delta}(\boldsymbol{b}, A)$. The opposite is generally not true: if $\boldsymbol{a}_j \in \partial\boldsymbol{\Delta}(\boldsymbol{b}, A)$, it does not necessarily mean that $\boldsymbol{c}_j^* \neq 0$.

The geometric structure of the solution is thus clear: all columns of A are outside the oracle region, but some columns of A are in $\partial\boldsymbol{\Delta}(\boldsymbol{b}, A)$ with some of these corresponding to nonzero coefficients.

Correctness of EnSC. Theorem 39 gives a sufficient condition for guaranteeing the correctness of EnSC when $\lambda \in [0, 1)$. In extending the result to the case $\lambda = 1$ we need a slightly stronger condition.

Theorem 42. *Let $\boldsymbol{x}_j \in \mathcal{S}_\ell$, and $\boldsymbol{\delta}_j$ and κ_j be defined as in Theorem 39. Then, for all $\lambda \in [0, 1]$, the solution $\boldsymbol{c}^*(\boldsymbol{x}_j, \mathbf{X}_{-j})$ is correct in identifying the subspace $\mathcal{S}_\ell$ if*

$$\max_{k:\boldsymbol{x}_k \notin \mathcal{S}_\ell} \mu(\boldsymbol{x}_k, \boldsymbol{\delta}_j) < \frac{\kappa_j^2}{\kappa_j + \frac{1-\lambda}{\lambda}}. \tag{3.190}$$

The difference between (3.190) and (3.186) is that the inequality is strict in (3.190). This modification is necessary to handle the case $\lambda = 1$, for the condition (3.186) does not exclude the case that $\boldsymbol{x}_k \notin \mathcal{S}_\ell$ may lie on the boundary of $\boldsymbol{\Delta}(\boldsymbol{x}_j, \mathbf{X}_{-j}^\ell)$ and yet correspond to a nonzero coefficient.

Finally, we discuss the implication of Theorem 42 in the context of SSC. When $\lambda = 1$, condition (3.190) simplifies to

$$\max_{k:\boldsymbol{x}_k \notin \mathcal{S}_\ell} \mu(\boldsymbol{x}_k, \boldsymbol{\delta}_j) < \kappa_j. \tag{3.191}$$

In [139] a sufficient condition for SSC is given by

$$\max_{k:\boldsymbol{x}_k \notin \mathcal{S}_\ell} \mu(\boldsymbol{x}_k, \boldsymbol{\delta}_j) < r_j. \tag{3.192}$$

Using the relationship $r_j \leq \kappa_j$, our condition in (3.191) is a weaker requirement than that in the previous work. Specifically, condition (3.192) requires that the entire subspace $\mathcal{S}_\ell$ be well-covered by the columns of $\mathbf{X}_{-j}^{\ell}$ so that r_j is large. In contrast, our condition in (3.191) only requires the neighborhood of the oracle point $\boldsymbol{\delta}_j$ to be well-covered, i.e., that there exists a column in $\mathbf{X}_{-j}^{\ell}$ that is close to $\boldsymbol{\delta}_j$. Another advantage of our condition (3.191) is that it can be verified when the ground truth is known. In contrast, the condition in (3.192) cannot be verified since the computation of r_j is generally NP-hard [139].

Chapter 4

Subspace Clustering with Large-scale Data

From the previous chapter, we have seen that under certain conditions, sparse methods can provably produce subspace-preserving solutions in the subspace classification and subspace clustering applications, therefore validating the SRC, SSC and EnSC methods for such tasks. In particular, the basis pursuit (BP) based methods (i.e., SRC-BP and SSC-BP) and the elastic net based method (i.e., EnSC) compute the sparse representations as the solution set to their corresponding convex optimization problems. From a practical perspective, it is important to develop numerical algorithms that can effectively solve such optimization problems.

Subspace clustering is particularly demanding for very efficient algorithms

for solving the BP and the elastic net problems, as the SSC and EnSC methods involve solving N such optimization problems where N is the number of data points. If N is very large, say more than $10{,}000$, then existing solvers (e.g., the alternating direction method of multipliers (ADMM) that is used in [60], the accelerated proximal gradient (APG) [19] that is used in [62] and the linearized alternating direction method (LADM) [104] that is used in [128]) take a very long time (see Figure 1.7 for an illustration). Moreover, these algorithms require computing and storing the full data Gramian matrix, therefore they have quadratic memory complexity. All these drawbacks of previous solvers call for the development of new algorithms that are not only efficient but also able to handle large-scale data.

In this chapter, we propose to use an *active support* strategy for solving the optimization problems in SSC and EnSC. The active support method is motivated by the observation that the target solution in a sparse recovery problem is sparse, therefore as long as we can correctly estimate the support of the optimal solution then the problem is reduced from a large-scale optimization problem to a small-scale optimization problem that can be solved much more efficiently. The key challenge in the active support approach lies in how to effectively find the optimal support set.

We start by showing that the OMP algorithm for sparse recovery is one particular instance of the active support approach, where the optimal support

is estimated iteratively via a greedy search procedure. As we have shown in the previous chapter, the sparse solutions computed from OMP are guaranteed to be subspace-preserving. Nonetheless, the update of the active support in OMP uses a heuristics, and is not guaranteed to find optimal solutions from the optimization perspective. This is likely to the underlying reason that SSC-OMP has relatively inferior clustering performance relative to SSC-BP and EnSC, as we will see in the experiments in Section 4.4.1.

Due to this drawback of OMP, we further develop an active support based algorithm for solving the basis pursuit and elastic-net problems, which are used in SSC-BP and EnSC, respectively. Our proposed algorithm exploits the fact that the optimal support of the elastic-net solution fall into an *oracle region*, which we use to define and efficiently update an active support. The proposed update rule leads to an iterative algorithm that is shown to converge to the optimal solution in a finite number of iterations.

Although active support methods are very efficient, they still have a computational complexity of $\mathcal{O}(N^2)$ and thus cannot deal with datasets containing 1 million points or more. Therefore, we further develop a divide-and-conquer based algorithm that is able to cluster 1 million data points in a reasonable amount of time. In this approach, the original data is split into chunks of moderate size so that points in each chunk can be efficiently clustered using SSC or EnSC. The clusters from different chunks that correspond to the same

subspace are then merged to obtain a complete clustering of the original data.

4.1 Prior art in scalable subspace clustering

Several scalable subspace clustering methods have been studied in the past few years, including methods based on truncated SVD [104, 181], factorization [137] and subsampling [2, 132, 147]. The works of [104] and [181] present fast algorithms for low rank representation (LRR) [107], a subspace clustering method that finds a subspace-preserving representation by learning a matrix of coefficients that is of low-rank. They exploit the fact that the optimal solution is of low-rank by using a truncated SVD at each iteration, which reduces the computational complexity from $O(N^3)$ to $O(N^2)$. However, since the representation matrix of LRR is non-sparse and requires $O(N^2)$ memory, LRR-based methods cannot be directly applied to a dataset of size, say, $100{,}000$ data points, as it would require ~ 80GB memory. To address the memory issue, [137] exploits the fact that the representation matrix in LRR is low rank and uses a factored form of the representation matrix to save memory. However, there are no theoretical guarantees that the method in [137] will give the correct clustering.

Subsampling-based methods have also been proposed to help scale existing

methods. [132] presented the Scalable SSC method, in which SSC is applied to a subset of the data drawn at random from the entire dataset. Once the subsampled data is clustered, the remaining data points are classified to one of the computed clusters. While this method is computationally efficient, its clustering accuracy becomes sensitive to the subsampled data, i.e., it requires the subsampled data to well represent the distribution of points in all subspaces. The work of [2] learns a small-sized dictionary and clusters the data based on the affinity between the data points and the dictionary atoms. While the learned dictionary would be expected to be more representative of the data than the random subsampled data used by Scalable SSC, there are no theoretical guarantees on the quality of the dictionary for the purpose of clustering. Indeed, the clustering accuracy is reduced for many cases in the empirical evaluation in [2]. Instead of learning a dictionary through dictionary learning techniques, a very recent work [147] uses a sketched dictionary that is simply generated by taking random linear combination of the data points.

4.2 Active support methods

A key step in SSC-BP, SSC-OMP and EnSC is the computation of a sparse representation, where the goal is to find a sparse vector $\boldsymbol{c}_0 \in \mathbb{R}^N$ that satisfies $\boldsymbol{b} = \mathbf{A}\boldsymbol{c}_0$ for some vector $\boldsymbol{b} \in \mathbb{R}^D$ and dictionary matrix $\mathbf{A} \in \mathbb{R}^{D \times N}$. Specifi-

cally, recall that OMP aims to solve the following optimization problem (see Eq. (2.4)):

$$\min_{\boldsymbol{c}} \|\boldsymbol{b} - \mathbf{A}\boldsymbol{c}\|_2 \quad \text{s.t. } \|\boldsymbol{c}\|_0 \leq k_{\max}, \tag{4.1}$$

where $k_{\max}$ is the sparsity of the target solution. BP is a convex relaxation based sparse recovery method which aims to solve the following problem

$$\min_{\boldsymbol{c}} \|\boldsymbol{c}\|_1 \quad \text{s.t. } \boldsymbol{b} = \mathbf{A}\boldsymbol{c}. \tag{4.2}$$

In practical applications, the data is usually corrupted by noise, therefore it makes more sense to use a penalty term $\|\mathbf{A}\boldsymbol{c} - \boldsymbol{b}\|_2^2$ in lieu of the constraint $\boldsymbol{b} = \mathbf{A}\boldsymbol{c}$ and solve the following optimization problem

$$\min_{\boldsymbol{c}} \|\boldsymbol{c}\|_1 + \frac{\gamma}{2}\|\boldsymbol{b} - \mathbf{A}\boldsymbol{c}\|_2^2, \tag{4.3}$$

where γ is a trade-off parameter that balances the sparse regularization $\|\boldsymbol{c}\|_1$ and the representation residual $\|\mathbf{A}\boldsymbol{c} - \boldsymbol{b}\|_2^2$. The BP optimization problem in (4.3) is a special case of the elastic net problem which is the following:

$$\min_{\boldsymbol{c}} \lambda\|\boldsymbol{c}\|_1 + \frac{1-\lambda}{2}\|\boldsymbol{c}\|_2^2 + \frac{\gamma}{2}\|\boldsymbol{b} - \mathbf{A}\boldsymbol{c}\|_2^2, \tag{4.4}$$

where λ and γ are trade-off parameters.

Solving the optimization problems (4.1), (4.3) and (4.4) can be difficult when

the dictionary $\mathbf{A}$ has a large number of columns. In this section, we introduce an active support strategy to handle such cases. The method is based on exploiting the fact that the optimal solutions to (4.1), (4.3) and (4.4) are expected to be sparse. This has the implication that the solution is determined by a subset of the columns of $\mathbf{A}$ that corresponds to the support of the true solution, and the size of this subset is typically much smaller than the total number of columns in $\mathbf{A}$. Motivated by this observation, we propose an iterative procedure for finding such an optimal support, which is achieved by solving a sequence of problems on small subsets of data. A meta-algorithm that illustrates the idea of this algorithm is presented in Algorithm 4.

Algorithm 4 Active support meta-algorithm for sparse recovery

Input: $\mathbf{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_N] \in \mathbb{R}^{D \times N}$, $\boldsymbol{b} \in \mathbb{R}^D$, and model parameters

1: Initialize the active support $\mathcal{W}^{(0)} \subseteq \{1, \cdots, N\}$ and set $k = 0$.

2: **loop**

3: Compute $\boldsymbol{c}^{(k)} \in \mathbb{R}^N$ as the solution to (4.1), (4.3) or (4.4) with the additional constraint that $\operatorname{supp}(\boldsymbol{c}^{(k)}) \subseteq \mathcal{W}^{(k)}$.

4: Compute the active support $\mathcal{W}^{(k+1)}$ based on the solution $\boldsymbol{c}^{(k)}$.

5: Break if certain conditions are met; otherwise, set $k \leftarrow k + 1$.

6: **end loop**

Output: The vector $\boldsymbol{c}^{(k)}$. Its support is a subset of $\mathcal{W}^{(k)}$.

In each iteration of Algorithm 4, step 3 involves solving the original sparse

optimization problem (4.1), (4.3) or (4.4), but with the additional constraint that the entries of $\boldsymbol{c}$ that are not in the current support set $\mathcal{W}_k$ are zero. Computationally, this is equivalent to solving (4.1) or (4.4) but with the dictionary $\mathbf{A}$ replaced by a sub-dictionary that contains columns of $\mathbf{A}$ that are indexed by $\mathcal{W}_k$. Therefore, when the size of $\mathcal{W}_k$ is much smaller than N, the optimization problem is of small-scale, and can be solved efficiently by existing solvers.

The solution $\boldsymbol{c}_{\mathcal{W}^{(k)}}$ from step 3 is expected to contain important information on the optimality of the current active support $\mathcal{W}^{(k)}$. Using such information, we update the active support set in step 4, with the expectation that the iterative update ultimately converges to the optimal support. Note that the active support update step is expected to have complexity at least $\mathcal{O}(N)$, as we will need to check all columns in $\mathbf{A}$ and determine whether each of them is included in $\mathcal{W}^{(k+1)}$ or not. Fortunately, this step typically involves only basic linear operations such as the inner product of a vector with all columns of $\mathbf{A}$, which can be carried out very efficiently even for very large-scale data $\mathbf{A}$. In addition, if the number of iterations in Algorithm 4 is small, then the entire procedure only requires a limited number of visits to the entire data $\mathbf{A}$. Therefore, Algorithm 4 can be significantly faster than previous optimization algorithms.

In the following, we explain in more details how the active support method applies to solving the optimization (4.1), (4.3) and (4.4). We start by showing that the OMP algorithm for solving (4.1) naturally fits into the framework of

active support algorithm. However, the OMP has the drawback that it adds only one point to the active set per iteration, therefore the number of iterations is lower bounded by the sparsity level. Moreover, the active support in OMP is updated in a heuristic manner, so it may not lead to optimal solutions. In light of the drawbacks of OMP, we present an oracle based active support update strategy based on the notion of an oracle region that we introduced in Chapter 3 for solving the BP and elastic net problems in (4.3) and (4.4), respectively. We will see that the update rule allows for adding and removing multiple entries in each iteration, therefore is expected to be faster than OMP. Moreover, the update rule allows an optimal solution to the optimization problem to be found.

4.2.1 Greedy based SSC-OMP algorithm

To see that OMP described in Algorithm 1 is an active support method, we rewrite its procedure in Algorithm 5. It is not hard to check that Algorithm 1 and Algorithm 5 are equivalent, i.e., they produce the same output.

From Algorithm 5, it is apparent that the OMP is an instance of the active support method for solving (4.1). In particular, OMP uses a heuristic way of updating the active support, that is, it starts with an empty active support and iteratively adds the data point that is the most correlated (in terms of inner product) with the residual $\boldsymbol{v}^{(k)}$ computed from the current solution $\boldsymbol{c}^{(k)}$. We will see in the experiments that SSC-OMP is an efficient algorithm and is able to

Algorithm 5 Active support algorithm for solving (4.1) **(a.k.a., OMP)**

Input: $\mathbf{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_N] \in \mathbb{R}^{D \times N}$, $\boldsymbol{b} \in \mathbb{R}^D$, $k_{\max}$, ϵ.

1: Initialize the active support $\mathcal{W}^{(0)} = \emptyset$ and set $k = 0$

2: **loop**

3: Compute $\boldsymbol{c}^{(k)} = \underset{\boldsymbol{c} \in \mathbb{R}^N : \text{supp}(\boldsymbol{c}) \subseteq \mathcal{W}^{(k)}}{\arg\min} \|\boldsymbol{b} - \mathbf{A}\boldsymbol{c}\|_2$.

4: Compute $\mathcal{W}^{(k+1)} = \mathcal{W}^{(k)} \bigcup \{j^*\}$, where $j^* = \underset{j=1,\ldots,N}{\arg\max} |\langle \boldsymbol{a}_j, \boldsymbol{v}^{(k)} \rangle|$ and $\boldsymbol{v}^{(k)} = \boldsymbol{b} - \mathbf{A}\boldsymbol{c}^{(k)}$.

5: Break if $k = k_{\max}$ or $\|\boldsymbol{v}^{(k)}\|_2 \leq \epsilon$; otherwise, set $k \leftarrow k + 1$

6: **end loop**

Output: The vector $\boldsymbol{c}^{(k)}$. Its support is a subset of $\mathcal{W}^{(k)}$.

handle datasets with a million data points.

From a theoretical perspective, we have seen from Chapter 3 that SSC-OMP is guaranteed to produce subspace-preserving affinities, making it theoretically justified for subspace clustering purposes. Nonetheless, the active support update strategy in OMP is based on a heuristic and is not guaranteed to be optimal for solving the optimization problem (4.1). This is perhaps an underlying reason for the fact that in the correctness conditions for SSC-OMP and SSC-BP in Theorem 35, the probability of success for SSC-OMP is smaller than that of SSC-BP.

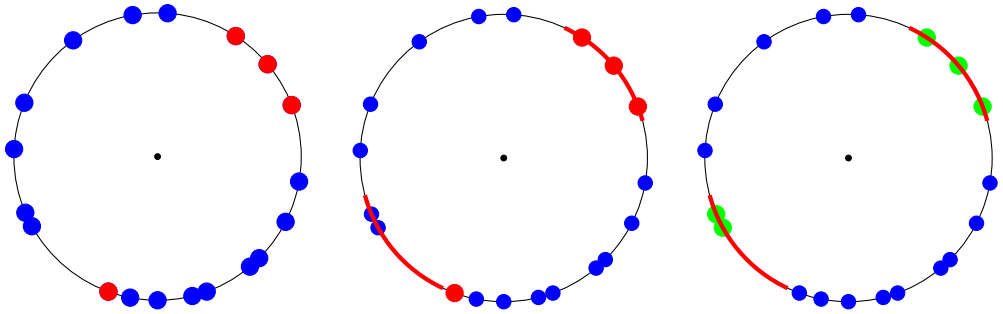
In the following, we will develop an active support method for solving the BP problem in SSC-BP. We will see that a more principled active support up-

date strategy exists in that case, which allows us to always find global optimal solutions to BP. As BP is a particular case of the elastic net problem, we start with considering the elastic net based subspace clustering (i.e. EnSC) method.

4.2.2 Oracle based EnSC algorithm

We consider solving the elastic net problem in (4.4) for λ in the range of $[0, 1)$. We will consider the case of $\lambda = 1$ (which corresponds to BP) in the next subsection.

Our active support method is summarized in Algorithm 6. The basic principle behind the algorithm is the fact that the support of the optimal solution contains points that lie in the oracle region (see Definition 21). Therefore, we propose to update the active support set from the oracle region computed on the previous active support set. Let $\mathcal{W}^{(k)}$ be the active set at iteration k. Then, the next active set $\mathcal{W}^{(k+1)}$ is selected to contain the indices of columns that are in the oracle region $\Delta(\boldsymbol{b}, \mathbf{A}_{\mathcal{W}^{(k)}})$ (see Definition 21), where $\mathbf{A}_{\mathcal{W}^{(k)}}$ denotes the submatrix of $\mathbf{A}$ with columns indexed by $\mathcal{W}^{(k)}$. We use Figure 4.1 for a conceptual illustration. In Figure 4.1a we show the columns of $\mathbf{A}$ that correspond to the active set $\mathcal{W}^{(k)}$ by labeling the corresponding columns of $\mathbf{A}_{\mathcal{W}^{(k)}}$ in red. The oracle region $\Delta(\boldsymbol{b}, \mathbf{A}_{\mathcal{W}^{(k)}})$ is the union of the red arcs in Figure 4.1b. Notice that at the bottom left there is one red dot that is not in $\Delta(\boldsymbol{b}, \mathbf{A}_{\mathcal{W}^{(k)}})$ and thus should not be included in $\mathcal{W}^{(k+1)}$, and two blue dots that are not in $\mathcal{W}^{(k)}$ but lie in the



(a) Active support $\mathcal{W}^{(k)}$ **(b)** $\Delta(\boldsymbol{b}, \mathbf{A}_{\mathcal{W}^{(k)}})$ **(c)** Active support $\mathcal{W}^{(k+1)}$

Figure 4.1: Conceptual illustration of the active support algorithm. All the dots on the unit circle illustrate the dictionary $\mathbf{A}$. (a) active set $\mathcal{W}^{(k)}$ at step k, illustrated by red dots. (b) The oracle region $\Delta(\boldsymbol{b}, \mathbf{A}_{\mathcal{W}^{(k)}})$ illustrated by red arcs. (c) The new active set $\mathcal{W}^{(k+1)}$ illustrated in green, which is the set of indices of points that are in $\Delta(\boldsymbol{b}, \mathbf{A}_{\mathcal{W}^{(k)}})$.

oracle region $\Delta(\boldsymbol{b}, \mathbf{A}_{\mathcal{W}^{(k)}})$ and thus must be included in $\mathcal{W}^{(k+1)}$. In Figure 4.1c we illustrate $\mathcal{W}^{(k+1)}$ by green dots. This iterative procedure is terminated once $\mathcal{W}^{(k+1)}$ does not contain any new points, i.e., when $\mathcal{W}^{(k+1)} \subseteq \mathcal{W}^{(k)}$, at which time $\mathcal{W}^{(k+1)}$ is the support of the optimal solution.

To facilitate the analysis of Algorithm 6, we review several notations from Chapter 3. Recall that we let

$$f_{EN}(\boldsymbol{c}; \boldsymbol{b}, \mathbf{A}) =: \lambda \|\boldsymbol{c}\|_1 + \frac{1-\lambda}{2}\|\boldsymbol{c}\|_2^2 + \frac{\gamma}{2}\|\boldsymbol{b} - \mathbf{A}\boldsymbol{c}\|_2^2 \tag{4.5}$$

and let $\boldsymbol{c}^*(\boldsymbol{b}, \mathbf{A}) := \arg\min_{\boldsymbol{c}} f_{EN}(\boldsymbol{c}; \boldsymbol{b}, \mathbf{A})$. Note that the vector $\boldsymbol{c}^{(k)}$ generated in Algorithm 6 is equal to $\boldsymbol{c}^*(\boldsymbol{b}, \mathbf{A}_{\mathcal{W}^{(k)}})$ with zeros padded to the entries corresponding to columns of $\mathbf{A}$ that are not in $\mathcal{W}^{(k)}$.

The next lemma helps explain why Algorithm 6 converges.

Algorithm 6 Active support algorithm for solving (4.4)

Input: $\mathbf{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_N] \in \mathbb{R}^{D \times N}$, $\boldsymbol{b} \in \mathbb{R}^D$, λ and γ.

1: Initialize the support set $\mathcal{W}^{(0)}$ and set $k = 0$.

2: **loop**

3: Compute $\boldsymbol{c}^{(k)} = \underset{\boldsymbol{c} \in \mathbb{R}^N : \operatorname{supp}(\boldsymbol{c}) \subseteq \mathcal{W}^{(k)}}{\arg\min} \lambda \|\boldsymbol{c}\|_1 + \frac{1-\lambda}{2}\|\boldsymbol{c}\|_2^2 + \frac{\gamma}{2}\|\boldsymbol{b} - \mathbf{A}\boldsymbol{c}\|_2^2$ using any solver.

4: Compute $\mathcal{W}^{(k+1)} \leftarrow \{j : |\langle \boldsymbol{a}_j, \boldsymbol{\delta}^{(k)} \rangle| > \lambda\}$, where $\boldsymbol{\delta}^{(k)} = \gamma \cdot (\boldsymbol{b} - \mathbf{A}\boldsymbol{c}^{(k)})$

5: Break if $\mathcal{W}^{(k+1)} \subseteq \mathcal{W}^{(k)}$; otherwise set $k \leftarrow k + 1$.

6: **end loop**

Output: The vector $\boldsymbol{c}^{(k)}$. Its support is a subset of $\mathcal{W}^{(k)}$.

Lemma 18. *In Algorithm 6, if $\mathcal{W}^{(k+1)} \not\subseteq \mathcal{W}^{(k)}$, then*

$$f_{EN}(\boldsymbol{c}^{(k+1)}; \boldsymbol{b}, \mathbf{A}) < f_{EN}(\boldsymbol{c}^{(k)}; \boldsymbol{b}, \mathbf{A}). \tag{4.6}$$

Proof. Let us define the sets

$$Q := \mathcal{W}^{(k)} \setminus \mathcal{W}^{(k+1)},$$

$$S := \mathcal{W}^{(k)} \cap \mathcal{W}^{(k+1)}, \text{ and}$$

$$R := \mathcal{W}^{(k+1)} \setminus \mathcal{W}^{(k)} \neq \emptyset,$$

where the fact that R is nonempty follows from the assumption $\mathcal{W}^{(k+1)} \subseteq \mathcal{W}^{(k)}$ in the statement of Lemma 18. By these definitions, $\mathcal{W}^{(k)} = Q \cup S$, and $\mathcal{W}^{(k+1)} =$

$S \cup R$.

By definition, $\mathcal{W}^{(k+1)}$ contains all columns of $\mathbf{A}$ that are in $\Delta(\boldsymbol{b}, \mathbf{A}_{\mathcal{W}^{(k)}})$, thus no column of $\mathbf{A}_Q$ is in $\Delta(\boldsymbol{b}, \mathbf{A}_{\mathcal{W}^{(k)}})$. By Proposition 1,

$$\boldsymbol{c}^*(\boldsymbol{b}, \mathbf{A}_{\mathcal{W}^{(k)}}) = \boldsymbol{c}^*(\boldsymbol{b}, [\mathbf{A}_S, \mathbf{A}_Q]) = \begin{bmatrix} \boldsymbol{c}^*(\boldsymbol{b}, \mathbf{A}_S) \\ \\ \mathbf{0} \end{bmatrix}, \tag{4.7}$$

in which we have assumed without loss of generality that columns of $\mathbf{A}_{\mathcal{W}^{(k)}}$ are arranged in the order such that $\mathbf{A}_{\mathcal{W}^{(k)}} = [\mathbf{A}_S, \mathbf{A}_Q]$. Using (4.7), we have

$$\begin{aligned} f_{EN}(\boldsymbol{c}^{(k)}; \boldsymbol{b}, \mathbf{A}) =& f_{EN}(\boldsymbol{c}^*(\boldsymbol{b}, \mathbf{A}_{\mathcal{W}^{(k)}});\ \boldsymbol{b}, \mathbf{A}_{\mathcal{W}^{(k)}}) = f_{EN}\left(\begin{bmatrix} \boldsymbol{c}^*(\boldsymbol{b}, \mathbf{A}_S) \\ \\ \mathbf{0} \end{bmatrix};\ \boldsymbol{b}, [\mathbf{A}_S, \mathbf{A}_R]\right) \\ &\geq \min_{\boldsymbol{c}} f_{EN}(\boldsymbol{c};\ \boldsymbol{b}, [\mathbf{A}_S, \mathbf{A}_R]) = f_{EN}(\boldsymbol{c}^*(\boldsymbol{b}, [\mathbf{A}_S, \mathbf{A}_R]);\ \boldsymbol{b}, [\mathbf{A}_S, \mathbf{A}_R]) \\ &= f_{EN}(\boldsymbol{c}^*(\boldsymbol{b}, \mathbf{A}_{\mathcal{W}^{(k+1)}});\ \boldsymbol{b}, \mathbf{A}_{\mathcal{W}^{(k+1)}}) = f_{EN}(\boldsymbol{c}^{(k+1)}; \boldsymbol{b}, \mathbf{A}). \end{aligned} \tag{4.8}$$

It remains to show that the inequality in (4.8) is strict. We show this by arguing that $[\boldsymbol{c}^*(\boldsymbol{b}, \mathbf{A}_S)^\top, \mathbf{0}^\top]^\top$ that appears on the second line of (4.8) is not an optimal solution to the optimization problem stated on the third line. Denote the solution to this optimization problem as

$$\boldsymbol{c}^*(\boldsymbol{b}, [\mathbf{A}_S, \mathbf{A}_R]) := \begin{bmatrix} \boldsymbol{c}_S \\ \\ \boldsymbol{c}_R \end{bmatrix}, \tag{4.9}$$

where $\boldsymbol{c}_S$ and $\boldsymbol{c}_R$ are of appropriate sizes. By (4.7) and the definition of the oracle region, we have

$$\Delta(\boldsymbol{b}, \mathbf{A}_S) = \Delta(\boldsymbol{b}, \mathbf{A}_{\mathcal{W}^{(k)}}). \tag{4.10}$$

Combining this with the facts that the columns of $\mathbf{A}_{\mathcal{W}^{(k+1)}}$ are in $\Delta(\boldsymbol{b}, \mathbf{A}_{\mathcal{W}^{(k)}})$ and $R \subseteq \mathcal{W}^{(k+1)}$, we know that the columns of $\mathbf{A}_R$ are in $\Delta(\boldsymbol{b}, \mathbf{A}_S)$. Consequently, by Proposition 2, we must have $\boldsymbol{c}_R \neq \mathbf{0}$. This shows that $[\boldsymbol{c}^*(\boldsymbol{b}, \mathbf{A}_S)^\top, \mathbf{0}^\top]^\top$ is not an optimal solution to the problem on the third line of (4.8) and thus the inequality in (4.8) is strict. □

The following convergence result holds for Algorithm 6.

Theorem 43. *Algorithm 6 converges to the optimal solution $\boldsymbol{c}^*(\boldsymbol{b}, \mathbf{A})$ in a finite number of iterations.*

Proof. We first prove that Algorithm 6 terminates in a finite number of iterations. Observe that the objective is strictly decreasing during each iteration before termination occurs (see Lemma 18). Since there are only finitely many different active sets, we must conclude that Algorithm 6 terminates after a finite number of iterations with $\mathcal{W}^{(k+1)} \subseteq \mathcal{W}^{(k)}$.

We now prove that when Algorithm 6 terminates, the output vector is optimal. By Theorem 36, for any $j \in \mathcal{W}_k$ it holds that $(1-\lambda) \cdot c_j^{(k)} = \mathcal{T}_\lambda(\boldsymbol{a}_j^\top \cdot \boldsymbol{\delta}^{(k)})$. For any $j \notin \mathcal{W}_k$, by the termination condition $\mathcal{W}^{(k+1)} \subseteq \mathcal{W}^{(k)}$ we know $j \notin \mathcal{W}^{(k+1)}$. Thus, by step 4, $0 = \mathcal{T}_\lambda(\boldsymbol{a}_j^\top \cdot \boldsymbol{\delta}^{(k)})$. Consequently, $\boldsymbol{c}^{(k)}$ satisfies the relation in

(3.163) and thus is the solution, i.e., $\boldsymbol{c}^{(k)} = \boldsymbol{c}^*(\boldsymbol{b}, \mathbf{A})$. □

Algorithm 6 solves large-scale problems by solving a sequence of reduced-size problems in step 3. If the active set $\mathcal{W}^{(k)}$ is small, then step 3 is a small-scale problem that can be efficiently solved. However, there is no procedure in Algorithm 6 that explicitly controls the size of $\mathcal{W}^{(k)}$. To address this concern, we propose an alternative to step 4 in which only a small number of new points—the ones most correlated with $\boldsymbol{\delta}$—are added. Specifically,

$$4\text{'} : \mathcal{W}^{(k+1)} \leftarrow \{j \in \mathcal{W}^{(k)} : |\langle \boldsymbol{a}_j, \boldsymbol{\delta}^{(k)} \rangle| > \lambda\} \cup \mathcal{V}^{(k)}, \tag{4.11}$$

where $\mathcal{V}^{(k)}$ holds the indices of the largest n entries in

$$\{|\langle \boldsymbol{a}_j, \boldsymbol{\delta}^{(k)} \rangle| : j \notin \mathcal{W}^{(k)},\ |\langle \boldsymbol{a}_j, \boldsymbol{\delta}^{(k)} \rangle| > \lambda\};$$

ideally, n should be chosen so that the size of $\mathcal{W}^{(k+1)}$ is bounded by a predetermined value $N_{\max}$ that represents the maximum size subproblem that can be handled in step 3. If $N_{\max}$ is chosen large enough that the second set in the union in (4.11) is always non-empty, then our convergence result still holds.

Initialization. We suggest the following procedure for computing the initial active set $\mathcal{W}^{(0)}$. First, compute the solution to (4.4) with $\lambda = 0$, which has a closed form solution and can be computed efficiently if the ambient dimension

D of the data is not too large. Then, the l largest entries (in absolute value) of the solution for some pre-specified value l are added to $\mathcal{W}^{(0)}$. Our experiments suggest that this strategy promotes fast convergence of Algorithm 6.

4.2.3 Oracle based SSC-BP algorithm

It is clear that the BP optimization problem (4.3) is a particular case of the elastic net optimization problem in (4.4). That is, if we set the parameter $\lambda = 1$ in (4.4), then the elastic net problem is the same as the problem in (4.3). Therefore, although the active support algorithm presented above is derived for the case of $\lambda \in [0, 1)$, it is reasonable to conjecture that (4.3) can be solved by extending Algorithm 6 to the case where $\lambda = 1$.

Surprisingly, simply applying Algorithm 6 to the case of $\lambda = 1$ does not give a valid algorithm, as the objective is no longer necessarily decreasing during each iteration (i.e., Lemma 18 is no longer true). The underlying reason for this can be seen from the geometry of the solution as we discussed in Chapter 3. For the case of $\lambda \in [0, 1)$, the optimal solution to the elastic net problem has the property that the nonzero entries correspond to columns of $\mathbf{A}$ that lie in the oracle region. This is exactly why in step 4 of Algorithm 6 the active support is chosen to include all points in the oracle region $\Delta(\boldsymbol{b}, \mathbf{A}_{\mathcal{W}^{(k)}})$. On the other hand, if $\lambda = 1$ then the geometry of the solution is different: the nonzero entries in the optimal solution correspond to columns of $\mathbf{A}$ that lie on the *boundary* of the

oracle region. In addition, there is no point from A that lies in the oracle region of the optimal solution. To add even more complications, it is not necessarily the case that all points on the boundary of the oracle region correspond to a nonzero entry in the optimal solution (all boundary points of the oracle region correspond to nonzero entries in the optimal solution if and only if the solution to the SSC-BP is unique). All these intricacies in the geometry of the solution lead to a challenging situation in the development of a provable correct active support update for solving (4.3).

We now present our solution to such a situation. We propose to use the following as the alternative to step 4 in Algorithm 6:

$$\textbf{4"}\text{: } \mathcal{W}^{(k+1)} \leftarrow \{j : |\langle \boldsymbol{a}_j, \boldsymbol{\delta}^{(k)} \rangle| > \lambda\} \cup \{j : [\boldsymbol{c}^{(k)}]_j \neq 0\}, \tag{4.12}$$

i.e., $\mathcal{W}^{(k+1)}$ contains the union of points in the oracle region $\Delta(\boldsymbol{b}, \mathbf{A}_{\mathcal{W}^{(k)}})$ and points that have nonzero entries in the current solution $\boldsymbol{c}^{(k)}$. Notice that $\{j : [\boldsymbol{c}^{(k)}]_j \neq 0\} \subseteq \partial\Delta(\boldsymbol{b}, \mathbf{A}_{\mathcal{W}^{(k)}})$, therefore the two operands in the union in (4.12) are disjoint sets. With this modification, one can show that Algorithm 6 converges to an optimal solution in a finite number of iterations. The proof is essentially the same as before and omitted here. In the case when the solution is not unique, the solution that Algorithm 6 converges to depends upon the initialization $\mathcal{W}^{(0)}$ as well as the specific solution given by the solver in step 3.

For $\lambda \in [0, 1)$, we have $\{j : [\boldsymbol{c}^{(k)}]_j \neq 0\} \subseteq \Delta(\boldsymbol{b}, \mathbf{A}_{\mathcal{W}^{(k)}})$ by the property of the oracle region. Thus, the alternative step specified by (4.12) applies to any $\lambda \in [0, 1]$. We write this as a theorem.

Theorem 44. *Algorithm 6 with the alternative step 4 specified in* (4.12) *converges to an optimal solution* $\boldsymbol{c}^*(\boldsymbol{b}, \mathbf{A})$ *in a finite number of iterations for all* $\lambda \in [0, 1]$.

4.3 Divide-and-conquer method

The active support approaches presented in the previous section are very efficient for solving sparse recovery problems and they have linear computational complexity in the number of data points. Nonetheless, sparse and elastic net subspace clustering methods require solving N sparse optimization problems, which raises the overall complexity to $\mathcal{O}(N^2)$. Consequently, their applicability is limited to data sets that contain, say, 1 million data points. In order to handle even larger scale data, it is necessary to consider beyond the scope of efficient sparse optimization algorithm. In this section, we present a new algorithm for subspace clustering on large scale data that is based on the idea of divide-and-conquer. We call our approach SSC-DC.

The entire procedure of SSC-DC (see Algorithm 7) consists of four phases. First, instead of learning a representation matrix for the entire data, SSC-DC

randomly splits the data into smaller chunks and then independently performs SSC (i.e., either SSC-BP or SSC-OMP) on each chunk. Second, since the resulting clusters may contain points from more than one subspace, SSC-DC uses an outlier pursuit procedure to separate inliers from outliers within each cluster. Third, a new similarity measure between clusters is used to merge clusters of inliers that correspond to the same subspace. Finally, once the clusters have been merged, the outliers are reclustered by assigning them to one of the merged clusters.

4.3.1 Phase 1: split and cluster

First, SSC-DC partitions the N data points into B disjoint chunks, $\{\mathbf{X}^{(b)}\}_{b=1}^{B}$, where the size of each chunk N/B (we assume that B divides N) is small enough so that the chunks can be handled by modern SSC methods. Once the chunks have been formed, SSC-BP or SSC-OMP is applied to each chunk $\mathbf{X}^{(b)}$ to get n clusters $\{\mathbf{X}_{\ell}^{(b)}\}_{\ell=1}^{n}$, where n is the number of subspaces and is set to be the same for all chunks.

4.3.2 Phase 2: detect outliers

If the clustering within each chunk from Phase 1 were perfect, then each cluster would only contain points from a single subspace. In practice, some

clusters could be corrupted by points from other subspaces, which could significantly affect the ability to merge the clusters.

To address this issue, we apply an outlier detection algorithm to each of the clusters to identify and remove outliers. Specifically, assume that each one of the clusters $\{\mathbf{X}_\ell^{(b)}\}_{\ell\in\{1,\cdots,n\}}^{b\in\{1,\cdots,B\}}$ obtained in Phase 1 contains many points from one of the subspaces as well as a few points from other subspaces. The goal of Phase 2 is to detect and remove points from other subspaces (a.k.a. outliers), thus generating a submatrix $\bar{\mathbf{X}}_\ell^{(b)}$ of the matrix $\mathbf{X}_\ell^{(b)}$ for all $\ell \in \{1,\dots,n\}$ and $b \in \{1,\dots,B\}$ that contains only points from one of the subspaces (a.k.a. inliers). The outlier detection problem has been studied in the context of robust PCA, e.g. see [98, 154, 183]. In this work, we use the Outlier Pursuit method of [183], which aims to decompose the data matrix as $\mathbf{X}_\ell^{(b)} = L + S$. Here, L is some low-rank matrix whose non-zero columns (the inliers) span the underlying subspace containing $\mathbf{X}_\ell^{(b)}$, and S is a column-sparse matrix (i.e. there are only a few nonzero columns) whose non-zero columns correspond to the outliers. To compute L and S, one solves

$$\min_{L,S} \|L\|_* + \lambda\|S\|_{2,1} \quad \text{s.t. } \mathbf{X}_\ell^{(b)} = L + S, \tag{4.13}$$

where $\lambda > 0$ is a trade-off parameter, $\|L\|_*$ is the nuclear norm of L defined as the sum of the singular values of L, and $\|S\|_{2,1}$ is the sum of the ℓ_2-norms of the

columns of S. Once the outliers have been detected as the non-zero columns of S, we assign them to an outlier set to be processed in Phase 4.

4.3.3 Phase 3: merge subspaces

Given the data matrices $\{\bar{\mathbf{X}}_\ell^{(b)}\}_{\ell\in\{1,\cdots,n\}}^{b\in\{1,\cdots,B\}}$, each one containing ideally data from only one subspace, the goal of Phase 3 is to merge clusters whose data come from the same subspace. For this purpose, we adopt a two-step procedure in which pairwise similarities between clusters are computed and spectral clustering is applied to the resulting similarity.

A classical measure of the similarity between two subspaces is their principal angle, which can be computed from the largest singular value of the matrix $U^\top V$, where U and V are orthogonal bases for the two subspaces. However, since real data typically contains noise, it can be difficult to compute a basis for each subspace since the dimensions of the subspaces are unknown and nontrivial to estimate. Given two data submatrices whose columns are from the same subspace, either overestimation or underestimation of the subspace dimension can result in an inaccurate estimation of the principal angle. Thus, there is a need to design measures of subspace similarity that do not require an estimate of the subspace dimension and are robust to noisy data.

Motivated by the fact that points in the same subspace can be used to mutually express each other, we design a "cross-expressiveness" based similarity

measure for two subspaces. The idea is that if the columns of two matrices Y_1 and Y_2 are drawn from the same subspace, then it holds that $Y_1 = Y_2 \cdot C_1$ for some C_1, i.e., each column of Y_1 can be expressed using the columns from Y_2. On the other hand, if Y_1 and Y_2 are drawn from two different subspaces (assume that one subspace does not contain the other), then such a representation C_1 does not exist because the columns of Y_1 cannot be expressed using columns from Y_2. Motivated by this ideal setting, and to cope with the possible existence of noise in the data, we compute the representation C_1 as:

$$C_1 = \arg\min_{C} \|Y_1 - Y_2 C\|_F^2 + \lambda \|C\|_F^2, \tag{4.14}$$

for some weighting parameter $\lambda > 0$. Note that problem (4.14) has the closed form solution $C_1 = (Y_2^\top Y_2 + \lambda I)^{-1} Y_2^\top Y_1$. Our new dissimilarity measure between Y_1 and Y_2 is then defined as

$$d(Y_1, Y_2) = \frac{1}{2} \left(\frac{\|Y_1 - Y_2 C_1\|_F}{\|Y_1\|_F} + \frac{\|Y_2 - Y_1 C_2\|_F}{\|Y_2\|_F} \right), \tag{4.15}$$

where C_1 is computed from (4.14) and C_2 is computed by swapping Y_1 and Y_2 in (4.14). Based on the dissimilarity measure (4.15), the similarity between Y_1 and Y_2 is computed as $\exp(-d(Y_1, Y_2)/(2\sigma^2))$ for some parameter $\sigma > 0$.

4.3.4 Phase 4: recluster outliers

After the merging procedure in Phase 3, the algorithm has generated n clusters $\{\bar{\mathbf{X}}_\ell\}_{\ell=1,\cdots,n}$, each of which will ideally contain only points from one of the ground-truth subspaces. However, the points that were detected as outliers in Phase 2 still need to be assigned to one of the clusters. A simple approach for assigning outliers to clusters would be to fit a subspace to each one of the clusters that have already been obtained and then assign each outlier to one of those n clusters. However, this would require us to know the dimension of the subspaces, and any errors in the estimation of the dimensions could lead to errors in the assignments.

To address this issue, we note that since the class labels for clusters $\{\bar{\mathbf{X}}_\ell\}_{\ell=1}^n$ have already been generated, we can treat $\{\bar{\mathbf{X}}_\ell\}_{\ell=1}^n$ as training data and use any supervised classification technique to classify each point in the outlier set. Here, we adopt a representation based classification technique [179, 196]. If we define $\bar{\mathbf{X}} = [\bar{\mathbf{X}}_1, \cdots, \bar{\mathbf{X}}_n]$, then a representation for any outlier point $\boldsymbol{y}$ may be found by solving[1]

$$\min_{\boldsymbol{c}} \|\boldsymbol{y} - \bar{\mathbf{X}}\boldsymbol{c}\|_2^2 + \lambda \|\boldsymbol{c}\|_2^2 \tag{4.16}$$

for some parameter $\lambda > 0$. Ideally, the nonzero entries in the representation

[1] Observe that this problem is potentially huge when D and N are large. However, when D is small one can use the inversion lemma to solve for $\boldsymbol{c}$ efficiently. As we shall see in our experiments, the time spent in Phase 4 is not significant in practice. That being said, simpler classification methods can be tried when D and N are both large.

Algorithm 7 SSC Divide-and-conquer

Input: The data matrix $\mathbf{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N]$, the number of clusters n, and the number of chunks B.

1: **Phase 1: Split and cluster**

Split the data matrix $\mathbf{X}$ evenly into $\{\mathbf{X}^{(b)}\}^{b=1,\cdots,B}$.

Apply SSC on each $\mathbf{X}^{(b)}$ to get clusters $\{\mathbf{X}_\ell^{(b)}\}_{\ell=1,\cdots,n}^{b=1,\cdots,B}$.

2: **Phase 2: Detect outliers**

For each matrix in $\{\mathbf{X}_\ell^{(b)}\}_{\ell=1,\cdots,n}^{b=1,\cdots,B}$ solve (4.13). Place points corresponding to nonzero columns of S into the outlier set, and denote the matrix containing the remaining points as $\bar{\mathbf{X}}_\ell^{(b)}$.

3: **Phase 3: Merge subspaces**

Compute similarities $\exp(-d(\bar{\mathbf{X}}_\ell^{(b)}, \bar{\mathbf{X}}_{\ell'}^{(b')})/(2\sigma^2))$ using (4.15) for all pairs $(b, \ell) \neq (b', \ell')$. Apply spectral clustering using this similarity matrix to get clusters $\{\bar{\mathbf{X}}_\ell\}_{\ell=1,\cdots,n}$.

4: **Phase 4: Recluster outliers**

Assign each $\boldsymbol{y}$ in the outlier set to one of the clusters in $\{\bar{\mathbf{X}}_\ell\}_{\ell=1,\cdots,n}$ by using (4.17).

Output: A clustering of $\mathbf{X}$ into n clusters $\{\bar{\mathbf{X}}_\ell\}_{\ell=1,\cdots,n}$.

vector $\boldsymbol{c}$ will correspond to points in $\bar{\mathbf{X}}$ that are from the same subspace as $\boldsymbol{y}$. In practice, nonzero entries of $\boldsymbol{c}$ may be distributed among multiple subspaces. Following the procedure in [179], let $\boldsymbol{\delta}_\ell(\boldsymbol{c})$ be a vector of the same size as $\boldsymbol{c}$

such that the entries of $\boldsymbol{\delta}_\ell(\boldsymbol{c})$ are all zero except for those that correspond to $\bar{\mathbf{X}}_\ell$, which are equal to the corresponding entries of $\boldsymbol{c}$. The point $\boldsymbol{y}$ is then assigned to the class ℓ that gives the minimum representation residual of $\boldsymbol{y}$ using $\boldsymbol{\delta}_\ell(\boldsymbol{c})$, i.e., to the class ℓ that solves

$$\min_\ell \|\boldsymbol{y} - \bar{\mathbf{X}}\boldsymbol{\delta}_\ell(\boldsymbol{c})\|_2. \tag{4.17}$$

This completes the description of our SSC-DC framework.

4.4 Experiments

4.4.1 Experiments on synthetic data

4.4.1.1 Evaluation of the active support method

We first demonstrate the computational efficiency of our active support algorithms over the existing optimization algorithms. To do this, we generate synthetic datasets that contain data lying in a union of subspaces, and apply SSC-BP where the BP optimization problem is solved by different algorithms.

Data. We randomly generate $n = 10$ subspaces each of dimension $d = 5$ in an ambient space of dimension $D = 10$. Each subspace contains $N_i = \rho d$ sample points randomly generated on the unit sphere, where ρ is varied from 2 to $20{,}000$

so that the number of points varies from 100 to $1{,}000{,}000$.

Methods. We compare our active support algorithm with existing sparse optimization algorithms. Due to the overwhelming volume of the literature in the area of sparse coding, we only compare with the following 5 most representative solvers: the interior point based ℓ_1 regularized least squares (L1LS) method in [87]; the gradient projection for sparse reconstruction (GPSR) algorithm in [66]; the proximal gradient based fast iterative soft-thresholding algorithm (FISTA) in [19]; the alternating direction method of multipliers (this is the method adopted in a previous paper [60] that studies SSC-BP); and the LASSO version of the LARS algorithm [56] that is implemented in the sparse modeling software (SPAMS) http://spams-devel.gforge.inria.fr/. For L1LS and FISTA, we use the implementation that accompanies the review paper [186]. For GPSR we use the implementation from the website http://www.lx.it.pt/ mtf/GPSR/. For ADMM, we use the code from the paper [60].

The L1LS, GPSR, FISTA and ADMM methods are based on iterative procedures and they do not converge to the exact solution to BP in a finite number of iterations (in general). Therefore, there is a need to choose a termination condition for them. For L1LS and FISTA, we set the number of iterations to be 20 and 150, respectively. For GPSR and ADMM, we use the default parameters in their respective implementations. The LARS converges to the optimal solution in a finite number of iterations.

For our active support method, we use the LARS algorithm to solve the subproblem in step 3 of Algorithm 6. Both of our active support method and all comparing methods solve the same optimization problem in (4.3) with γ set to be 100.

Metrics. We use the following metrics to evaluate the performance of different algorithms.

– *Clustering accuracy:* this is the percentage of correctly labeled data points. It is computed by matching the estimated and true labels as

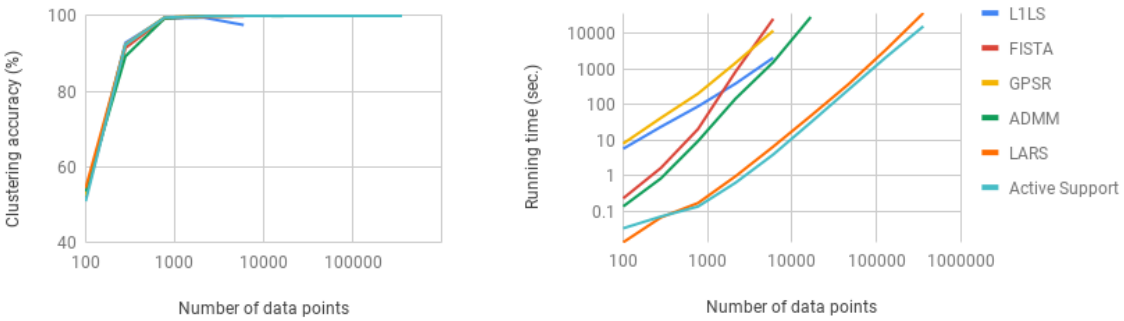
$$\max_{\pi} \frac{100}{N} \sum_{ij} Q^{est}_{\pi(i)j} Q^{true}_{ij},$$

where π is a permutation of the n groups, Q^{est} and Q^{true} are the estimated and ground-truth labeling of data, respectively, with their (i,j)th entry being equal to one if point j belongs to cluster i and zero otherwise.

– *Running time:* this is the running time for each clustering task using ®Matlab.

The reported numbers in all the experiments are averages over 10 trials.

Results. In Figure 4.2 we report the clustering accuracy as well as the running time of different BP method. We see that the clustering accuracy of different methods are very close to each other, which is expected as all methods solve the same BP optimization problem. In terms of running time, both L1LS and GPSR are two orders of magnitude slower than our active support method,



(a) Clustering accuracy **(b)** Running time

Figure 4.2: Comparison of the active support method with other algorithms for solving the BP optimization problem.

and they can only handle around $6{,}000$ data points with in the time limit of 24 hours. The FISTA and ADMM are more efficient than L1LS and GPSR when the number of data point is relatively small. But still, both FISTA and ADMM are very slow relative to the active support method, and they cannot handle large scale data.

The LARS and the active support methods are the two most efficient algorithms and both of them can handle $\sim 360{,}000$ data points within 24 hours. While LARS is almost as fast as the active support method in this experiment, its running time increases dramatically as the dimension of the subspaces increases. This is illustrated in the following experiment.

Effect of subspace dimension. To evaluate the effect of subspace dimension on the performance of our active support method, we fix the number of data points to be 200 for each subspace and vary the dimension of the subspaces. Specifically, we set the ambient dimension to be $D = 40$ and vary the subspace

dimension d in the range of $[2, 20]$. The running time of the active support method as well as that of the LARS algorithm are reported in Figure 4.3. We can clearly see that the active support method is much less affected by the subspace dimension when compared with the LARS.

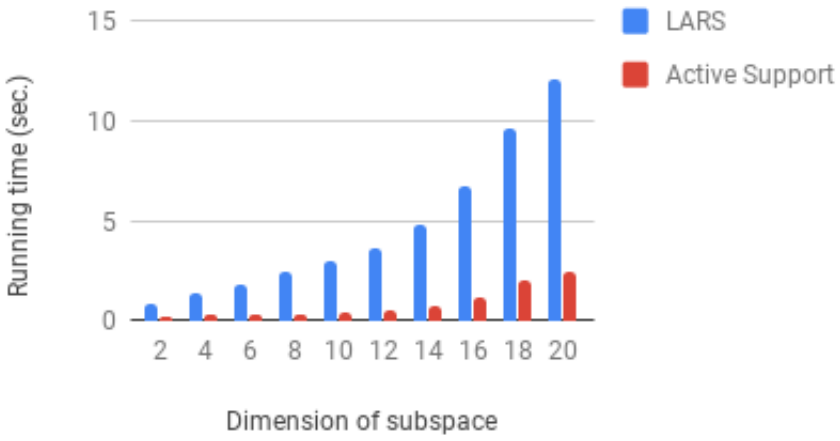


Figure 4.3: Effect of subspace dimension on the running time of the active support method.

4.4.1.2 Comparison of SSC-BP, SSC-OMP and EnSC

Equipped with the active support algorithms for solving their respective optimization problems in SSC-BP, SSC-OMP and EnSC, all these three subspace clustering techniques can be carried out very efficiently. In this subsection, we conduct further experiments to compare these three methods in terms of their efficiency as well as their ability in delivering subspace-preserving solutions, connected similarity graphs, and good clustering accuracy.

Data. We generate synthetic data following the same procedure as described

in Section 4.4.1.1.

Metrics. In addition to clustering accuracy and running time that are described in Section 4.4.1.1, we also using the following metrics.

To evaluate the degree to which the subspace-preserving property is satisfied, we use two metrics where the first one is a direct measure of whether the solution is subspace-preserving or not, and the second one measures how close the coefficients are from being subspace-preserving.

– *Percentage of subspace-preserving representations:* this is the percentage of points whose representations are subspace-preserving. The coefficients with absolute value less than 10^{-5} are considered zero. For a subspace-preserving solution, the percentage of subspace-preserving representations is 100.

– *Subspace-preserving representation error [60]:* for each $\boldsymbol{c}_j$ in the representation matrix, we compute the fraction of its ℓ_1 norm that comes from other subspaces and then average over all j, i.e., $\frac{100}{N}\sum_j(1-\sum_i(\omega_{ij}\cdot|c_{ij}|)/\|\boldsymbol{c}_j\|_1)$, where $\omega_{ij}\in\{0,1\}$ is the true data similarity matrix. A subspace-preserving representation gives zero subspace-preserving representation error.

Now, the performance of subspace clustering depends not only on the subspace-preserving property, but also the connectivity of the similarity graph, i.e., whether the data points in each cluster form a connected component of the graph. To evaluate the ability of different methods in delivering well-connected similarity graphs, we use the following two metrics.

– *Number of nonzero entries:* Denser graphs typically implies better connections in the similarity graph. Given the representation matrix $\mathbf{C} = [\boldsymbol{c}_1, \cdots, \boldsymbol{c}_N]$, we report the averaged number of nonzero entries across different columns, i.e., $\sum_{j=1}^{N} \|\boldsymbol{c}_j\|_0 / N$.

– *Connectivity:* For an undirected graph with weights $\mathbf{W} \in \mathbb{R}^{N \times N}$ and degree matrix $\mathbf{D} = \text{diag}(\mathbf{W} \cdot \mathbf{1})$, where $\mathbf{1}$ is the vector of all ones, we use the second smallest eigenvalue λ_2 of the normalized Laplacian $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ to measure the connectivity of the graph; λ_2 is in the range $[0, \frac{n-1}{n}]$ and is zero if and only if the graph is not connected [43, 65]. In our case, we compute the algebraic connectivity for each cluster, λ_2^ℓ, and take the quantity $\min_\ell \lambda_2^\ell$ as the measure of connectivity.

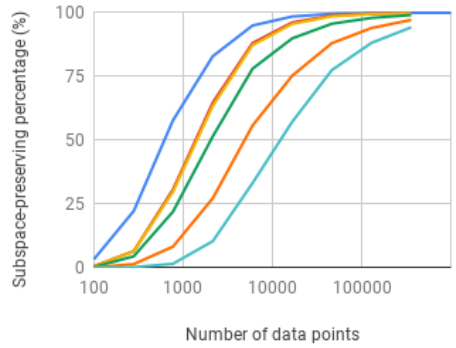
The reported numbers in all the experiments are averages over 10 trials.

Algorithm parameters. Unless otherwise specified, we set ϵ in Algorithm 1 to be 10^{-10} and $k_{\max}$ to be $d = 10$ for SSC-OMP. For SSC-BP and EnSC, we set γ in (4.3) and (4.4) to be 100. We use the active support algorithm for solving the BP and elastic net optimization problems in SSC-BP and EnSC, where the subproblems in the active support algorithm are solved by the LARS algorithm in the SPAMS package and the RFSS solvers [86], respectively. EnSC has another tuning parameter λ in (4.4) which balances between the ℓ_1 and ℓ_2 regularizations. To understand the effect of this parameter, we report results with varying values of λ and denote the corresponding methods as EnSC$_\lambda$. Note

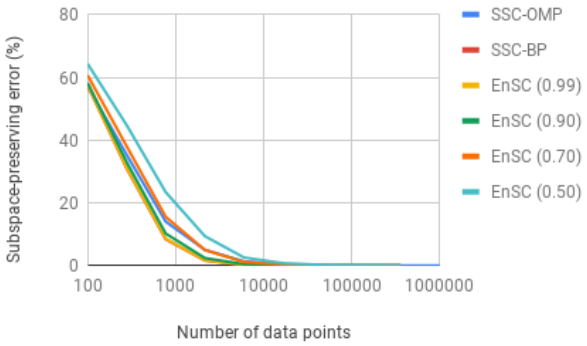
that by this notation, EnSC_λ is the same as SSC-BP.

Results. The subspace-preserving representation percentage and error are plotted in Figure 4.4a and 4.4b. Observe that the probability that SSC-BP and SSC-OMP give subspace-preserving solutions grows as the density of data point increases. This matches our theoretical analysis of these two methods stated in Theorem 35. When comparing SSC-BP with SSC-OMP, we can see that SSC-OMP has higher chance of getting a subspace-preserving solution than SSC-BP, while the subspace-preserving error of SSC-BP is lower than that of SSC-OMP. As for EnSC, we see that the percentage of subspace-preserving solution decreases and the subspace-preserving error increases as the parameter λ decreases.

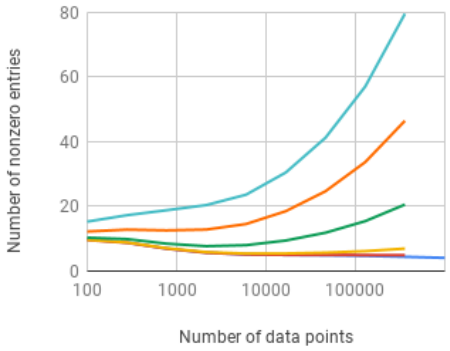
From a subspace clustering perspective, we are more interested in how well the method performs in terms of clustering accuracy, which depends not only on the subspace-preserving property but also on the connectivity of the representation. In Figure 4.4c and Figure 4.4d, we show the number of nonzero entries in the representation matrix as well as the connectivity measure. We observe that while SSC-BP and SSC-OMP have similar number of nonzero entries in their representations, the connectivity of SSC-BP is significantly higher than that of SSC-OMP. This may explain why SSC-BP is significantly better than SSC-OMP in terms of clustering accuracy, as we can see from Figure 4.4e. However, we observe that as the density of data points increases, the difference in



(a) Subspace-preserving percentage

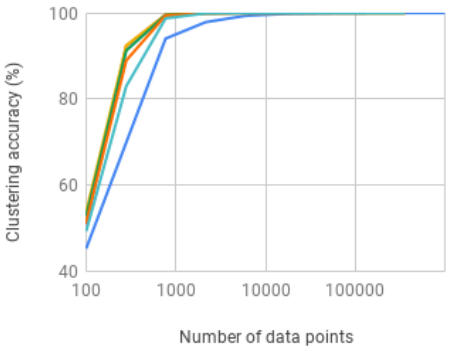


(b) Subspace-preserving error

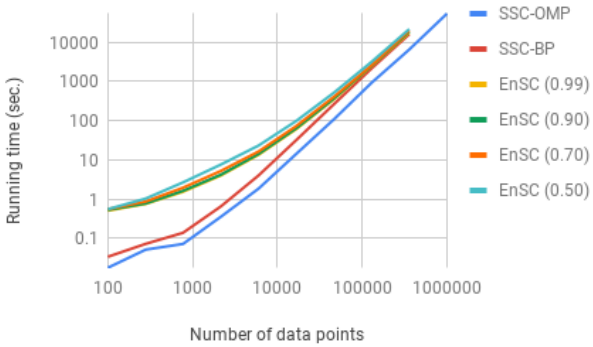


(c) Number of nonzero entries

(d) Connectivity



(e) Clustering accuracy



(f) Computational time

Figure 4.4: Performance of SSC-OMP, SSC-BP and EnSC on synthetic data. The data are drawn from 10 subspaces of dimension 5 in ambient dimension 10. Each subspace contains the same number of points and the overall number of points is varied from 100 to $1{,}000{,}000$ and is shown in log scale. Notice that the figure that reports the running time uses log scale in the y-axis.

clustering accuracy also decreases, and SSC-OMP seems to achieve arbitrarily good clustering accuracy for large N. The EnSC with decreasing values of λ gives higher number of nonzero entries and higher connectivity, showing that it is potentially able to reduce the connectivity issue. However, in this experiment the SSC-BP does not appear to suffer from the connectivity issue, and the clustering accuracy of EnSC is not higher than that of SSC.

In terms of running time, it is evident from Figure 4.4f that SSC-OMP is slightly faster than the SSC-BP, so it is more suitable for large scale problems. The EnSC is slower than SSC when the number of data point is small, probably because the RFSS solver is slower than the LARS solver for small scale problems. This difference in running time diminishes when the scale of the problem increases.

4.4.1.3 Evaluation of SSC-DC

To test the effectiveness of SSC-DC on large-scale datasets and to study how the number of chunks affects the running time, we design experiments using synthetic data. First, we choose 10 subspaces each of dimension 5, independently and uniformly at random in an ambient space of dimension 15. Second, we generate an equal number of data points uniformly at random on each of the subspaces. The total number of data points is varied from $10{,}000$ to $1{,}000{,}000$.

For Phase 1 of SSC-DC, the data is randomly divided into 10, 50 or 100

chunks and SSC-BP is applied to each chunk. We use the SPAMS package to solve the sparse representation problem used by SSC-BP. The clusters are merged according to Phase 3 of Algorithm 7 with $\lambda = 0.1$ in (4.14) and $\sigma = 1$. We do not perform outlier detection (i.e. Phase 2) and reclustering (i.e. Phase 4) in this experiment. The running times of different algorithms on different number of data points are shown in Figure 4.5.

The curve for SSC is the baseline, which was obtained by applying SSC-BP to the entire data, i.e., it uses one chunk. We see that while it only needs one minute to handle $10{,}000$ data points, it takes around 14 hours to cluster $\sim 360{,}000$ data points. Since the running time for each algorithm is limited to 24 hours, SSC does not finish clustering $1{,}000{,}000$ data points. If we use SSC-DC with 10 chunks, we can observe a significant reduction on the running time for all scales of the tested data. In particular, SSC-DC with 10 chunks uses around one hour to cluster $\sim 360{,}000$ in comparison to the 14 hours used by SSC. By using the divide-and-conquer strategy, our method is able to cluster $1{,}000{,}000$ data points.

When the number of chunks in SSC-DC is increased from 10 to 50, we observe that the running time increases when the data size is relatively small, but decreases when the data size is relatively large. This illustrates the trade-off in choosing the number of chunks in SSC-DC. Although increasing the number of chunks reduces the scale of the problems solved by SSC on each chunk,

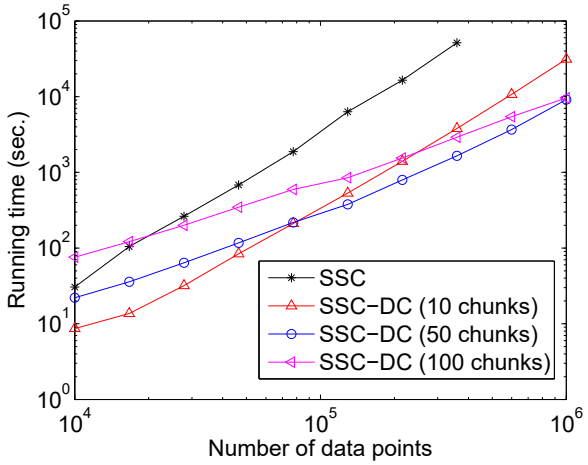


Figure 4.5: Performance of SSC-DC with different numbers of chunks on synthetic data. We randomly generate 10 subspaces of dimension 5 in an ambient space of dimension 15, and then randomly draw the same number of points from each subspace. The x-axis gives the overall number of points, which is varied from $10{,}000$ to $1{,}000{,}000$ and shown in log scale. The y-axis reports the running time in log scale. The missing points for SSC indicate that the algorithm does not finish within 24 hours.

it also increases the number of subspaces to be merged in Phase 3. Therefore, the computational cost associated with these two effects should be balanced in practice. As shown in Figure 4.5, SSC-DC (100 chunks) is less efficient than SSC-DC (50 chunks), although it is likely that SSC-DC (100 chunks) will be more efficient that SSC-DC (50 chunks) when the dataset size grows beyond $1{,}000{,}000$ data points.

4.4.2 Experiments on real data

In this section, we compare the performance of SSC-BP, SSC-OMP, EnSC and SSC-DC to several other spectral clustering based subspace clustering

methods on several large scale image datasets.

Datasets. We use four datasets presented in Table 4.1. The extended Yale B [71] dataset contains frontal face images of 38 individuals each under 64 different illumination conditions. The face images are of size 192×168, for which we downsample to 48×42. The major intra-class variation in the extended Yale B is the illumination conditions, therefore the data can be well-approximated by a union of subspaces [18]. The COIL-100 dataset [125] contains $7{,}200$ gray-scale images of 100 different objects. Each object has 72 images taken at pose intervals of 5 degrees, with the images being of size 32×32. The CIFAR-10 dataset [88] contains $60{,}000$ images from 10 object categories. The MNIST dataset [92] contains $70{,}000$ images of handwritten digits 0–9. For each image in the COIL-100, CIFAR-10 and MNIST datasets, we extract a feature vector of dimension $3{,}472$ via the scattering convolution network [28], and then project to dimension 500 using PCA. The scattering convolution network is able to extract features that are translational invariant and deformation stable (i.e. it linearizes small deformations). Therefore, these features from the COIL-100, CIFAR-10 and MNIST datasets approximately follow a union of subspaces model.

Methods. For SSC-BP and EnSC, we use the active support method to solve the BP and elastic net optimization problems, respectively, where each subproblem is solved using the LARS and the RFSS algorithms, respectively. The

Table 4.1: Dataset information for testing subspace clustering on real image databases.

	Image type	N (#data)	D (ambient dim.)	n (#groups)
Extended Yale B	Faces	2,432	2016	38
COIL-100	Objects	7,200	500	100
CIFAR-10	Objects	60,000	500	10
MNIST	Digits	70,000	500	10

parameter γ in SSC-BP and EnSC are set as $\gamma = \alpha\gamma_0$ where $\alpha > 1$ is a hyperparameter and γ_0 is the smallest value of γ such that the solutions to (4.3) and (4.4) are zero vectors, respectively. The parameter λ in EnSC is varied to show its effect on the clustering performance. In addition to constructing the similarity graph as described in Algorithm 3, we also explore a k-nearest neighbor (kNN) based similarity graph for computing a final clustering result from the representation coefficients. First, the coefficient vectors $\{\boldsymbol{c}_j\}$ are normalized, i.e., we set $\tilde{\boldsymbol{c}}_j = \boldsymbol{c}_j / \|\boldsymbol{c}_j\|_2$. Then, for each $\tilde{\boldsymbol{c}}_j$ we find k-nearest neighbors with the largest positive inner product with $\tilde{\boldsymbol{c}}_j$. Finally, we compute an affinity matrix from the k-nearest neighbors and apply spectral clustering to get the segmentation. We will denote the version of SSC-BP and EnSC$_\lambda$ that use this procedure for computing the segmentation by SSC-BP (k-NN) and EnSC$_\lambda$ (k-NN), respectively. For SSC-OMP, we observe that this k-NN method for computing the segmentation does not improve the clustering accuracy, therefore we do not report its results.

We compare our method with several state-of-the-art subspace clustering

Table 4.2: Performance of subspace clustering methods on large-scale data - clustering accuracy (in percentage). The value "M" means that the memory limit of 20GB was exceeded, and the value "T" means that the time limit of 24 hours was reached. The three highest clustering accuracy for each dataset are shown in boldface.

Datasets	EYaleB	COIL-100	CIFAR-10	MNIST
k-Means	9.4	50.2	20.0	62.6
Spectral	48.5	78.6	22.8	85.4
LRR	67.6	57.3	M	M
LRR (k-NN)	92.8	**84.5**	M	M
LRSC	70.9	57.3	M	M
LRSC (k-NN)	92.9	**84.2**	M	M
O-LRSC	15.4	52.9	20.6	76.0
LSR	68.4	59.2	M	M
LSR (k-NN)	**95.2**	83.5	M	M
SSC-ADMM	65.2	78.2	M	M
SSC-ADMM (k-NN)	**94.1**	50.5	M	M
ℓ_0-SSC	85.2	76.0	T	T
ℓ_0-SSC (k-NN)	89.5	45.5	T	T
NSN	83.7	72.6	T	T
SSC-OMP	80.6	65.9	10.4	95.5
SSC-BP	68.1	81.9	16.5	92.5
SSC-BP (k-NN)	87.9	44.6	**27.5**	**98.3**
EnSC$_{0.99}$	67.7	**86.2**	16.5	92.5
EnSC$_{0.90}$	67.9	79.7	16.5	94.0
EnSC$_{0.70}$	59.8	73.8	20.7	82.8
EnSC$_{0.50}$	63.9	72.0	22.3	83.0
EnSC$_{0.30}$	65.0	69.3	22.4	82.9
EnSC$_{0.99}$ (k-NN)	90.4	46.3	**27.6**	98.2
EnSC$_{0.90}$ (k-NN)	91.8	42.0	27.3	98.2
EnSC$_{0.70}$ (k-NN)	**94.5**	45.7	**27.7**	**98.3**
EnSC$_{0.50}$ (k-NN)	88.0	54.0	T	**98.3**
EnSC$_{0.30}$ (k-NN)	87.3	60.3	T	85.7

Table 4.3: Performance of subspace clustering methods on large-scale data - running time (in seconds). The value "M" means that the memory limit of 20GB was exceeded, and the value "T" means that the time limit of 24 hours was reached. The three lowest running time for each dataset are shown in boldface.

Method \ Dataset	EYaleB	COIL-100	CIFAR-10	MNIST
k-Means	65	104	**104**	**123**
Spectral	14	45	4966	**1380**
LRR	2629	671	M	M
LRR (k-NN)	2629	671	M	M
LRSC	13	62	M	M
LRSC (k-NN)	13	62	M	M
O-LRSC	224	1491	**340**	304
LSR	**5**	46	M	M
LSR (k-NN)	**5**	46	M	M
SSC-ADMM	591	5781	M	M
SSC-ADMM (k-NN)	592	5619	M	M
ℓ_0-SSC	4132	16164	T	T
ℓ_0-SSC (k-NN)	16807	15683	T	T
NSN	383	1075	T	T
SSC-OMP	**8**	33	5460	1475
SSC-BP	46	34	**957**	**1151**
SSC-BP (k-NN)	46	**26**	2261	1453
EnSC$_{0.99}$	46	34	1122	2056
EnSC$_{0.90}$	47	38	1092	2234
EnSC$_{0.70}$	60	**24**	1248	2228
EnSC$_{0.50}$	54	**22**	2505	2482
EnSC$_{0.30}$	67	29	9264	3263
EnSC$_{0.99}$ (k-NN)	46	50	12178	3149
EnSC$_{0.90}$ (k-NN)	47	48	10209	3562
EnSC$_{0.70}$ (k-NN)	60	49	10771	2264
EnSC$_{0.50}$ (k-NN)	54	56	T	3439
EnSC$_{0.30}$ (k-NN)	67	73	T	3069

methods that may be categorized into three groups. The first group contains k-means clustering and spectral clustering on the k-nearest neighbors graph, named "Spectral" in the following figures and tables. It is known [80] that Spectral is a provably correct method for subspace clustering. The k-means and k-d trees algorithms used to compute the k-nearest neighbor graph in Spectral are implemented using the VLFeat toolbox [159].

The second group consists of low-rank (i.e., LRR, LRSC and O-LRSC) and least squares (i.e., LSR) based methods. Such methods produce dense data similarity matrices, there are are not able to handle large scale datasets due to memory constraints. On exception is the O-LRSC [137], which is an online version of the LRSC and is designed to handle large scale data.

The final group consists of SSC-ADMM [60] (i.e., SSC-BP with a different solver), ℓ_0-SSC [188] and NSN [129]. These algorithms build sparse similarity matrices.

Results. To the best of our knowledge, a comparison of all these methods in the datasets we have chosen has not been reported in prior work. Thus, we run all experiments and tune the parameters for each method to give the best clustering accuracy. The results are reported in Table 4.2 and Table 4.3.

We see that on the CIFAR-10 and MNIST databases, very few methods other than SSC-OMP, SSC-BP and EnSC are able to produce a result under our 20GB memory and 24 hours running time constraints of the experiments. This

clearly indicates that our active support method is very efficient. On MNIST, both SSC-OMP and SSC-BP achieve > 90 percent accuracy which is significantly higher than other baseline methods. On CIFAR-10, no method is able to produce high clustering accuracy since the dataset is very difficult. Nonetheless, the SSC-BP (k-NN) achieves the best clustering performance (other than EnSC). On both MNIST and CIFAR-10, we can see that the EnSC is able to further improve over SSC-BP for appropriate values of λ. For example, on MNIST one can clearly see that the $\text{EnSC}_{0.90}$ produces significantly higher clustering accuracy than SSC-BP (theoretically is the same as $\text{EnSC}_{1.00}$) and EnSC_{λ} with other values of λ. This verifies the design idea that the EnSC with properly chosen λ is able to achieve the best trade-off between subspace-preserving property and connectivity, which leads to the best clustering performance.

The EYaleB and the COIL-100 are relatively small scale databases which allow us to compare with all other methods that are being tested. We can see that the EnSC with appropriate values of λ are among the best performing methods, while the SSC-OMP and SSC-BP are slightly worse.

In Table 4.4 we report clustering accuracy and running time for SSC-DC on the MNIST dataset. In particular, we use SSC-OMP for subspace clustering in Phase 1. Note that the solution for S in (4.13), which is supposed to be column sparse, contains dense noise because the dataset contains noise. Therefore, we compute the ℓ_2-norm of the columns of S, and declare those that are larger than

a threshold as outliers; we found it difficult to determine a proper threshold for declaring outliers. For simplicity, we sort the ℓ_2-norms of the columns of S in descending order, and declare the data that correspond to the first 10% as outliers. While this may produce many false outliers, it is unlikely to significantly affect the final clustering result because the false outliers are reclustered in Phase 4.

From Table 4.4, we see that in terms of clustering accuracy, the performance of SSC-DC becomes worse as the number of chunks increases. This is in accordance with the empirical results for SSC-OMP as reported in Figure 4.4, where it is shown that the clustering performance of SSC-OMP becomes better when there are more samples in each subspace. Therefore, the clustering on each chunk in Phase 1 becomes less accurate as the number of chunks increases, which affects the final clustering accuracy. Table 4.4 also reports the performance of SSC-OMP, which exactly corresponds to the same computation in Phase 1 of SSC-DC (1 chunk). Note that the clustering accuracy of SSC-OMP is considerably lower than SSC-DC (1 chunk), which demonstrates that the outlier detection and reclassification procedures in Phase 2 and Phase 4 of SSC-DC effectively boost clustering accuracy.

The third column of Table 4.4 reports the overall running time of SSC-DC. It shows that the running time first decreases as the number of chunks increases, and then starts increasing once the number of chunks is larger than 20.

Table 4.4: Performance of SSC-DC on the MNIST dataset. The results are averages over 10 independent trials.

Method	Acc. (%)	Time (sec.)				
		Total	P1	P2	P3	P4
SSC-DC (1)	96.55	5254	1825	3304	30	93
SSC-DC (2)	96.10	4390	1049	3185	59	94
SSC-DC (5)	94.90	1596	436	937	134	88
SSC-DC (10)	93.04	1081	272	454	266	88
SSC-DC (20)	91.46	1081	196	274	523	87
SSC-DC (50)	89.07	1689	169	183	1243	93
SSC-DC (100)	85.46	2635	148	132	2260	94
SSC-DC (200)	78.93	5518	144	119	5161	93
SSC-OMP	93.07	1825	NA	NA	NA	NA

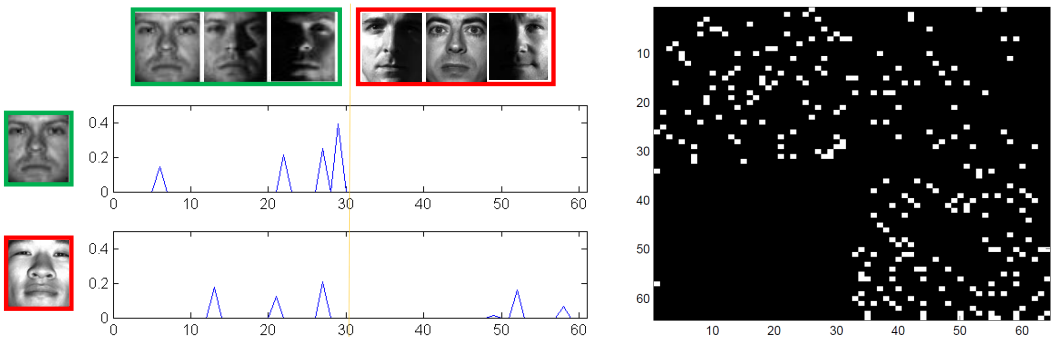
This behavior can be explained by the breakdown of the running time for the 4 phases as reported in columns 4–7 of Table 4.4. In particular, notice that the running time of Phase 1 strictly decreases as the number of chunks increases, showing that SSC becomes more efficient when the chunks are smaller. On the other hand, the running time of Phase 3 strictly increases as the number of chunks increases, showing that it quickly becomes prohibitively expensive to merge subspaces as the number of subspace increases. At last, we note that Phase 2 (outlier detection via Outlier Pursuit) is the bottleneck in running time when the number of chunks is small. Thus, SSC-DC has the potential to be more efficient by using more scalable algorithms for outlier detection.

Chapter 5

Subspace Clustering with Outliers

When a given set of data points lies in a union of subspaces with no corruptions, the sparse subspace clustering (SSC) and elastic-net subspace clustering (EnSC) methods are able to produce data similarity with desired subspace-preserving properties for data clustering. In practice, many computer vision tasks involve processing data that is contaminated by outliers, which are points that do not lie in the union of the inlier subspaces. In such cases, the performance of SSC and EnSC for subspace clustering can be significantly compromised (see Figure 1.8).

In this chapter, we address the issue of outliers in subspace clustering tasks. The solution that we propose is a novel outlier detection method based on the



(a) Two exemplar representation vectors **(b)** Representation matrix C

Figure 5.1: An illustration of a self-representation matrix C in the presence of outliers. The first 32 columns of the data matrix X correspond to 32 images of one individual under different illuminations from the Extended Yale B database, and the next 32 images are randomly chosen from all other individuals; three examples from each category are shown near the top of 5.1a. We also show a typical representation vector for an inlier and an outlier image in 5.1a, and the complete representation matrix C in 5.1b, where white and black denote $c_{ij} \neq 0$ and $c_{ij} = 0$. Notice that inliers use only other inliers in their representation, while outliers use both inliers and outliers in their representations.

self-expressiveness property of data as in EnSC. Recall from the previous chapter that if the columns of $\mathbf{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N]$ lie in multiple subspaces, then EnSC computes self-representation coefficients by solving the following optimization problem

$$\min_{\boldsymbol{c}_j} \lambda \|\boldsymbol{c}_j\|_1 + \frac{1-\lambda}{2}\|\boldsymbol{c}\|_2^2 + \frac{\gamma}{2}\|\boldsymbol{x}_j - \mathbf{X}\boldsymbol{c}_j\|_2^2 \quad \text{s.t. } c_{jj} = 0 \tag{5.1}$$

for some $\lambda \in [0, 1]$ and $\gamma > 0$. Then, an *undirected* similarity graph is constructed from $\mathbf{C} = [\boldsymbol{c}_1, \cdots, \boldsymbol{c}_N]$ in which each vertex corresponds to a data point, and vertices corresponding to $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are connected if either c_{ij} or c_{ji} is nonzero. Such a graph is then used to segment the data into their respective

subspaces by applying spectral clustering to the graph's Laplacian.

Consider now the case where X contains outliers to the subspaces. Figure 5.1 illustrates an example representation matrix C computed from (5.1) for data drawn from a single subspace (face images from one individual) plus outliers (other images). In this case, the representation C is such that inliers express themselves as linear combinations of a few other inliers, while outliers express themselves as linear combinations of both inliers and outliers. Motivated by this observation, we use a *directed* graph to model data relations: a directed edge from $\boldsymbol{x}_j$ to $\boldsymbol{x}_i$ indicates that $\boldsymbol{x}_j$ uses $\boldsymbol{x}_i$ in its representation (i.e. $c_{ij} \neq 0$). Then a random walk on the representation graph initialized at an outlier will not return to the set of outliers since once the random walk reaches an inlier it cannot return to the outliers. Therefore, we design a random walk process and identify outliers as those whose probabilities tend to zero.

5.1 Prior art in outlier detection

In this chapter, we address the problem of outlier detection in the setting when the inlier data are assumed to lie close to a union of unknown low-dimensional subspaces (low relative to the dimension of the ambient space). A traditional method for solving this problem is RANSAC [67], which is based on randomly selecting a subset of points, fitting a subspace to them, and counting

the number of points that are well fit by this subspace; this process is repeated for sufficiently many trials and the best fit is chosen. RANSAC is intrinsically combinatorial and the number of trials needed to find a good estimate of the subspace grows exponentially with the subspace dimension. Consequently, the methods of choice have been to robustly learn the subspaces by penalizing the sum of *unsquared* distances (in lieu of *squared* distances used in classical methods such as PCA) of points to the closest subspace [52, 95, 198, 199]. Such a penalty is robust to outliers because it reduces the contributions from large residuals arising from outliers. However, the optimization problem is usually nonconvex and a good initialization is extremely important for finding the optimal solution.

The groundbreaking work of Wright et al. [176] and Candès et al. [32] on using convex optimization techniques to solve the PCA problem with robustness to corrupted entries has led to many recent methods for PCA with robustness to outliers [98, 102, 115, 183, 200]. For example, Outlier Pursuit [183] uses the nuclear norm $\|\cdot\|_*$ to seek low-rank solutions by solving the problem $\min_{\mathbf{L}} \|\mathbf{X}-\mathbf{L}\|_{2,1}+\lambda\|\mathbf{L}\|_*$ for some $\lambda > 0$. REAPER [98] models the subspace by the orthoprojector Π that minimizes $\|\mathbf{X} - \Pi\mathbf{X}\|_{2,1}$ and relaxes the orthoprojection constraint to a convex constraint. A prominent advantage of convex optimization techniques is that they are guaranteed to correctly identify outliers under certain conditions. Very recently, several nonconvex outlier detection methods

have also been developed with guaranteed correctness [41,96]. See [97] for a review of more recent development. Nonetheless, these methods typically model a *unique* inlier subspace, e.g., by a low rank matrix $\mathbf{L}$ in Outlier Pursuit, and therefore cannot deal with multiple inlier subspaces since the union of multiple subspaces could be high-dimensional.

Another class of methods with theoretical guarantees for correctness utilizes the fact that outliers are expected to have low similarities with other data points. In [10, 39], a multi-way similarity is introduced that is defined from the polar curvature, which has the advantage of exploiting the subspace structure. However, the number of combinations in multi-way similarity can be prohibitively large. Some recent works have explored using inner products between data points for outlier detection [80, 134]. For example, the coherence pursuit (CoP) [134] claims a point to be an outlier if the sum of its inner products with all other points is small. Although computationally very efficient, these methods require the inliers to be well distributed and densely sampled within the subspaces.

Prior work has also explored using data self-representation as a tool for outlier detection in a union of subspaces. Specifically, motivated by the observation that outliers do not have *sparse* representations, [45, 139] declare a point $\boldsymbol{x}_j$ as an outlier if $\|\boldsymbol{c}_j\|_1$ is above a threshold. However, this ℓ_1-thresholding strategy is not robust to outliers that are close to each other since their representation

vectors may have small ℓ_1-norms. The LRR [107] solves for a low-rank self-representation matrix $\mathbf{C}$ in lieu of a sparse representation and penalizes the sum of unsquared self-representation errors $\|\boldsymbol{x}_j - \mathbf{X}\boldsymbol{c}_j\|_2$, which makes it more robust to outliers. However, LRR requires the subspaces to be independent and the sum of the union of subspaces to be low-dimensional [108].

Finally, we talk about two categories of related outlier detection methods that are not designed for data from low-dimensional subspaces.

Outlier detection by maximum consensus. In a diverse range of contexts such as maximum consensus [42, 201] and robust linear regression [118, 168], people have studied problems of the form

$$\min_{\boldsymbol{b}} \sum_{i=1}^{N} \mathbb{I}(|\boldsymbol{x}_i^\top \boldsymbol{b} - y_i| \geq \epsilon), \tag{5.2}$$

in which $\mathbb{I}(\cdot)$ is the indicator function. Note that if we set $y_i = 1$ for all i, then (5.2) can be interpreted as detecting outliers in data $\mathbf{X}$ where the inliers lie close to an *affine* hyperplane. A problem closely related to (5.2) is

$$\min_{\boldsymbol{b}} \sum_{i=1}^{N} \mathbb{I}(|\boldsymbol{x}_i^\top \boldsymbol{b}| \geq \epsilon) \quad \text{s.t. } \boldsymbol{b} \neq 0, \tag{5.3}$$

which appears in many applications (e.g. see [133]). In particular, (5.3) can be used to learn a *linear* hyperplane from data corrupted by outliers. To detect outliers in a general low-dimensional subspace, one can apply (5.2) and (5.3)

recursively to find a basis for the orthogonal complement of the subspace [154]. However, such an approach is limited because there can be only one inlier subspace and the dimension of that subspace must be known in advance.

Outlier detection by random walk. Perhaps the most well-known random walk based algorithm is PageRank [25]. Originally introduced to determine the authority of website pages from web graphs, PageRank and its variants have been used in different contexts for ranking the centrality of the vertices in a graph. In particular, [121, 122] propose the OutRank, which ranks the "outlierness" of points in a dataset by applying PageRank to an undirected graph in which the weight of an edge is the cosine similarity or RBF similarity between the two connected data points. Then, points that have low centrality are regarded as outliers. The outliers returned by OutRank are those that have low similarity to other data points. Therefore, OutRank does not work if points in a subspace are not dense enough.

5.2 Outlier detection by representation graph

In this section, we present our data self-representation based outlier detection method. We first describe the data self-representation and its associated

properties for inliers and outliers. We then design a random walk algorithm on the representation graph whose limiting behavior allows us to identify the sets of inliers and outliers.

5.2.1 Self-representation of outliers

Given an unlabeled dataset $\mathbf{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N]$ containing inliers and outliers, the first step of our algorithm is to construct the data self-representation matrix denoted by $\mathbf{C} = [\boldsymbol{c}_1, \cdots, \boldsymbol{c}_N]$. As briefly discussed above (see also Figure 5.1), a self-representation matrix $\mathbf{C}$ computed from (5.1) is observed to have different properties for inliers and outliers. Specifically, inliers usually use only other inliers for self-representation, i.e. for an inlier $\boldsymbol{x}_j$, the representation is such that $c_{ij} \neq 0$ only if $\boldsymbol{x}_i$ is also an inlier, where c_{ij} is the (i, j)-th entry of $\mathbf{C}$. This property is expected to hold if the inliers lie in a union of low dimensional subspaces, as evidenced from Chapter 3. As an intuitive explanation, if the inliers lie in a low dimensional subspace, then any inlier has a *sparse* representation using other points in this subspace. Thus such a representation can be found by using sparsity-inducing regularization as seen in (5.1). In contrast, outliers are generally randomly distributed in the ambient space, so that a self-representation usually contains both inliers and outliers.

5.2.2 Representation graph and random walk

We use a directed graph G, which we call a *representation graph*, to capture the behavior of inliers and outliers from the representation matrix $\mathbf{C}$. The vertices of G correspond to the data points $\mathbf{X}$, and the edges are given by the (weighted) adjacency matrix $\mathbf{A} := |\mathbf{C}|^\top \in \mathbb{R}^{N \times N}$ with the absolute value taken elementwise, i.e., the weight of the edge from $\boldsymbol{x}_i$ to $\boldsymbol{x}_j$ is given by $a_{ij} = |c_{ji}|$. In the representation graph, we expect that vertices corresponding to inliers will have edges that only lead to inliers, while vertices that are outliers will have edges that lead to both inliers and outliers. In other words, we do not expect to have any edges that lead from an inlier to an outlier.

Using the previous paragraph as motivation, we design a random walk procedure to identify the outliers. A random walk on the representation graph G is a discrete time Markov chain, for which the transition probability from $\boldsymbol{x}_i$ at a given time to $\boldsymbol{x}_j$ at the next time is given by $p_{ij} := a_{ij}/d_i$ with $d_i := \sum_j a_{ij}$. By this definition, if the starting point of a random walk is an inlier then it will never escape the set of inliers as there is no edge going from any inlier to any outlier. In contrast, a random walk starting from an outlier will likely end up in an inlier state since once it enters any inlier it will never return to an outlier state. Thus, by using different data points to initialize random walks, outliers can be identified by observing the final probability distribution of the state of the random walks (see Figure 5.2).

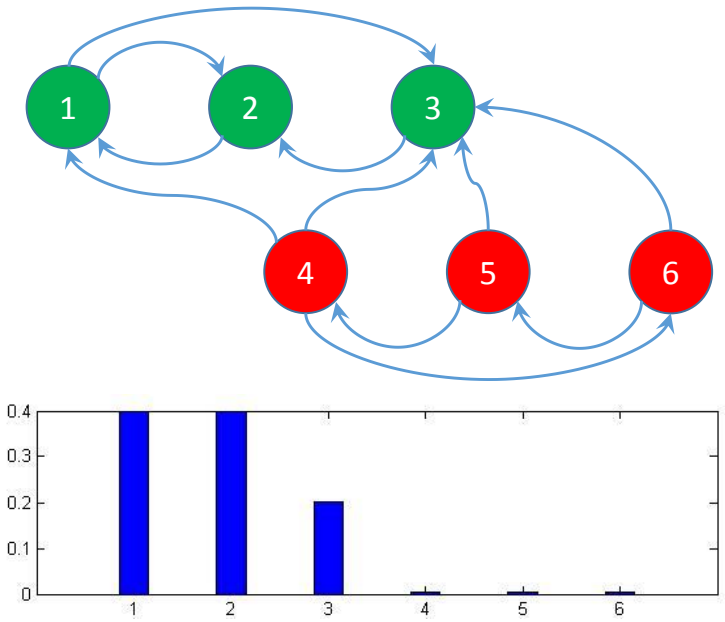


Figure 5.2: Illustration of random walks on a representation graph. Top: green balls represent inliers and red balls represent outliers, and arrows represent edges among nodes. Notice that there is no edge going from inliers to outliers. A random walk starting from any point will end up at only inlier points. Bottom: bar plot of $\bar{\boldsymbol{\pi}}^{(100)}$ with the ith bar corresponding to the ith entry in $\bar{\boldsymbol{\pi}}^{(100)}$. The use of thresholding on this probability distribution will correctly distinguish outliers from inliers.

If $\mathbf{P} \in \mathbb{R}^{N \times N}$ is the transition matrix with entries p_{ij}, then $\mathbf{P}$ is related to the representation matrix $\mathbf{C}$ by

$$p_{ij} = |c_{ji}| / \|\boldsymbol{c}_i\|_1 \quad \text{for all} \quad \{i, j\} \subset \{1, 2, \cdots N\}. \tag{5.4}$$

We define $\boldsymbol{\pi}^{(t)} = [\pi_1^{(t)}, \ldots, \pi_N^{(t)}]$ to be the state probability distribution at time t, then the state transition is given by $\boldsymbol{\pi}^{(t+1)} = \boldsymbol{\pi}^{(t)} P$. Thus, a t-step transition is $\boldsymbol{\pi}^{(t)} = \boldsymbol{\pi}^{(0)} \mathbf{P}^t$ with $\boldsymbol{\pi}^{(0)}$ the chosen initial state probability distribution.

5.2.3 Main algorithm: Outlier detection by R-graph

We propose to perform outlier detection by using random walks on the representation graph G. We set the initial probability distribution as $\boldsymbol{\pi}^{(0)} = [1/N, \cdots, 1/N]$, and then compute the t-step transition $\boldsymbol{\pi}^{(t)} = \boldsymbol{\pi}^{(0)} P^t$. This can be interpreted as initializing a random walk from each of the N data points, and then finding the sum of probability distributions of all random walks after t steps. It is expected that all random walks—starting from either an inlier or an outlier—will eventually have high probabilities for the inlier states and low probabilities for the outlier states.

We note that the $\boldsymbol{\pi}^{(t)}$ defined as above need not converge, as shown by the 2-dimensional example $\mathbf{P} = \left[\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right]$. Instead, we choose to use the T-step Cesàro mean, given by

$$\bar{\boldsymbol{\pi}}^{(T)} = \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{\pi}^{(0)} P^t \equiv \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{\pi}^{(t)}, \tag{5.5}$$

which is the average of the first T t-step probability distributions (see Figure 5.2). The sequence $\{\bar{\boldsymbol{\pi}}^{(T)}\}$ has the benefit that it always converges, and its limit is the same as that of $\boldsymbol{\pi}^{(t)}$ whenever the latter exists. In the next section, we give a more detailed discussion of this choice, its properties for outlier detection, and its convergence behavior.

Our complete algorithm is stated as Algorithm 8.

Algorithm 8 Outlier detection by representation graph

Input: Data $\mathbf{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N]$, number of iterations T, threshold ϵ.

1: Use $\mathbf{X}$ to solve for $\mathbf{C} = [\boldsymbol{c}_1, \cdots, \boldsymbol{c}_N]$ using (5.1).

2: Compute $\mathbf{P}$ from $\mathbf{C}$ using (5.4).

3: Initialize $t = 0$, $\boldsymbol{\pi} = [1/N, \cdots, 1/N]$, and $\bar{\boldsymbol{\pi}} = \mathbf{0}$.

4: **for** $t = 1, 2, \ldots T$ **do**

5: Compute $\boldsymbol{\pi} \leftarrow \boldsymbol{\pi} \cdot P$, and then set $\bar{\boldsymbol{\pi}} \leftarrow \bar{\boldsymbol{\pi}} + \boldsymbol{\pi}$.

6: **end for**

7: $\bar{\boldsymbol{\pi}} \leftarrow \bar{\boldsymbol{\pi}}/T$.

Output: An indicator of outliers: $\boldsymbol{x}_j$ is an outlier if $\bar{\boldsymbol{\pi}}_j \leq \epsilon$.

5.3 Theoretical guarantees for correctness

Let us first formally define the problem of outlier detection when data is drawn from a union of subspaces.

Problem 1 (Outlier detection in a union of subspaces)**.** *Given data matrix* $\mathbf{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N] \in \mathbb{R}^{D \times N}$ *whose columns contain inliers that are drawn from an unknown number of unknown subspaces* $\{\mathcal{S}_\ell\}_{\ell=1}^n$*, and outliers that are outside of* $\cup_{\ell=1}^n \mathcal{S}_\ell$*, the goal is to identify the set of outliers.*

Recall that motivation for our method is that ideally there will be no edge

going from an inlier to an outlier in the representation graph. This motivates us to assume that a random walk starting at any inlier will eventually return to itself, i.e. inliers are *essential states* of the Markov chain, while outliers are those that have a chance of never coming back to itself, i.e. outliers are *inessential states*. Formally, we work with a (time homogeneous) Markov chain with state space $\Omega = \{1, \cdots, N\}$, in which each state j corresponds to data $\boldsymbol{x}_j$, and the transition probability $\mathbf{P}$ is given by (5.4). Given $\{i, j\} \subset \Omega$, we say that j is accessible from i, denoted as $i \to j$, if there exists some $t > 0$ such that the (i, j)-th entry of $\mathbf{P}^t$ is positive. Intuitively, $i \to j$ if a random walk can move from i to j in finitely many steps.

Definition 22 (Essential and inessential state [99])**.** A state $i \in \Omega$ is essential if for all j such that $i \to j$ it is also true that $j \to i$. A state is inessential if it is not essential.

Our aim in this section is to establish that if inliers connect to themselves, i.e. they are *subspace-preserving* (Section 5.3.1), and the representation $\mathbf{C}$ satisfies certain connectivity conditions (Section 5.3.2), then inliers are essential states of the Markov chain and outliers are inessential states. Subsequently, in Section 5.3.3 we show that the Cesàro mean (5.5) identifies essential and inessential states, thus establishing the correctness of Algorithm 8 for outlier detection.

5.3.1 Subspace-preserving representation

We have seen from Chapter 3 that in the context of subspace clustering, the solution to (5.1) has the property that the data points express themselves with only other data points from its own subspace. Such property is called *subspace-preserving* (see Definition 6). In the context of outlier detection in a union of subspaces, we have the following result which states that the solution to (5.1) is subspace-preserving for all inliers.

Theorem 45. *Let $\boldsymbol{x}_j \in \mathcal{S}_\ell$ be an inlier. Define the oracle point of $\boldsymbol{x}_j$ to be $\boldsymbol{\delta}_j := \gamma \cdot (\boldsymbol{x}_j - \mathbf{X}^\ell_{-j} \cdot \boldsymbol{c}^\ell_j)$, where $\mathbf{X}^\ell_{-j}$ is the matrix containing all points in $\mathcal{S}_\ell$ except $\boldsymbol{x}_j$ and*

$$\boldsymbol{c}^\ell_j := \arg\min_{\boldsymbol{c}} \lambda \|\boldsymbol{c}\|_1 + \frac{1-\lambda}{2}\|\boldsymbol{c}\|_2^2 + \frac{\gamma}{2}\|\boldsymbol{x}_j - \mathbf{X}^\ell_{-j}\boldsymbol{c}\|_2^2.$$

The solution $\boldsymbol{c}_j$ to (5.1) *is subspace-preserving if*

$$\max_{k \neq j, \boldsymbol{x}_k \in \mathcal{S}_\ell} |\langle \boldsymbol{x}_k, \bar{\boldsymbol{\delta}}_j \rangle| - \max_{k: \boldsymbol{x}_k \notin \mathcal{S}_\ell} |\langle \boldsymbol{x}_k, \bar{\boldsymbol{\delta}}_j \rangle| > \frac{1-\lambda}{\lambda}, \tag{5.6}$$

where $\bar{\boldsymbol{\delta}}_j := \boldsymbol{\delta}_j / \|\boldsymbol{\delta}_j\|_2$.

This result generalizes Theorem 39 in Chapter 3. Note that the oracle point $\boldsymbol{\delta}_j$ lies in $\mathcal{S}_\ell$ and that its definition only depends on points in $\mathcal{S}_\ell$. The first term in condition (5.6) captures the distribution of points in $\mathcal{S}_\ell$ near $\bar{\boldsymbol{\delta}}_j$, and is expected to be large if the neighborhood of $\bar{\boldsymbol{\delta}}_j$ is well-covered by points from $\mathcal{S}_\ell$.

The second term characterizes the similarity between the oracle point $\bar{\boldsymbol{\delta}}_j$ and all other data points, which includes the outliers and the inliers from other subspaces. The condition requires the former to be larger than the latter by a margin of $\frac{1-\lambda}{\lambda}$, which is close to zero if λ is close to 1. Overall, condition (5.6) requires that points in $\mathcal{S}_\ell$ are dense around $\bar{\boldsymbol{\delta}}_j$, which is itself in $\mathcal{S}_\ell$, and that outliers and inliers from other subspaces do not lie close to $\bar{\boldsymbol{\delta}}_j$.

Even if (5.6) holds for all the inliers in the dataset, we cannot automatically establish an equivalence between inliers/outliers and essential/inessential states because of potential complications related to the graph's *connectivity*. This is addressed next.

5.3.2 Connectivity considerations

Recall that the sparse subspace clustering suffers from the connectivity issue, which states that points in the same subspace may not be well-connected in the representation graph, which may cause oversegmentation of the true clusters. Thus, one has to make the assumption that each true cluster is connected to guarantee correct clustering. For the outlier detection problem, it may happen that an inlier is inessential and thus classified as an outlier when the inliers are not well-connected; similarly, an outlier may be essential and thus classified as an inlier if it is not connected to at least one inlier. In fact, the situation is even more involved since the representation graph is directed

and inliers and outliers behave differently.

Suppose, as a first example, that there exists an inlier that is never used to express any other inliers. This is equivalent to saying that there is no edge going into this point from any other inliers. Note that the subspace-preserving property can still hold if this inlier expresses itself using other inliers. Yet, since a random walk leaving this point would never return it can not be identified as an inlier. To avoid such cases, we need the following assumption.

Assumption 1. *For any inlier subspace $\mathcal{S}_\ell$, the vertices $\{\boldsymbol{x}_j \in \mathcal{S}_\ell\}$ of the representation graph are strongly connected, i.e. there is a path in each direction between each pair of vertices.*

Assumption 1 requires good connectivity between points from the same inlier subspace. We also need good connectivity between outliers and inliers. Consider the example when there is a subset of outliers for which all of their outgoing edges lead only to points within that same subset. In this case, the subset of points can not be detected as outliers since their representation pattern is the same as for the inliers. The next assumption rules out this case.

Assumption 2. *For each subset of outliers there exists an edge in the representation graph that goes from a point in this subset to an inlier or to an outlier outside this subset.*

5.3.3 Main theorem: guaranteed outlier detection

We can now establish guaranteed outlier detection by our representation graph based method stated as Algorithm 8.

Theorem 46. *If the representation $\boldsymbol{c}_j$ is subspace-preserving for each inlier $\boldsymbol{x}_j$ and satisfies Assumptions 1 and 2, then Algorithm 8 with $T = \infty$ and $\epsilon = 0$ correctly identifies outliers.*

Theorem 46 is a direct consequence of the following two facts:

Lemma 19. *If the representation $\boldsymbol{c}_j$ is subspace-preserving for each inlier $\boldsymbol{x}_j$ and Assumptions 1 and 2 hold, then inliers and outliers correspond to essential and inessential states, respectively.*

Proof. Recall that we work with a Markov chain with state space $\Omega = \{1, \cdots, N\}$, in which each state i corresponds to the point $\boldsymbol{x}_i$ in the data matrix $\mathbf{X}$.

First, we show that any inlier point $\boldsymbol{x}_i$ corresponds to an essential state of the Markov chain. Let $\boldsymbol{x}_j$ be any point such that $i \to j$. Since the representation matrix is subspace-preserving, we know that $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ lie in the same subspace. Furthermore, by Assumption 1, all points in the same subspace are strongly connected, which implies that $j \to i$. Thus, i is an essential state.

Second, we show that any outlier point $\boldsymbol{x}_i$ corresponds to an inessential state of the Markov chain. Consider the set $\Omega_i = \{k : i \to k\}$, i.e. the set of

points that are accessible from $\boldsymbol{x}_i$. By Assumption 2, the set Ω_i cannot contain only outliers. Thus, there exists $\boldsymbol{x}_j$ such that $i \to j$ and $\boldsymbol{x}_j$ is an inlier. However, since the representation is subspace-preserving, we know that $j \not\to i$. Therefore, i is not an essential state, i.e. it is an inessential state. □

Lemma 20. *For any probability transition matrix* P*, the averaged probability distribution in* (5.5) *satisfies* $\lim_{T\to\infty} \bar{\boldsymbol{\pi}}^{(T)} = \boldsymbol{\pi}$*, where* $\boldsymbol{\pi}$ *is such that* $\pi_j = 0$ *if and only if state* j *is inessential.*

Proof. According to Theorem 47, the state space of the Markov chain can be decomposed into $\mathbf{I} \cup \mathcal{E}_1 \cup \cdots \cup \mathcal{E}_n$, in which $\mathbf{I}$ contains the inessential states and each $\mathcal{E}_\ell$ is a closed communicating class containing essential states. Assume, without loss of generality, that the transition probability matrix has the form of (5.8). By using (5.13), the Cesàro mean in (5.5) has the following limiting behavior:

$$\begin{aligned}\boldsymbol{\pi} := \lim_{T\to\infty} \bar{\boldsymbol{\pi}}^{(T)} = \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{\pi}_0 P^t \\ = \left[\frac{N_1 + \sum \boldsymbol{f}_{\mathbf{I}\to\mathcal{E}_1}}{N} \cdot \boldsymbol{\pi}_{\mathcal{E}_1}, \cdots, \frac{N_n + \sum \boldsymbol{f}_{\mathbf{I}\to\mathcal{E}_n}}{N} \cdot \boldsymbol{\pi}_{\mathcal{E}_n}, \mathbf{0} \right], \quad (5.7)\end{aligned}$$

where N_ℓ for $\ell = 1, \ldots, n$ is the number of states in class $\mathcal{E}_\ell$, each $\boldsymbol{f}_{\mathbf{I}\to\mathcal{E}_\ell}$ is a vector of hitting probabilities for each state in $\mathbf{I}$ to class $\mathcal{E}_\ell$, and $\boldsymbol{\mu}_{\mathcal{E}_\ell}$ is a positive vector of the stationary distributions of states in $\mathcal{E}_\ell$. Therefore, π_j is zero if and only if j is an inessential state. This finishes the proof. □

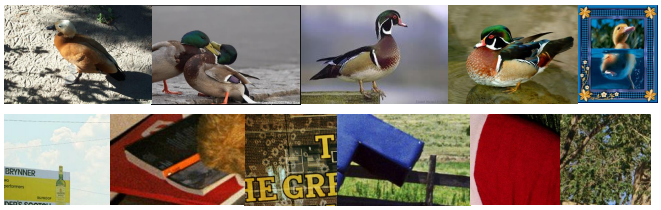
Theorem 46 shows that Problem 1 is solved by Algorithm 8 if the data X satisfies the geometric conditions in (5.6) and the representation graph satisfies the required connectivity assumptions.

We note that the random walk by the Cesàro mean adopted here is different from the popular random walk with restart as adopted by PageRank, for example. The benefit of PageRank is that the random walk converges to the unique stationary distribution. However, it is not clear whether this stationary distribution identifies the outliers. In fact, all states in the random walk of PageRank are essential, so that outliers do not converge to zero probabilities. In contrast, the random walk in our method does not necessarily have a unique stationary distribution, but the Cesàro mean does converge to one of the stationary distributions, which we have shown can be used to identify outliers. A detailed discussion is in the Appendix.

5.4 Experiments

We use several image databases (see Figure 5.3) to evaluate our outlier detection method (Algorithm 8). For computing the representation $\boldsymbol{c}_j$ in (5.1), we use the solver in [86] with $\lambda = 0.95$ and $\gamma = \alpha \cdot \frac{\lambda}{\max_{i:i\neq j} |\boldsymbol{x}_j^\top \boldsymbol{x}_i|}$, where α is a parameter tuned to each dataset. In particular, the solution to (5.1) is nonzero if and only if $\alpha > 1$. The number of iterations T is set to be $1{,}000$.

(a) Extended Yale B



(b) Caltech-256

(c) Coil-100

Figure 5.3: Examples of data used for outlier detection. For each database, the top row shows examples of the inlier set and the bottom row shows examples from the outlier set.

5.4.1 Experimental setup

Databases. We construct outlier detection tasks from three publicly available databases. The Extended Yale B [71] dataset contains frontal face images of 38 individuals each under 64 different illumination conditions. The face images are of size 192×168, for which we downsample to 48×42. The Caltech-256 [76] is a database that contains images from 256 categories that have more than 80 images each. There is also an additional "clutter" category in this database that

contains 827 images of different varieties, which are used as outliers. The Coil-100 dataset [125] contains $7{,}200$ images of 100 different objects. Each object has 72 images taken at pose intervals of 5 degrees, with the images being of size 32×32. For the Extended Yale B and Coil-100 datasets we use raw pixel intensity as the feature representation. Images in Caltech-256 are represented by a $4{,}096$-dimensional feature vector extracted from the last fully connected layer of the 16-layer VGG network [138].

Baselines. We compare with 6 other representative methods that are designed for detecting outliers in one or multiple subspaces: CoP [134], OutlierPursuit (OP) [183], REAPER [98], DPCP [154], LRR [107] and ℓ_1-thresholding (ℓ_1-thr) [139]. We also compare with a graph based method: OutRank (OR) [121, 122]. We implement the inexact ALM [103] for solving the optimization in OutlierPursuit. For LRR, we use the code available online at `https://sites.google.com/site/guangcanliu/`. For DPCP, we use the code provided by the authors. All other methods are implemented according to the description in their respective papers.

Evaluation metric. Each outlier detection method generates a numerical value for each data point that indicates its "outlierness", and a threshold value is required for determining inliers and outliers. A Receiver Operating Characteristic (ROC) curve plots the true positive rate and false positive rate for all threshold values. We use the area under the curve (AUC) as a metric of perfor-

Table 5.1: Results on the Extended Yale B database. Inliers are taken to be the images of either one or three randomly chosen subjects, and outliers are randomly chosen from the other subjects (at most one from each subject). For R-graph we set $\alpha = 5$ in the definition of γ.

	OR	CoP	REAPER	OP	LRR	DPCP	ℓ_1-thr	R-graph
*Inliers: images from **one** subject Outliers:* 35%*, taken from other subjects*								
AUC	0.536	0.556	0.964	0.972	0.857	0.952	0.844	**0.986**
F1	0.552	0.563	0.911	0.918	0.797	0.885	0.763	**0.951**
*Inliers: images from **three** subjects Outliers:* 15%*, taken from other subjects*								
AUC	0.519	0.529	0.932	0.968	0.807	0.888	0.848	**0.985**
F1	0.288	0.292	0.758	0.856	0.509	0.653	0.545	**0.878**

mance in terms of the ROC. The AUC is always between 0 and 1, with a perfect model having an AUC of 1 and a model that guesses randomly having an AUC of approximately 0.5.

As a second metric, we provide the F1-score, which is the harmonic mean of precision and recall. The F1-score is dependent upon the threshold, and we report the largest F1-score across all thresholds. An F1-score of 1 means there exists a threshold that gives both precision and recall equal to 1, i.e. a perfect separation of inliers and outliers.

The reported numbers for all experiments discussed in this section are the averages over 50 trials.

5.4.2 Outliers in face images

Suppose we are given a set of images of one or more individuals but that the data set is also corrupted by face images of a variety of other individuals.

The task is to detect and remove those outlying face images. It is known that images of a face under different lighting conditions lie approximately in a low dimensional subspace. Thus, this task can be modeled as the problem of outlier detection in one subspace or in a union of subspaces.

We use the extended Yale B database. In the first experiment, we randomly choose a single individual from the 38 subjects and use all 64 images of this subject as the inliers. We then choose images from the remaining 37 subjects as outliers with at most one image from each subject. The overall data set has 25% outliers. The average AUC and F1 measures over 50 trials are reported in Table 5.1. For a fair comparison, we fine-tuned the parameters for all methods.

Comparing to state of the art. We see that our representation graph based method R-graph outperforms the other methods. Besides our method, the REAPER, Outlier Pursuit and DPCP algorithms all perform well. These three methods learn a single subspace and treat those that do not fit the subspace as outliers, thus making them well suited for this data (the images of one individual can be well-approximated by a single low dimensional subspace).

The LRR and ℓ_1-thresholding methods use data self-representation, which is also the case for our method. However, LRR does not give good outlier detection results, probably because its algorithm for solving the LRR model is not guaranteed to converge to a global optimum. The ℓ_1-thresholding also does not give good results, showing that the magnitude of the representation vector is

not a robust measure for classifying outliers. By considering the connection patterns in the representation graph, our method achieves significantly better results.

The performance of OutRank and CoP is significantly worse than that of the other methods. This poor performance can be explained by the use of a coherence-based distance, which fails to capture similarity between data points when the data lie in subspaces. For example, it can be argued that the coherence between two faces with the same illumination condition can be higher than two images of the same face under different illumination conditions.

Dealing with multiple inlier groups. In order to test the ability of the methods to deal with multiple inlier groups, we designed a second experiment in which inliers are taken to be images of 3 randomly chosen subjects, and outliers are randomly drawn from other subjects as before. For all methods, we use the same parameters as in the previous experiment to test the robustness to parameter tuning. The results of this experiment are reported in Table 5.1.

We can see that Outlier Pursuit and our R-graph are the two best methods. Although Outlier Pursuit only models a single low dimensional subspace, it can still deal with this data since the union of the three subspaces corresponding to the three subjects in the inlier set is still low dimensional and can be treated as a single low dimensional subspace. However, we postulate that Outlier Pursuit will eventually fail as we increase the number of inlier groups, since the union

Table 5.2: Results on the Caltech-256 database. Inliers are taken to be images of one, three, or five randomly chosen categories, and outliers are randomly chosen from category 257-clutter. For R-graph we set $\alpha = 20$ in the definition of γ.

	OR	CoP	REAPER	OP	LRR	DPCP	ℓ_1-thr	R-graph
Inliers: ***one*** *category of images* *Outliers:* 50%								
AUC	0.897	0.905	0.816	0.837	0.907	0.783	0.772	**0.948**
F1	0.866	0.880	0.808	0.823	0.893	0.785	0.772	**0.914**
Inliers: ***three*** *categories of images* *Outliers:* 50%								
AUC	0.574	0.676	0.796	0.788	0.479	0.798	0.810	**0.929**
F1	0.682	0.718	0.784	0.779	0.671	0.777	0.782	**0.880**
Inliers: ***five*** *categories of images* *Outliers:* 50%								
AUC	0.407	0.487	0.657	0.629	0.337	0.676	0.774	**0.913**
F1	0.667	0.672	0.716	0.711	0.667	0.715	0.762	**0.858**

of low dimensional subspaces will no longer be low rank. Our method does not have this limitation.

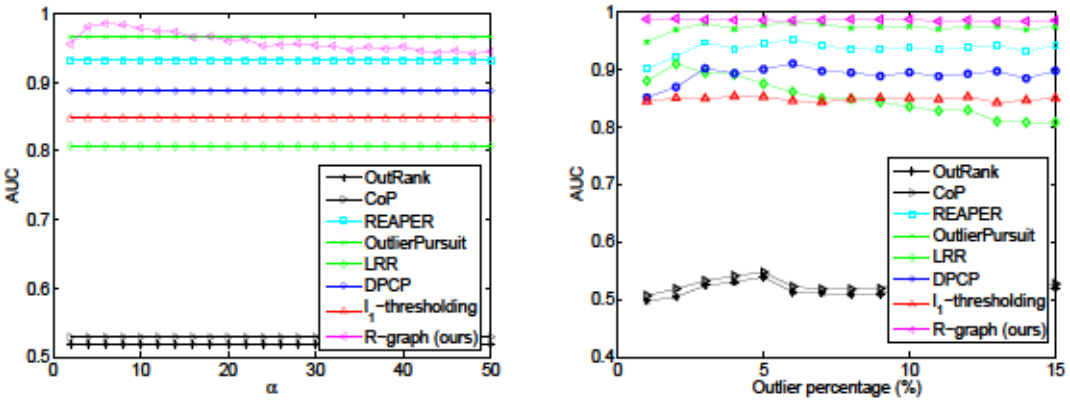
Similar to Outlier Pursuit, both REAPER and DPCP can, in principle, handle multiple inlier groups by fitting a single subspace to their union. However, REAPER and DPCP require as input the dimension of the union of the inlier subspaces, which can be hard to estimate in practice. Indeed, in Table 5.1, we observe that the performances of REAPER and DPCP are less competitive in comparison to Outlier Pursuit and our R-graph for the three subspace case.

Computational time comparison. Table 5.4 reports the average running time of the experiment on the Extended Yale B database with three inlier groups and 15% outliers (226 images in total). From the table we observe that the running times of OutRank and CoP are much smaller than the other methods. This comes from the fact that OutRank and CoP are based on computing

Table 5.3: Results on the Coil-100 database. Inliers are taken to be images of one, four, or seven randomly chosen categories, and outliers are randomly chosen from other categories (at most one from each category). For R-graph we set $\alpha = 10$ in the definition of γ.

	OR	CoP	REAPER	OP	LRR	DPCP	ℓ_1-thr	R-graph
*Inliers: all images from **one** category Outliers:* 50%								
AUC	0.836	0.843	0.900	0.908	0.847	0.900	0.991	**0.997**
F1	0.862	0.866	0.892	0.902	0.872	0.882	0.978	**0.990**
*Inliers: all images from **four** categories Outliers:* 25%								
AUC	0.613	0.628	0.877	0.837	0.687	0.859	0.992	**0.996**
F1	0.491	0.500	0.703	0.686	0.541	0.684	0.941	**0.970**
*Inliers: all images from **seven** categories Outliers:* 15%								
AUC	0.570	0.580	0.824	0.822	0.628	0.804	0.991	**0.996**
F1	0.342	0.346	0.541	0.528	0.366	0.511	0.897	**0.955**

data pairwise inner products, which is efficient for small scale data. In contrast, the other methods solve optimization problems. In particular, REAPER, OutlierPursuit and LRR require computing an eigendecomposition of a matrix of size $D \times D$ (D is the ambient dimension) during each iteration, which is time consuming when D is large. In our experiments we observe that REAPER converges much faster than OutlierPursuit and LRR, thus the running time of REAPER is typically much smaller. The ℓ_1-thresholding method and R-graph method (our algorithm) both compute the representation matrix by solving an ℓ_1 optimization problem for each of the data points with all other data points as the dictionary. Subsequently, ℓ_1-thresholding rejects outliers simply by computing the ℓ_1 norms of the representations, while R-graph requires a random walk on the graph defined from the representation. We note that the random walk for R-graph is computationally efficient because of the sparsity of the represen-



(a) The effect of varying the parameter α.

(b) The effect of varying the outlier percentage.

Figure 5.4: Additional results for experiments on Extended Yale B with three inlier groups and 15% outliers.

tation matrix. In each step of the random walk, the computational complexity is on the order of sN where N is the number of data points and $s \ll N$ is the average number of nonzero entries in the representation vectors $\{c_j\}$.

Table 5.4: Running time of experiments on Extended Yale B data with three inlier groups and 15% outliers.

	OR	CoP	REAPER	OP	LRR	DPCP	ℓ_1-thr	R-graph
Time (sec.)	0.019	0.003	0.079	1.186	3.502	0.182	0.312	0.272

Influence of the algorithm parameters. The first step of our method is to compute the data self-representation matrix using the optimization problem (5.1). We illustrate the effect that the parameter γ in (5.1) has on the performance of our method. Recall that for our numerical experiments we set $\gamma = \alpha \cdot \frac{\lambda}{\max_{i:i\neq j} |\boldsymbol{x}_j^T \boldsymbol{x}_i|}$ and that the solution to (5.1) is nonzero if and only if $\alpha > 1$. We run experiments on Extended Yale B database with 3 inlier groups and 15%

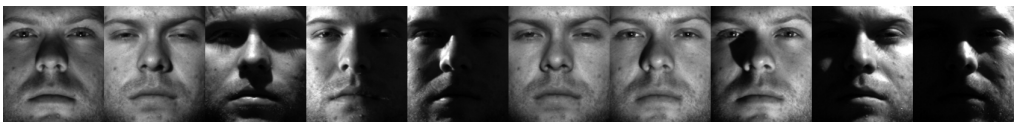
outliers while varying α in the range $[1, 50]$; the results are shown in Figure 5.4a. We can see that the R-graph performs well over a wide range of the parameter α. For comparison, Figure 5.4a also plots the performance of the other methods on the same dataset.

Influence of the percentage of outliers. In this experiment, we fix the number of inlier groups to be 3 and vary the percentage of outliers from 1% to 15%. The performances of the different methods are reported in Figure 5.4b. Note that the parameters for all methods are fixed across the different percentages of outliers. We see that the performance of our method is stable with respect to the percentage of outliers. Moreover, our method also achieves the best performance among all methods.

Visualization of the outliers. To supplement the AUC and F1 measures previously provided, and also to better understand the outliers returned by our outlier detection method, we conducted additional experiments that display the top outliers detected in each experiment. The set of inliers is taken to be the 64 images of the first subject of the Extended Yale B database, and the outlier set is chosen as 10 images randomly chosen from the remaining 37 subjects (see Figure 5.5). The top 10 outliers returned by different methods are reported in Figure 5.6. Images with red boxes are outliers (i.e. true positives) and images with green boxes are inliers (i.e. false positives).

False positives for all methods are mostly images taken under extreme il-

lumination conditions. Such images have large shadows, which has the effect of removing them from the underlying subspace associated with the individual thus making them more likely to be detected as outliers. The results show that REAPER, Outlier Pursuit, DPCP and R-graph are relatively robust. In particular, R-graph is significantly better than ℓ_1-thresholding even though both are sparse representation based methods. This shows that while the magnitude of the representation vector adopted by ℓ_1-thresholding can be sensitive to corruptions, the connectivity behavior explored by R-graph is more robust.



(a) Inliers: 64 images of one individual (displaying 10 out of 64).

(b) Outliers: 10 images from 10 other individuals.

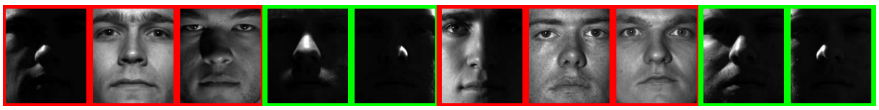
Figure 5.5: An outlier detection dataset for visualizing the top 10 outliers returned by diffferent methods.

5.4.3 Outliers in images of objects

We test the ability of the methods to identify one or several object categories that frequently appear in a set of images amidst outliers that consist of objects that rarely occur. For Caltech-256, images in $n \in \{1, 3, 5\}$ randomly chosen categories are used as inliers in three different experiments. From each cat-

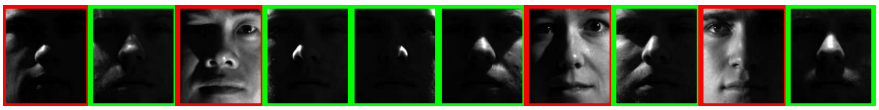
(a) Top 10 outliers by OutRank

(b) Top 10 outliers by CoP



(c) Top 10 outliers by REAPER

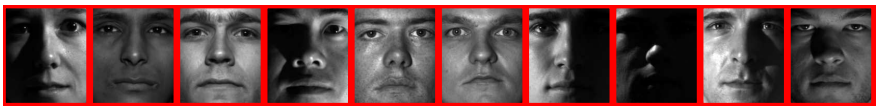
(d) Top 10 outliers by OutlierPursuit



(e) Top 10 outliers by LRR

(f) Top 10 outliers by DPCP

(g) Top 10 outliers by ℓ_1-thresholding



(h) Top 10 outliers by R-graph (ours)

Figure 5.6: Visualizing the top 10 outliers from different methods. Image in red box: true outlier. Image in green box: true inlier.

egory, we use the first 150 images if the category has more than 150 images. We then randomly pick a certain number of images from the "clutter" category as outliers such that there are 50% outliers in each experiment. For Coil-100, we randomly pick $n \in \{1, 4, 7\}$ categories as inliers and pick at most one image from each of the remaining categories as outliers.

The results are reported in Table 5.2 and Table 5.3. We see that our R-graph method achieves the best performance. The two geometric distance based methods, OutRank and CoP, achieve good results when there is one inlier category, but deteriorate when the number of inlier categories increases. The performance of REAPER, Outlier Pursuit and DPCP are similar to each other and worse than our method. This may be because they all try to fit a linear subspace to the data, while the data in these two databases may be better modeled by a nonlinear manifold. The ℓ_1-thresholding and the representation graph method are all based on data self-expression, and seem to be more powerful for this data.

5.5 Appendix: Background on Markov chain theory

We present background material on Markov chain theory that will help us understand the Cesàro mean (5.5) used for outlier detection in our method.

The following material is organized from textbooks [70, 99, 136, 143] and the website `http://www.math.uah.edu/stat`.

We consider a Markov chain $(\mathbf{X}_0, \mathbf{X}_1, \cdots)$ on a finite state space Ω with transition probabilities p_{ij} for $i, j \in \Omega$. The t-step transition probabilities are defined to be $p_{ij}^{(t)} := P\{\mathbf{X}_t = j | \mathbf{X}_0 = i\}$.

5.5.1 Decomposition of the state space

A Markov chain can be decomposed into more basic and manageable parts.

Definition 23. State j is accessible from state i, denoted as $i \to j$, if $p_{ij}^{(t)} > 0$ for some $t > 0$. We say that the states i and j communicate with each other, denoted by $i \leftrightarrow j$, if $i \to j$ and $j \to i$.

Since it can be shown that $\leftrightarrow$ is an equivalence relation, it induces a partition of the state space Ω into disjoint equivalence classes known as *communicating classes*. We are interested in each of the *closed* communicating classes.

Definition 24. A non-empty set $C \subseteq \Omega$ is called a closed set if $p_{ij} = 0$ for $i \in C$ and $j \notin C$.

Note that states in a closed communicating class are essential while states in other communicating classes are inessential [99].

Theorem 47 ([136]). *The state space Ω has the unique decomposition $\Omega = \mathbf{I} \cup \mathcal{E}_1 \cup \ldots \mathcal{E}_n$, where $\mathbf{I}$ is the set of inessential states, and $\mathcal{E}_1, \ldots, \mathcal{E}_n$ are closed communicating classes containing essential states.*

By Theorem 47, the state space of any Markov chain is composed of the essential states and inessential states, and the essential states can be further decomposed into a union of communicating classes. Therefore, the probability transition matrix P can be written in the following form (up to permutation of the states):

$$P = \begin{bmatrix} P_{\mathcal{E}_1 \to \mathcal{E}_1} & & \mathbf{0} & \mathbf{0} \\ & \ddots & & \vdots \\ \mathbf{0} & & P_{\mathcal{E}_n \to \mathcal{E}_n} & \mathbf{0} \\ P_{\mathbf{I} \to \mathcal{E}_1} & \cdots & P_{\mathbf{I} \to \mathcal{E}_n} & P_{\mathbf{I} \to \mathbf{I}} \end{bmatrix} \tag{5.8}$$

5.5.2 Stationary distribution

A nonnegative row vector $\boldsymbol{\pi}$ is called a *stationary distribution* for the Markov chain if it satisfies $\boldsymbol{\pi} = \boldsymbol{\pi} P$.

Theorem 48 ([99, Proposition 1.14, Corollary 1.17]). *A Markov chain consisting of one closed communicating class has a unique stationary distribution. Moreover, each entry of the stationary distribution is positive.*

By Theorem 48, each component $\mathcal{E}_\ell$ for $\ell = 1, \cdots, n$ in the decomposition of the Markov chain in Theorem 47 has a unique positive stationary distribution

$\boldsymbol{\pi}_{\mathcal{E}_\ell}$, i.e.

$$\boldsymbol{\pi}_{\mathcal{E}_\ell} = \boldsymbol{\pi}_{\mathcal{E}_\ell} \cdot P_{\mathcal{E}_\ell \to \mathcal{E}_\ell} \text{ with } \boldsymbol{\pi}_{\mathcal{E}_\ell} > 0 \text{ and } \sum_j (\boldsymbol{\pi}_{\mathcal{E}_\ell})_j = 1. \tag{5.9}$$

We may then define a stationary distribution for $\mathbf{P}$ as

$$[\alpha_1 \boldsymbol{\pi}_{\mathcal{E}_1}, \ldots, \alpha_n \boldsymbol{\pi}_{\mathcal{E}_n}, \mathbf{0}] \text{ for any } \alpha_\ell \geq 0, \sum_{\ell=1}^{n} \alpha_\ell = 1. \tag{5.10}$$

Note that there is not a unique stationary distribution for $\mathbf{P}$ when $n \geq 2$.

5.5.3 Convergence of the Cesàro mean $\frac{1}{T}\sum_{t=1}^{T} P^t$

Let $f_{ij}^{(t)} := P\{\mathbf{X}_t = j, \mathbf{X}_{t'} \neq j \text{ for } 1 \leq t' < t | \mathbf{X}_0 = i\}$ be the probability that the chain starting at i enters j for the first time at the t-th step. The *hitting probability* $f_{ij} = P\{\mathbf{X}_t = j \text{ for some } t > 0 | \mathbf{X}_0 = i\}$ is the probability that the random walk ever makes a transition to state j when started at i, i.e.

$$f_{ij} = \sum_{t=1}^{\infty} f_{ij}^{(t)}. \tag{5.11}$$

The *mean return time* $\mu_j := \sum_{t=1}^{\infty} t f_{jj}^{(t)}$ is the expected time for a random walk starting from state j will return to state j. A general convergence result is stated as follows.

Theorem 49 ([143, Theorem 3.3.1])**.** *For any $i, j \in \Omega$,*

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} p_{ij}^{(t)} = \frac{f_{ij}}{\mu_j}. \tag{5.12}$$

This result can be simplified by using the decomposition in Theorem 47, which leads to the following lemma.

Lemma 21. *If $i, j \in \Omega$ are in the same closed communicating class, then $f_{ij} = f_{ji} = 1$. Also, if $i \in \Omega$ is an inessential state and $\mathcal{E}_\ell \subseteq \Omega$ is a closed communicating class, then $f_{ij} = f_{i\to\mathcal{E}_\ell}$ for all $j \in \mathcal{E}_\ell$, where $f_{i\to\mathcal{E}_\ell}$ is the hitting probability from state i to class $\mathcal{E}_\ell$.*

The following result relates the mean return time with the stationary distribution.

Lemma 22. *For every closed communicating class $\mathcal{E}_\ell \subseteq \Omega$, it holds that $\boldsymbol{\mu}_{\mathcal{E}_\ell} = 1/\boldsymbol{\pi}_{\mathcal{E}_\ell}$ (entry-wise division), where $\boldsymbol{\mu}_{\mathcal{E}_\ell}$ is the vector of mean return times of states in $\mathcal{E}_\ell$. If $i \in \Omega$ is an inessential state, then $\mu_i = \infty$.*

By combining Theorem 49 with Lemma 21 and Lemma 22, the Cesàro limit

of a probability transition matrix of the form in (5.8) can be written as

$$\lim_T \frac{1}{T}\sum_{t=1}^{T} P^t = \begin{bmatrix} \mathbf{1}\cdot\boldsymbol{\pi}_{\mathcal{E}_1} & & \mathbf{0} & \mathbf{0} \\ & \ddots & & \vdots \\ \mathbf{0} & & \mathbf{1}\cdot\boldsymbol{\pi}_{\mathcal{E}_n} & \mathbf{0} \\ \boldsymbol{f}_{\mathbf{I}\to\mathcal{E}_1}\cdot\boldsymbol{\pi}_{\mathcal{E}_1} & \cdots & \boldsymbol{f}_{\mathbf{I}\to\mathcal{E}_1}\cdot\boldsymbol{\pi}_{\mathcal{E}_n} & \mathbf{0} \end{bmatrix}, \tag{5.13}$$

in which $\boldsymbol{f}_{\mathbf{I}\to\mathcal{E}_\ell}$ is a column vector of hitting probability from each state in $\mathbf{I}$ to class $\mathcal{E}_\ell$.

We note that while the Cesàro mean converges, the t-step transition probability $\mathbf{P}^t$ does not necessarily converge. Consider, for example, the probability transition matrix $\mathbf{P} = \left[\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right]$. In this case, $p_{12}^{(t)} = 1$ when t is odd and $p_{12}^{(t)} = 0$ when t is even, i.e. $p_{12}^{(t)}$ is oscillating and never converges. In general, $\mathbf{P}^t$ converges if and only if each of the closed communicating classes $\mathcal{E}_\ell$ for $\ell = 1, \ldots, n$ is *aperiodic*.

5.5.4 Discussion

In this section, we provide additional comments on using the Cesàro mean $\bar{\boldsymbol{\pi}}^{(T)}$ in (5.5) for outlier detection.

Stationary distributions. By (5.7), the vector that $\bar{\boldsymbol{\pi}}^{(T)}$ converges to is a stationary distribution of the Markov chain (see (5.10)). In fact, any convex

combination of the stationary distribution of each closed communicating class is a stationary distribution of the Markov chain, and the particular stationary distribution that $\bar{\boldsymbol{\pi}}^{(T)}$ converges to depends on the choice of the initial state distribution $\boldsymbol{\pi}_0$.

A T-step probability distribution and PageRank. Traditionally, PageRank and many other spectral ranking algorithms use the limit of the T-step probability distribution $\boldsymbol{\pi}^{(T)}$ rather than $\bar{\boldsymbol{\pi}}^{(T)}$ as adopted in our method. However, the sequence $\boldsymbol{\pi}^{(T)}$ converges if and only if each closed communicating class of the Markov chain is aperiodic, which is not necessarily satisfied in many cases. To address this, PageRank adopts a random walk with restart algorithm. It can be interpreted as a random walk on a transformed Markov chain that adds a small probability of transition from each state to the other states on the transition probability of the original Markov chain. By doing so, the transformed Markov chain contains a single communicating class that is aperiodic. Therefore, the stationary distribution necessarily becomes unique, and the sequence $\boldsymbol{\pi}^{(T)}$ for the transformed Markov chain converges to the unique stationary distribution regardless of the initial state distribution.

Despite the advantages of the random walk used by PageRank, all states of the Markov Chain are essential, so that outliers do not converge to zero probabilities. Therefore, it is less clear whether the stationary distribution that the algorithm converges to can effectively identify outliers.

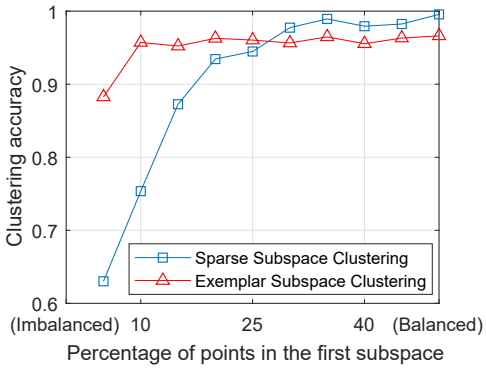
5.6 Conclusion

In this chapter of the thesis, we presented an outlier detection method that combines data self-representation and random walks on a representation graph. Unlike many prior methods for robust PCA, our method is able to deal with multiple subspaces and does not require the number of subspaces or their dimensions to be known. Our analysis showed that the method is guaranteed to identify outliers when certain geometric conditions are satisfied and two connectivity assumptions hold. In our experiments on face image and object image databases, our method achieves the state-of-the-art performance.

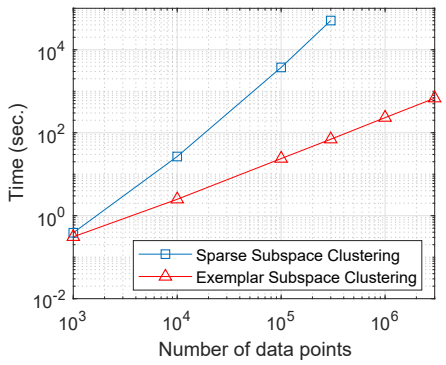
Chapter 6

Subspace Clustering with Imbalanced Data

Despite the great success of sparse subspace clustering (SSC) and elastic-net subspace clustering (EnSC) in their theoretical justifications and empirical performance, previous experimental evaluations focused primarily on existing labeled databases which are usually balanced, i.e. there are approximately equal number of samples from each cluster. In practice, the number of data samples in unlabeled datasets usually varies widely for different classes, and and the effect of such an imbalance on clustering performance has scarcely been studied. Recall that in SSC-BP, each data point is expressed as a sparse linear combination of other data points by solving the following optimization



(a) Clustering on imbalanced data.



(b) Clustering on large-scale data.

Figure 6.1: Subspace clustering on imbalanced data and large-scale data. (a) x and $100 - x$ points (x is varied in the x-axis) are drawn uniformly at random from 2 subspaces of dimension 3 drawn uniformly at random in an ambient space of dimension 5. Note that the clustering accuracy of SSC decreases dramatically as the dataset becomes imbalanced. (b) 10 subspaces of dimension 5 are drawn uniformly at random in an ambient space of dimension 20. An equal number of points is drawn uniformly at random from each subspace. Note that the runtime of SSC increases dramatically with data size.

problem:

$$\min_{\boldsymbol{c}_j \in \mathbb{R}^N} \|\boldsymbol{c}_j\|_1 + \frac{\lambda}{2} \cdot \|\boldsymbol{x}_j - \sum_{i \neq j} c_{ij} \boldsymbol{x}_i\|_2^2, \tag{6.1}$$

where $\lambda > 0$ and $\boldsymbol{c}_j = [c_{1j}, \cdots, c_{Nj}]^\top$. Theoretically, we conjecture that such representation for data point $\boldsymbol{x}_j$ in an under-represented class is more likely to have nonzero entries corresponding to data points in over-represented classes, which gives false connections in the graph affinity. A proof of this conjecture will be the subject of future work. As a preliminary experiment, Figure 6.1a shows that skewed data distribution can significantly compromise the performance of SSC. Another issue with SSC is that it is limited to small or medium scale datasets. Figure 6.1b illustrates the running time of SSC as a function of

the number of data points N, which is roughly quadratic in N.

In this chapter, we present an exemplar-based subspace clustering approach to address the issues of imbalanced and large-scale data. Given a dataset $\mathcal{X}$, the idea is to select a subset $\mathcal{X}_0$, which we call *exemplars*, and write each data point as a linear combination of points in $\mathcal{X}_0$ (rather than $\mathcal{X}$ as in SSC):

$$\min_{\boldsymbol{c}_j \in \mathbb{R}^N} \|\boldsymbol{c}_j\|_1 + \frac{\lambda}{2}\|\boldsymbol{x}_j - \sum_{i:\boldsymbol{x}_i \in \mathcal{X}_0} c_{ij}\boldsymbol{x}_i\|_2^2. \tag{6.2}$$

Observe that (6.2) is potentially more robust to imbalanced data than (6.1) in finding subspace-preserving representations when $\mathcal{X}_0$ is balanced across classes. Moreover, (6.2) can potentially be solved more efficiently than (6.1) when $\mathcal{X}_0$ is small relative to the original data $\mathcal{X}$. Thus, to achieve robustness to imbalanced data and scalability to large datasets, we need an efficient algorithm for selecting exemplars $\mathcal{X}_0$ that is more balanced across classes.

In this chapter, we present a new model for selecting a set of exemplars $\mathcal{X}_0$ that is based on minimizing a maximum representation cost of the data $\mathcal{X}$. Moreover, we introduce an efficient algorithm for solving the optimization problem that has linear time and memory complexity. Compared to SSC, exemplar-based subspace clustering is less sensitive to imbalanced data and more efficient for big data (see Figure 6.1). In particular, we prove that when the data lies in a union of independent subspaces, our method is guaranteed to

select sufficiently many data points from each subspace and construct correct data affinities, even when the data is imbalanced.

6.1 Related work

Sparse dictionary learning (SDL). Sparse representation of a given dataset is a well studied problem in signal processing and machine learning [3, 13]. Given a set $\mathcal{X} \subseteq \mathbb{R}^D$ and an integer k, SDL computes a dictionary of atoms $\mathcal{D} \subseteq \mathbb{R}^D$ with $|\mathcal{D}| \leq k$ that minimizes the sparse representation cost. Based on SDL, [2] proposed a linear time subspace clustering algorithm that is guaranteed to be correct if the atoms in dictionary $\mathcal{D}$ lie in the same union of subspaces as the input data $\mathcal{X}$. However, there is little evidence that such a condition is satisfied in real data as the atoms of the dictionary $\mathcal{D}$ are not constrained to be a subset of $\mathcal{X}$. Another recent work [147], which used data-independent random matrices as dictionaries, also suffers from this issue and lacks correctness guarantees.

Sparse dictionary selection. Three variations of the SDL model that explicitly constrain the dictionary atoms to be taken from $\mathcal{X}$ are simultaneous sparse representation [150] and dictionary selection [36, 49], which use greedy algorithms to solve their respective optimization problems, and group sparse representative selection [45,46,58,117,149,167], which uses a convex optimiza-

tion based approach based on group sparsity. In particular, when the data is drawn from a union of independent subspaces, the method in [58] is shown to select a few representatives from each of the subspaces. However, these methods have quadratic complexity in the number of points in $\mathcal{X}$. Moreover, convex optimization based methods are not flexible in selecting a desired number of representatives since the size of the subset cannot be directly controlled by adjusting an algorithm parameter.

Subset selection. Selecting a representative subset of the entire data has been studied in a wide range of contexts such as Determinantal Point Processes [20, 72, 89], Rank Revealing QR [37], Column subset selection [7, 22], separable Nonnegative Matrix Factorization [12, 90], and so on [57]. However, they do not model data as coming from a union of subspaces and there is no evidence that they can select good representatives from such data. Several recent works [1, 4, 5], which use different subset selection methods for subspace clustering, also lack justification that their selected exemplars are representative of the subspaces.

k-centers and k-medoids. The k-centers problem is a data clustering problem studied in theoretical computer science and operations research. Given a set $\mathcal{X}$ and an integer k, the goal is to find a set of centers $\mathcal{X}_0 \subseteq \mathcal{X}$ with $|\mathcal{X}_0| \leq k$ that minimizes the quantity $\max_{\boldsymbol{x} \in \mathcal{X}} d^2(\boldsymbol{x}, \mathcal{X}_0)$, where $d^2(\boldsymbol{x}, \mathcal{X}_0) := \min_{\boldsymbol{v} \in \mathcal{X}_0} \|\boldsymbol{x} - \boldsymbol{v}\|_2^2$ is the squared distance of $\boldsymbol{x}$ to the closest point in $\mathcal{X}_0$. A partition of $\mathcal{X}$ is given

by the closest center to which each point $\boldsymbol{x} \in \mathcal{X}$ belongs. The k-medoids is a variant of k-centers that minimizes the sum of the squared distances, i.e., minimizes $\sum_{\boldsymbol{x} \in \mathcal{X}} d^2(\boldsymbol{x}, \mathcal{X}_0)$ instead of the maximum distance. However, both k-centers and k-medoids model data as concentrating around several cluster centers, and do not generally apply to data lying in a union of subspaces.

6.2 Exemplar-based Subspace Clustering (ESC)

In this section, we present our ESC method for clustering a given set of data points $\mathcal{X} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\}$. We first formulate the model for selecting a subset $\mathcal{X}_0$ of exemplars from $\mathcal{X}$. Since the model is a combinatorial optimization problem, we present an efficient algorithm for solving it approximately. Finally, we describe the procedure for generating the cluster assignments from the exemplars $\mathcal{X}_0$.

6.2.1 Exemplar selection via self-representation cost

Without loss of generality, we assume that all data in $\mathcal{X}$ are normalized to have unit ℓ_2 norm. Recall that in SSC, each data point $\boldsymbol{x}_j \in \mathcal{X}$ is written as

a linear combination of all other data points with coefficient vector $\boldsymbol{c}_j$. While the nonzero entries in each $\boldsymbol{c}_j$ determine a subset of $\mathcal{X}$ that can represent $\boldsymbol{x}_j$ with the minimum ℓ_1-norm on the coefficients, the collection of all $\boldsymbol{x}_j$ often needs the whole dataset $\mathcal{X}$. In ESC, the goal is to find a small subset $\mathcal{X}_0 \subseteq \mathcal{X}$ that represents all data points in $\mathcal{X}$. In particular, the set $\mathcal{X}_0$ should contain exemplars from each subspace such that the solution $\boldsymbol{c}_j$ to (6.2) for each data point $\boldsymbol{x}_j \in \mathcal{X}$ is *subspace-preserving*, i.e. the nonzero entries of $\boldsymbol{c}_j$ correspond to points in the same subspace as $\boldsymbol{x}_j$. In the following, we define a cost function from the optimization in (6.2) and then present our exemplar selection model.

Definition 25 (Self-representation cost function)**.** Given $\mathcal{X} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\} \subseteq \mathbb{R}^D$, we define the self-representation cost function $F_\lambda : 2^{\mathcal{X}} \to \mathbb{R}$ as

$$F_\lambda(\mathcal{X}_0) := \sup_{\boldsymbol{x}_j \in \mathcal{X}} f_\lambda(\boldsymbol{x}_j, \mathcal{X}_0), \quad \text{where} \tag{6.3}$$

$$f_\lambda(\boldsymbol{x}_j, \mathcal{X}_0) := \min_{\boldsymbol{c}_j \in \mathbb{R}^N} \|\boldsymbol{c}_j\|_1 + \frac{\lambda}{2}\|\boldsymbol{x}_j - \sum_{i:\boldsymbol{x}_i \in \mathcal{X}_0} c_{ij}\boldsymbol{x}_i\|_2^2, \tag{6.4}$$

and $\lambda \in (1, \infty)$ is a parameter. By convention, we assume $f_\lambda(\boldsymbol{x}_j, \emptyset) = \frac{\lambda}{2}$ for all $\boldsymbol{x}_j \in \mathcal{X}$, where $\emptyset$ denotes empty set.

Geometrically, $f_\lambda(\boldsymbol{x}, \mathcal{X}_0)$ measures how well data point $\boldsymbol{x} \in \mathcal{X}$ is covered by the subset $\mathcal{X}_0$ (see Section 6.3). The function $f_\lambda(\boldsymbol{x}, \mathcal{X}_0)$ has the following

properties.

Lemma 23. *The function $f_\lambda(\boldsymbol{x}, \cdot)$ is monotone with respect to the partial order defined by set inclusion, i.e., $f_\lambda(\boldsymbol{x}, \mathcal{X}_0') \geq f_\lambda(\boldsymbol{x}, \mathcal{X}_0'')$ for any $\emptyset \subseteq \mathcal{X}_0' \subseteq \mathcal{X}_0'' \subseteq \mathcal{X}$.*

Proof. Consider the optimization problem

$$\boldsymbol{c}' = [c_1', \cdots, c_N']^\top \in \underset{\boldsymbol{c} \in \mathbb{R}^N}{\arg\min} \|\boldsymbol{c}\|_1 + \frac{\lambda}{2}\|\boldsymbol{x} - \sum_{i:\boldsymbol{x}_i \in \mathcal{X}_0'} c_i \boldsymbol{x}_i\|_2^2 \tag{6.5}$$

and

$$\boldsymbol{c}'' = [c_1'', \cdots, c_N'']^\top \in \underset{\boldsymbol{c} \in \mathbb{R}^N}{\arg\min} \|\boldsymbol{c}\|_1 + \frac{\lambda}{2}\|\boldsymbol{x} - \sum_{i:\boldsymbol{x}_i \in \mathcal{X}_0''} c_i \boldsymbol{x}_i\|_2^2. \tag{6.6}$$

From the optimality of $\boldsymbol{c}'$ for the optimization problem (6.5), we know that $c_i' = 0$ for all $i \in \{1, \cdots, N\}$ such that $\boldsymbol{x}_i \notin \mathcal{X}_0'$. Therefore, by using $\mathcal{X}_0' \subseteq \mathcal{X}_0''$ we have

$$f_\lambda(\boldsymbol{x}, \mathcal{X}_0') = \|\boldsymbol{c}'\|_1 + \frac{\lambda}{2}\|\boldsymbol{x} - \sum_{i:\boldsymbol{x}_i \in \mathcal{X}_0'} c_i' \boldsymbol{x}_i\|_2^2 = \|\boldsymbol{c}'\|_1 + \frac{\lambda}{2}\|\boldsymbol{x} - \sum_{i:\boldsymbol{x}_i \in \mathcal{X}_0''} c_i' \boldsymbol{x}_i\|_2^2. \tag{6.7}$$

Furthermore, note that $\boldsymbol{c}'$ is feasible (not necessarily optimal) for the optimization problem in (6.6), which allows us to conclude that

$$\|\boldsymbol{c}'\|_1 + \frac{\lambda}{2}\|\boldsymbol{x} - \sum_{i:\boldsymbol{x}_i \in \mathcal{X}_0''} c_i' \boldsymbol{x}_i\|_2^2 \geq \|\boldsymbol{c}''\|_1 + \frac{\lambda}{2}\|\boldsymbol{x} - \sum_{i:\boldsymbol{x}_i \in \mathcal{X}_0''} c_i'' \boldsymbol{x}_i\|_2^2 = f_\lambda(\boldsymbol{x}, \mathcal{X}_0''). \tag{6.8}$$

Combining (6.7) and (6.8) shows that $f_\lambda(\boldsymbol{x}, \mathcal{X}_0') \geq f_\lambda(\boldsymbol{x}, \mathcal{X}_0'')$, as claimed. □

Lemma 24. *The value of $f_\lambda(\boldsymbol{x}, \mathcal{X}_0)$ lies in $[1 - \frac{1}{2\lambda}, \frac{\lambda}{2}]$. The lower bound is achieved*

if and only if $\boldsymbol{x} \in \mathcal{X}_0$ *or* $-\boldsymbol{x} \in \mathcal{X}_0$*, and the upper bound is achieved when* $\mathcal{X}_0 = \emptyset$*.*

Proof. See Section 6.6.1. □

Observe that if $\mathcal{X}_0$ contains enough exemplars from the subspace containing $\boldsymbol{x}_j$ and the optimal solution $\boldsymbol{c}_j$ to (6.4) is subspace-preserving, then it is expected that $\boldsymbol{c}_j$ will be sparse and that the residual $\boldsymbol{x}_j - X_0\boldsymbol{c}_j$ will be close to zero. This suggests that we should select the subset $\mathcal{X}_0$ such that the value $f_\lambda(\boldsymbol{x}_j, \mathcal{X}_0)$ is small. As the value $F_\lambda(\mathcal{X}_0)$ is achieved by the data point $\boldsymbol{x}_j$ that has the largest value $f(\boldsymbol{x}_j, \mathcal{X}_0)$, we propose to perform exemplar selection by searching for a subset $\mathcal{X}_0^* \subseteq \mathcal{X}$ that minimizes the self-representation cost function, i.e.,

$$\mathcal{X}_0^* = \operatorname*{arg\,min}_{|\mathcal{X}_0| \leq k} F_\lambda(\mathcal{X}_0), \tag{6.9}$$

where $k \in \mathbb{Z}$ is the target number of exemplars. Note that the objective function $F_\lambda(\cdot)$ in (6.9) is monotone according to the following result.

Lemma 25. *For any* $\emptyset \subseteq \mathcal{X}_0' \subseteq \mathcal{X}_0'' \subseteq \mathcal{X}$*, we have* $F_\lambda(\mathcal{X}_0') \geq F_\lambda(\mathcal{X}_0'')$*.*

Proof. Let $\boldsymbol{x}' \in \arg\sup_{\boldsymbol{x} \in \mathcal{X}} f_\lambda(\boldsymbol{x}, \mathcal{X}_0')$ and $\boldsymbol{x}'' \in \arg\sup_{\boldsymbol{x} \in \mathcal{X}} f_\lambda(\boldsymbol{x}, \mathcal{X}_0'')$. We have

$$F_\lambda(\mathcal{X}_0') = f_\lambda(\boldsymbol{x}', \mathcal{X}_0') \geq f_\lambda(\boldsymbol{x}'', \mathcal{X}_0') \geq f_\lambda(\boldsymbol{x}'', \mathcal{X}_0'') = F_\lambda(\mathcal{X}_0''), \tag{6.10}$$

where the first inequality holds because $\boldsymbol{x}'$ is a maximizer of $f_\lambda(\boldsymbol{x}, \mathcal{X}_0')$, and the second inequality follows from the monotonicity of $f_\lambda(\boldsymbol{x}'', \cdot)$ (see Lemma 23). □

Solving the optimization problem (6.9) is NP-hard in general as it requires evaluating $F_\lambda(\mathcal{X}_0)$ for each subset $\mathcal{X}_0$ of size at most k. In the next section, we present an approximate algorithm that is computationally efficient.

6.2.2 A Farthest First Search (FFS) algorithm for ESC

In Algorithm 9 we present an efficient algorithm for approximately solving (6.9). The algorithm progressively grows a candidate subset $\mathcal{X}_0$ (initialized as the empty set) until it reaches the desired size k. At each iteration i, step 3 of the algorithm selects the point $\boldsymbol{x} \in \mathcal{X}$ that is worst represented by the current subset $\mathcal{X}_0^{(i)}$ as measured by $f_\lambda(\boldsymbol{x}, \mathcal{X}_0^{(i)})$. A geometric interpretation of this step is presented in Section 6.3. In particular, it is shown in Lemma 24 that $f_\lambda(\boldsymbol{x}, \mathcal{X}_0^{(i)}) = 1 - \frac{1}{2\lambda}$ for all $\boldsymbol{x} \in \mathcal{X}_0^{(i)}$ and $f_\lambda(\boldsymbol{x}, \mathcal{X}_0^{(i)}) > 1 - \frac{1}{2\lambda}$ if neither $\boldsymbol{x} \in \mathcal{X}_0^{(i)}$ nor $-\boldsymbol{x} \in \mathcal{X}_0^{(i)}$. Thus, $\boldsymbol{x} \notin \mathcal{X}_0^{(i)}$ during every iteration of Algorithm 9.

We also note that the FFS algorithm can be viewed as an extension of the farthest first traversal algorithm (see, e.g. [175]), which is an approximation algorithm for the k-centers problem discussed in Section 6.1.

Efficient implementation. Observe that each iteration of Algorithm 9 requires evaluating $f_\lambda(\boldsymbol{x}, \mathcal{X}_0^{(i)})$ for every $\boldsymbol{x} \in \mathcal{X}$. Therefore, the complexity of Algorithm 9 is linear in the total number of data points N assuming k is fixed

Algorithm 9 Farthest first search (FFS) for exemplar selection

Input: Data $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subseteq \mathbb{R}^D$, parameters $\lambda > 1$ and $k \ll N$.

1: Select $\boldsymbol{x} \in \mathcal{X}$ at random and set $\mathcal{X}_0^{(1)} \leftarrow \{\boldsymbol{x}\}$.

2: **for** $i = 1, \cdots, k-1$ **do**

3: $\quad \mathcal{X}_0^{(i+1)} = \mathcal{X}_0^{(i)} \cup \arg\max_{\boldsymbol{x} \in \mathcal{X}} f_\lambda(\boldsymbol{x}, \mathcal{X}_0^{(i)})$

4: **end for**

Output: $\mathcal{X}_0^{(k)}$

and small. However, computing $f_\lambda(\boldsymbol{x}, \mathcal{X}_0^{(i)})$ itself is not easy as it requires solving a sparse optimization problem. In the following, we introduce an efficient implementation in which we skip the computation of $f_\lambda(\boldsymbol{x}, \mathcal{X}_0^{(i)})$ for some $\boldsymbol{x}$ in each iteration.

The idea underpinning this computational savings is the monotonicity of $f_\lambda(\boldsymbol{x}, \cdot)$ as discussed in Section 6.2.1. That is, for any $\emptyset \subseteq \mathcal{X}_0' \subseteq \mathcal{X}_0'' \subseteq \mathcal{X}$ we have $f_\lambda(\boldsymbol{x}_j, \mathcal{X}_0') \geq f_\lambda(\boldsymbol{x}_j, \mathcal{X}_0'')$. In the FFS algorithm where the set $\mathcal{X}_0^{(i)}$ is progressively increased, this implies that $f_\lambda(\boldsymbol{x}_j, \mathcal{X}_0^{(i)})$ is non-increasing in i. Using this result, our efficient implementation is outlined in Algorithm 10. In step 2 we initialize $b_j = f_\lambda(\boldsymbol{x}_j, \mathcal{X}_0^{(1)})$ for each $j \in \{1, \cdots, N\}$, which is an upper bound for $f_\lambda(\boldsymbol{x}_j, \mathcal{X}_0^{(i)})$ for $i \geq 1$. In each iteration i, our goal is to find a point $\boldsymbol{x} \in \mathcal{X}$ that maximizes $f_\lambda(\boldsymbol{x}, \mathcal{X}_0^{(i)})$. To do this, we first find an ordering $o_1, \cdots, o_N$ of $1, \cdots, N$ such that $b_{o_1} \geq \cdots \geq b_{o_N}$ (step 4). We then compute $f_\lambda(\cdot, \mathcal{X}_0^{(i)})$ sequentially for points in the list $\boldsymbol{x}_{o_1}, \cdots, \boldsymbol{x}_{o_N}$ (step 7) while keeping track of the highest value of $f_\lambda(\cdot, \mathcal{X}_0^{(i)})$ by

Algorithm 10 An efficient implementation of FFS

Input: Data $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subseteq \mathbb{R}^D$, parameters $\lambda > 1$ and k.

Select $\boldsymbol{x} \in \mathcal{X}$ at random and initialize $\mathcal{X}_0^{(1)} \leftarrow \{\boldsymbol{x}\}$.
Compute $b_j = f_\lambda(\boldsymbol{x}_j, \mathcal{X}_0^{(1)})$ for $j = 1, \cdots, N$.
for $i = 1, \cdots, k-1$ **do**
  Let $o_1, \cdots, o_N$ be an ordering of $1, \cdots, N$ such that $b_{o_p} \geq b_{o_q}$ when $p < q$.
  Initialize *max_cost* $= 0$.
  for $j = 1, \cdots, N$ **do**
    Set $b_{o_j} = f_\lambda(\boldsymbol{x}_{o_j}, \mathcal{X}_0^{(i)})$.
    if $b_{o_j} >$ *max_cost* **then**
      Set *max_cost* $= b_{o_j}$, *new_index* $= o_j$.
    end if
    if $j = N$ or *max_cost* $\geq b_{o_{j+1}}$ **then**
      break
    end if
  end for
  $\mathcal{X}_0^{(i+1)} = \mathcal{X}_0^{(i)} \cup \{\boldsymbol{x}_{\textit{new\_index}}\}$.
end for

Output: $\mathcal{X}_0^{(k)}$

the variable *max_cost* (step 9). Once the condition that *max_cost* $\geq b_{o_{j+1}}$ is met (step 11), we can assert that for any $j' > j$ the point $\boldsymbol{x}_{o_{j'}}$ is not a maximizer of $f_\lambda(\boldsymbol{x}, \mathcal{X}_0^{(i)})$. This can be seen from $f_\lambda(\boldsymbol{x}_{o_{j'}}, \mathcal{X}_0^{(i)}) \leq b_{o_{j'}} \leq b_{o_{j+1}} \leq$ *max_cost*, where the first inequality follows from the monotonicity of $f_\lambda(\boldsymbol{x}_{o_{j'}}, \mathcal{X}_0^{(i)})$ as a function of i. Therefore, we can break the loop (step 12) and avoid computing $f_\lambda(\boldsymbol{x}_{o_j}, \mathcal{X}_0^{(i)})$ for the remaining j's.

6.2.3 Generating cluster assignments

After exemplars have been selected by Algorithm 10, we use them to compute a segmentation of $\mathcal{X}$. Specifically, for each $\boldsymbol{x}_j \in \mathcal{X}$ we compute $\boldsymbol{c}_j$ as a solution to the optimization problem (6.2). As we will see in Theorem 51, the vector $\boldsymbol{c}_j$ is expected to be subspace-preserving. As such, for any two points $\{\boldsymbol{x}_i, \boldsymbol{x}_j\} \subseteq \mathcal{X}$, one has $\langle \boldsymbol{c}_i, \boldsymbol{c}_j \rangle \neq 0$ only if $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are from the same subspace.

Using this observation, we use a nearest neighbor approach to compute the segmentation of $\mathcal{X}$ (see Algorithm 11). First, the coefficient vectors $\{\boldsymbol{c}_j\}$ are normalized, i.e., we set $\tilde{\boldsymbol{c}}_j = \boldsymbol{c}_j / \|\boldsymbol{c}_j\|_2$. Then, for each $\tilde{\boldsymbol{c}}_j$ we find t-nearest neighbors with the largest positive inner product with $\tilde{\boldsymbol{c}}_j$. (Although it is natural to use the t largest inner-products in absolute value, that approach did not perform as well in our numerical experiments.) Finally, we compute an affinity matrix from the t-nearest neighbors and apply spectral clustering to get the segmentation.

Algorithm 11 Subspace clustering by ESC-FFS

Input: Data $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subseteq \mathbb{R}^D$, parameters $\lambda > 1$, k and t.

1: Compute $\mathcal{X}_0$ from Algorithm 10, and then compute $\{\boldsymbol{c}_j\}$ from (6.2). Let $\tilde{\boldsymbol{c}}_j = \boldsymbol{c}_j / \|\boldsymbol{c}_j\|_2$.

2: Set $W_{ij} = 1$ if $\tilde{\boldsymbol{c}}_j$ is a t-nearest neighbor of $\tilde{\boldsymbol{c}}_i$ and 0 otherwise; Set $A = W + W^\top$.

3: Apply spectral clustering to A to obtain a segmentation of $\mathcal{X}$.

Output: Segmentation of $\mathcal{X}$.

6.3 Geometric analysis of ESC

In this section, we present a geometric interpretation of the exemplar selection model from Section 6.2.1 and the FFS algorithm from Section 6.2.2, and study their properties in the context of subspace clustering. To simplify the analysis, we assume that the self-representation $\boldsymbol{x}_j = \sum_{i \neq j} c_{ij} \boldsymbol{x}_i$ is strictly enforced by extending (6.4) to $\lambda = \infty$, i.e., we let

$$f_\infty(\boldsymbol{x}, \mathcal{X}_0) = \min_{\boldsymbol{c} \in \mathbb{R}^N} \|\boldsymbol{c}\|_1 \quad \text{s.t. } \boldsymbol{x} = \sum_{i: \boldsymbol{x}_i \in \mathcal{X}_0} c_{ij} \boldsymbol{x}_i. \tag{6.11}$$

By convention, we let $f_\infty(\boldsymbol{x}, \mathcal{X}_0) = \infty$ if the optimization problem is infeasible.

6.3.1 Geometric interpretation

We first provide a geometric interpretation of the exemplars selected by (6.9). Given any $\mathcal{X}_0$, we denote the convex hull of the symmetrized data points in $\mathcal{X}_0$ as $\mathcal{K}_0$, i.e., $\mathcal{K}_0 := \text{conv}(\pm\mathcal{X}_0)$ (see an example in Figure 6.2). Recall from Chapter 3 that the Minkowski functional [160] associated with a set $\mathcal{K}_0$ is given by the following.

Definition 26 (Minkowski functional)**.** The Minkowski functional associated with the set $\mathcal{K}_0 \subseteq \mathbb{R}^D$ is a map $\mathbb{R}^D \to R \cup \{+\infty\}$ given by

$$\|\boldsymbol{x}\|_{\mathcal{K}_0} := \inf\{t > 0 : \boldsymbol{x}/t \in \mathcal{K}_0\}. \tag{6.12}$$

In particular, we define $\|\boldsymbol{x}\|_{\mathcal{K}_0} := \infty$ if the set $\{t > 0 : \boldsymbol{x}/t \in \mathcal{K}_0\}$ is empty.

Our geometric interpretation is characterized by the reciprocal of $\|\boldsymbol{x}\|_{\mathcal{K}_0}$. The Minkowski functional is a norm in $\text{span}(\mathcal{K}_0)$, the space spanned by $\mathcal{K}_0$, and its unit ball is $\mathcal{K}_0$. Thus, for any $\boldsymbol{x} \in \text{span}(\mathcal{K}_0)$, the point $\boldsymbol{x}/\|\boldsymbol{x}\|_{\mathcal{K}_0}$ is the intersection of the ray $\{t\boldsymbol{x} : t \geq 0\}$ and the boundary of $\mathcal{K}_0$. The green and red dots in Figure 6.2 are examples of $\boldsymbol{x}$ and $\boldsymbol{x}/\|\boldsymbol{x}\|_{\mathcal{K}_0}$, respectively. It follows that the quantity $1/\|\boldsymbol{x}\|_{\mathcal{K}_0}$ is the length of the ray $\{t\boldsymbol{x} : t \geq 0\}$ inside the convex hull $\mathcal{K}_0$.

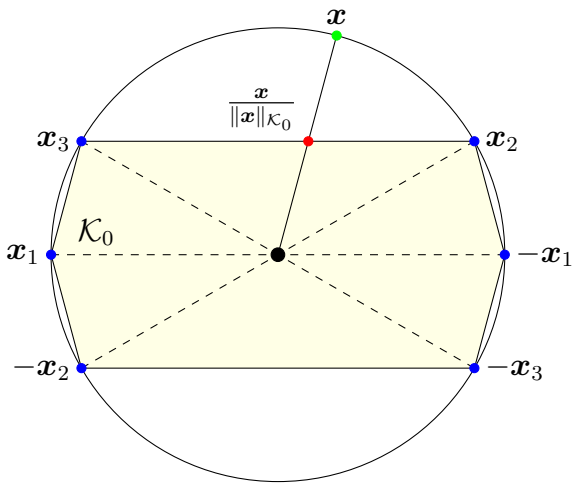


Figure 6.2: A geometric illustration of the solution to (6.9) with $\mathcal{X}_0 = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3\}$. The shaded area is the convex hull $\mathcal{K}_0$.

Using Definition 26, one can show that the following holds [54, 139]:

$$\|\boldsymbol{x}\|_{\mathcal{K}_0} = f_\infty(\boldsymbol{x}, \mathcal{X}_0) \;\text{ for all }\; \boldsymbol{x} \in \mathbb{R}^D. \tag{6.13}$$

A combination of (6.13) and the interpretation of $1/\|\boldsymbol{x}\|_{\mathcal{K}_0}$ above provides a geometric interpretation of $f_\infty(\boldsymbol{x}, \mathcal{X}_0)$. That is, $f_\infty(\boldsymbol{x}, \mathcal{X}_0)$ is large if the length of the ray $\{t\boldsymbol{x} : t \geq 0\}$ inside $\mathcal{K}_0$ is small. In particular, $f_\infty(\boldsymbol{x}, \mathcal{X}_0)$ is infinity if $\boldsymbol{x}$ is not in the span of $\mathcal{X}_0$, i.e., $\boldsymbol{x}$ cannot be linearly represented by $\mathcal{X}_0$.

By using (6.13), the exemplar selection model in (6.9) is equivalent to computing

$$\mathcal{X}_0^* = \operatorname*{arg\,max}_{|\mathcal{X}_0| \leq k} \inf_{\boldsymbol{x} \in \mathcal{X}} 1/\|\boldsymbol{x}\|_{\mathcal{K}_0}. \tag{6.14}$$

Therefore, the solution to (6.9) is the subset $\mathcal{X}_0$ of $\mathcal{X}$ that maximizes the intersection of $\mathcal{K}_0$ and the ray $\{t\boldsymbol{x} : t \geq 0\}$ for every data $\boldsymbol{x} \in \mathcal{X}$ (i.e., maximizes the

minimum of such intersections over all $\boldsymbol{x}$).

Furthermore, from (6.13) we can see that each iteration of Algorithm 9 selects the point $\boldsymbol{x} \in \mathcal{X}$ that minimizes $1/\|\boldsymbol{x}\|_{\mathcal{K}_0}$. Therefore, each iteration of FFS adds the point $\boldsymbol{x}$ that minimizes the intersection of the ray $\{t\boldsymbol{x} : t > 0\}$ with $\mathcal{K}_0$.

Relationship to the sphere covering problem. Let us now consider the special case when the dataset $\mathcal{X}$ coincides with the unit sphere of $\mathbb{R}^D$, i.e., $\mathcal{X} = \mathbb{S}^{D-1}$. In this case, we establish that (6.9) is related to finding the minimum *covering radius*. Recall from Chapter 3 that covering radius is defined as follows.

Definition 27 (Covering radius)**.** The covering radius of a set of points $\mathcal{V} \subseteq \mathbb{S}^{D-1}$ is defined as

$$\gamma(\mathcal{V}) := \max_{\boldsymbol{w} \in \mathbb{S}^{D-1}} \min_{\boldsymbol{v} \in \mathcal{V}} \cos^{-1}(\langle \boldsymbol{v}, \boldsymbol{w} \rangle). \tag{6.15}$$

The covering radius of the set $\mathcal{V}$ can be interpreted as the minimum angle such that the union of spherical caps centered at each point in $\mathcal{V}$ with this radius covers the entire unit sphere $\mathbb{S}^{D-1}$. The following result establishes a relationship between the covering radius and our cost function.

Lemma 26. *For any finite $\mathcal{X}_0 \subseteq \mathcal{X} = \mathbb{S}^{D-1}$ we have $F_\infty(\mathcal{X}_0) = 1/\cos\gamma(\pm\mathcal{X}_0)$.*

Proof. From the definition of $F(\cdot)$ in Definition 25, the relation in (6.13) and

Definition 26 we get

$$F_\infty(\mathcal{X}_0) = \sup_{\boldsymbol{x}\in\mathbb{S}^{D-1}} f_\infty(\boldsymbol{x},\mathcal{X}_0) = \sup_{\boldsymbol{x}\in\mathbb{S}^{D-1}} \|\boldsymbol{x}\|_{\mathcal{K}_0} = \sup_{\boldsymbol{x}\in\mathbb{S}^{D-1}} \inf\{t>0 : \boldsymbol{x}/t \in \mathcal{K}_0\}. \quad (6.16)$$

On the other hand, recall from Chapter 3 that the inradius of a set $\mathcal{K}_0$, denoted by $r(\mathcal{K}_0)$, is defined as the largest Euclidean ball inscribed in $\mathcal{K}_0$. By definition, $r(\mathcal{K}_0)$ can be written as follows.

$$r(\mathcal{K}_0) = \sup\{r>0 : r\boldsymbol{x} \in \mathcal{K}_0, \forall \boldsymbol{x} \in \mathbb{S}^{D-1}\} = \inf_{\boldsymbol{x}\in\mathbb{S}^{D-1}} \sup\{r>0 : r\boldsymbol{x}\in\mathcal{K}_0\}. \quad (6.17)$$

By comparing the right hand side of (6.16) and (6.17) we have

$$F_\infty(\mathcal{X}_0) = 1/r(\mathcal{K}_0). \quad (6.18)$$

The conclusion then follows by combining (6.18) with (3.13). □

It follows from Lemma 26 that $\arg\min_{|\mathcal{X}_0|\le k} F_\infty(\mathcal{X}_0) = \arg\min_{|\mathcal{X}_0|\le k} \gamma(\pm\mathcal{X}_0)$ when $\mathcal{X} = \mathbb{S}^{D-1}$, i.e., the exemplars $\mathcal{X}_0$ selected by (6.9) give the solution to the problem of finding a subset with minimum covering radius. Note that the covering radius $\gamma(\pm\mathcal{X}_0)$ of the subset $\mathcal{X}_0$ with $|\mathcal{X}_0| \le k$ is minimized when the points in the symmetrized set $\pm\mathcal{X}_0$ are as uniformly distributed on the sphere $\mathbb{S}^{D-1}$ as possible. The problem of equally distributing points on the sphere without symmetrizing them, i.e. $\min_{|\mathcal{X}_0|\le k} \gamma(\mathcal{X}_0)$, is known as the sphere covering problem.

This problem was first studied by [146] and remains unsolved in geometry [47].

6.3.2 ESC on a union of subspaces

We now study the properties of our exemplar selection method when applied to data from a union of subspaces. Let $\mathcal{X}$ be drawn from a collection of subspaces $\{\mathcal{S}_\ell\}_{\ell=1}^n$ of dimensions $\{d_\ell\}_{\ell=1}^n$ with each subspace $\mathcal{S}_\ell$ containing at least d_ℓ samples that span $\mathcal{S}_\ell$. We assume that the subspaces are independent, which is commonly used in the analysis of subspace clustering methods [59, 107, 111, 165, 193].

Assumption 3. *The subspaces $\{\mathcal{S}_\ell\}_{\ell=1}^n$ are independent, i.e., $\sum_{\ell=1}^n d_\ell$ is equal to the dimension of $\sum_{\ell=1}^n \mathcal{S}_\ell$.*

The next result shows that the solution to (6.9) contains enough exemplars from each subspace.

Theorem 50. *Under Assumption 3, for all $k \geq \sum_{\ell=1}^n d_\ell$, the solution $\mathcal{X}_0^*$ to the optimization problem in* (6.9) *contains at least d_ℓ linearly independent points from each subspace $\mathcal{S}_\ell$. Moreover, each point $\boldsymbol{x} \in \mathcal{X}$ is expressed as a linear combination of points in $\mathcal{X}_0^*$ that are from its own subspace.*

Theorem 50 shows that when k is set to be $\sum_{\ell=1}^n d_\ell$, then d_ℓ points are selected from subspace $\mathcal{S}_\ell$ regardless of the number of points in that subspace. Therefore, when the data is class imbalanced, (6.9) is able to select a subset

that is more balanced provided that the dimensions of the subspaces do not differ dramatically. This discounts the effect that, when writing a data point as a linear combination of points from $\mathcal{X}$, it is more likely to choose points from oversampled subspaces.

Theorem 50 also shows that only $\sum_{\ell=1}^{n} d_\ell$ points are needed to correctly represent all data points in $\mathcal{X}$. In other words, the required number of exemplars for representing the dataset does not scale with the size of the dataset $\mathcal{X}$.

Before presenting a proof for Theorem 50, we first state the following lemma.

Lemma 27. *Suppose that $\boldsymbol{x} \in \mathcal{S}_\ell$. Under Assumption 3, any optimal solution $\boldsymbol{c}^*$ to the optimization problem in* (6.11) *(if it exists) satisfies $\boldsymbol{x} = \sum_{i:\boldsymbol{x}_i \in \mathcal{X}_0 \cap \mathcal{S}_\ell} c_i^* \boldsymbol{x}_i$ and $\sum_{m \neq \ell} \sum_{i:\boldsymbol{x}_i \in \mathcal{X}_0^* \cap \mathcal{S}_m} |c_i^*| = 0$, i.e., $\boldsymbol{x}$ is expressed as a linear combination of points in $\mathcal{X}_0$ that are from its own subspace.*

Proof. If an optimal solution $\boldsymbol{c}^*$ exists, it must be feasible. Therefore we have

$$\boldsymbol{x} = \sum_{i:\boldsymbol{x}_i \in \mathcal{X}_0} c_i^* \boldsymbol{x}_i = \sum_{i:\boldsymbol{x}_i \in \mathcal{X}_0 \cap \mathcal{S}_\ell} c_i^* \boldsymbol{x}_i + \sum_{m \neq \ell} \sum_{i:\boldsymbol{x}_i \in \mathcal{X}_0 \cap \mathcal{S}_m} c_i^* \boldsymbol{x}_i. \tag{6.19}$$

By rearranging the terms of the equality above, we get

$$\boldsymbol{x} - \sum_{i:\boldsymbol{x}_i \in \mathcal{X}_0 \cap \mathcal{S}_\ell} c_i^* \boldsymbol{x}_i = \sum_{m \neq \ell} \sum_{i:\boldsymbol{x}_i \in \mathcal{X}_0 \cap \mathcal{S}_m} c_i^* \boldsymbol{x}_i. \tag{6.20}$$

In this equality, the left hand side is a vector that lies in $\mathcal{S}_\ell$, while the right

hand side is a vector that lies in $\sum_{m\neq\ell}\mathcal{S}_m$. By Assumption 3 it holds that $\mathcal{S}_\ell \cap \sum_{m\neq\ell}\mathcal{S}_m = \{\mathbf{0}\}$, which then implies that $\boldsymbol{x} = \sum_{i:\boldsymbol{x}_i\in\mathcal{X}_0\cap\mathcal{S}_\ell} c_i^*\boldsymbol{x}_i$.

We can therefore construct another feasible $\hat{\boldsymbol{c}}$ to (6.11) where $\hat{c}_i = c_i^*$ for all $i : \boldsymbol{x}_i \in \mathcal{X}_0^* \cap \mathcal{S}_\ell$ and $\sum_{m\neq\ell}\sum_{i:\boldsymbol{x}_i\in\mathcal{X}_0^*\cap\mathcal{S}_m} |\hat{c}_i| = 0$. By this construction, we have

$$\begin{aligned}\|\hat{\boldsymbol{c}}\|_1 = \sum_{i:\boldsymbol{x}_i\in\mathcal{X}_0^*\cap\mathcal{S}_\ell} |\hat{c}_i| &= \sum_{i:\boldsymbol{x}_i\in\mathcal{X}_0^*\cap\mathcal{S}_\ell} |c_i^*| \\ &\leq \sum_{i:\boldsymbol{x}_i\in\mathcal{X}_0^*\cap\mathcal{S}_\ell} |c_i^*| + \sum_{m\neq\ell}\sum_{i:\boldsymbol{x}_i\in\mathcal{X}_0^*\cap\mathcal{S}_m} |c_i^*| = \|\boldsymbol{c}^*\|_1. \quad (6.21)\end{aligned}$$

On the other hand, by the optimality of $\boldsymbol{c}^*$ as a solution to (6.11) we also have $\|\hat{\boldsymbol{c}}\|_1 \geq \|\boldsymbol{c}^*\|_1$. Combining this result with (6.21) we get $\|\hat{\boldsymbol{c}}\|_1 = \|\boldsymbol{c}^*\|_1$. This further implies that the equality in (6.21) holds, i.e., $\sum_{m\neq\ell}\sum_{i:\boldsymbol{x}_i\in\mathcal{X}_0^*\cap\mathcal{S}_m} |c_i^*| = 0$. □

Proof of Theorem 50. Fix any $k \geq \sum_{\ell=1}^n d_\ell$. There always exists a set $\mathcal{X}_0 \subseteq \mathcal{X}$ with $|\mathcal{X}_0| = k$ that contains d_ℓ linearly independent points from $\mathcal{S}_\ell$ for each $\ell \in \{1, \cdots, n\}$. For this $\mathcal{X}_0$, we have $f_\infty(\boldsymbol{x}, \mathcal{X}_0) < \infty$ for any $\boldsymbol{x} \in \mathcal{X}$, hence $F_\infty(\mathcal{X}_0) < \infty$. This implies that $F_\infty(\mathcal{X}_0^*) \leq F_\infty(\mathcal{X}_0) < \infty$, i.e., $F_\infty(\mathcal{X}_0^*)$ is finite.

We now show that $\mathcal{X}_0^*$ contains at least d_ℓ linearly independent points from each subspace $\mathcal{S}_\ell$. For a proof by contradiction, assume that there exists a subspace, say $\mathcal{S}_\ell$, for which $\mathcal{X}_0^*$ does not contain d_ℓ linearly independent points from $\mathcal{S}_\ell$. This assumption implies that the dimension of $\mathrm{span}(\mathcal{X}_0^* \cap \mathcal{S}_\ell)$ is strictly less than d_ℓ. Consequently, there is a point $\bar{\boldsymbol{x}} \in \mathcal{X} \cap \mathcal{S}_\ell$ such that $\bar{\boldsymbol{x}} \notin \mathrm{span}(\mathcal{X}_0^* \cap$

$\mathcal{S}_\ell$). Now, since $f_\infty(\bar{\boldsymbol{x}}, \mathcal{X}_0^*) \leq F_\infty(\mathcal{X}_0^*) < \infty$, the following problem is feasible:

$$\arg\min_{\boldsymbol{c}} \|\boldsymbol{c}\|_1 \quad \text{s.t. } \bar{\boldsymbol{x}} = \sum_{i:\boldsymbol{x}_i \in \mathcal{X}_0^*} c_i \boldsymbol{x}_i. \tag{6.22}$$

Let $\bar{\boldsymbol{c}}$ be any of the solutions to this optimization problem. Applying Lemma 27 we get $\bar{\boldsymbol{x}} = \sum_{i:\boldsymbol{x}_i \in \mathcal{X}_0^* \cap \mathcal{S}_\ell} \bar{c}_i \boldsymbol{x}_i$, which contradicts the fact that $\bar{\boldsymbol{x}} \notin \text{span}(\mathcal{X}_0^* \cap \mathcal{S}_\ell)$. Therefore, we have proved that $\mathcal{X}_0^*$ contains at least d_ℓ linearly independent points from each subspace $\mathcal{S}_\ell$.

Finally, the claim that each point in $\mathcal{X}$ is expressed as a linear combination of points in $\mathcal{X}_0^*$ that are from its own subspace follows directly from Lemma 27.

□

Although the FFS algorithm in Section 6.2.2 is an approximation algorithm and does not necessarily give the solution to (6.9), the following result shows that it gives an approximate solution with attractive properties for subspace clustering.

Theorem 51. *The conclusion of Theorem 50 holds for $\mathcal{X}_0^{(k)}$ returned by Algorithm 9 provided $k \geq \sum_{\ell=1}^n d_\ell$.*

Proof. Let $\bar{k} = \sum_{\ell=1}^n d_\ell$. To show that the set $\mathcal{X}_0^{(k)}$ contains at least d_ℓ linearly independent points from $\mathcal{S}_\ell$ for each ℓ and each $k \geq \sum_{\ell=1}^n d_\ell$, it suffices to show that the set $\mathcal{X}_0^{(\bar{k})}$ contains d_ℓ linearly independent points from $\mathcal{S}_\ell$ for each ℓ.

For a proof by contradiction, assume that there exists some $\ell \in \{1, \cdots, n\}$

such that the set $\mathcal{X}_0^{(\bar{k})}$ contains linearly dependent points from $\mathcal{S}_\ell$. Then, there exists a $\hat{k} < \bar{k}$ such that the data point that is selected as an exemplar in step $\hat{k}$, denoted as $\boldsymbol{x}^{(\hat{k})} := \arg\max_{\boldsymbol{x}\in\mathcal{X}} f_\infty(\boldsymbol{x}, \mathcal{X}_0^{(\hat{k})})$, lies in the range of $\mathcal{X}_0^{(\hat{k})} \cap \mathcal{S}_\ell$. This implies that $f_\infty(\boldsymbol{x}^{(\hat{k})}, \mathcal{X}_0^{(\hat{k})}) < \infty$ since the optimization is feasible. In addition, note that $F_\infty(\mathcal{X}_0^{(\hat{k})}) = \max_{\boldsymbol{x}\in\mathcal{X}} f_\infty(\boldsymbol{x}, \mathcal{X}_0^{(\hat{k})}) = f_\infty(\boldsymbol{x}^{(\hat{k})}, \mathcal{X}_0^{(\hat{k})})$. Therefore, we have shown that $F_\infty(\mathcal{X}_0^{(\hat{k})}) < \infty$.

On the other hand, since $\hat{k} < \bar{k}$, there exists a subspace $\mathcal{S}_m$ (where m is not necessarily equal to ℓ) such that $\mathcal{X}_0^{(\hat{k})}$ contains less than d_ℓ points from $\mathcal{S}_m$, i.e., $\mathcal{X}_0^{(\hat{k})} \cap \mathcal{S}_m < d_m$. This implies $\operatorname{span}(\mathcal{X}_0^{(\hat{k})} \cap \mathcal{S}_m) \neq \mathcal{S}_m$. Consequently, there is a point $\bar{\boldsymbol{x}} \in \mathcal{X} \cap \mathcal{S}_m$ such that $\bar{\boldsymbol{x}} \notin \operatorname{span}(\mathcal{X}_0^{(\hat{k})} \cap \mathcal{S}_m)$. In addition, from the fact $F_\infty(\mathcal{X}_0^{(\hat{k})}) < \infty$ from the previous paragraph and the relation $f_\infty(\bar{\boldsymbol{x}}, \mathcal{X}_0^{(\hat{k})}) \leq F_\infty(\mathcal{X}_0^{(\hat{k})})$, we get $f_\infty(\bar{\boldsymbol{x}}, \mathcal{X}_0^{(\hat{k})}) < \infty$. Then, it follows from Lemma 27 that $\bar{\boldsymbol{x}} \in \operatorname{span}(\mathcal{X}_0^{(\hat{k})} \cap \mathcal{S}_m)$, which contradicts the fact that $\bar{\boldsymbol{x}} \notin \operatorname{span}(\mathcal{X}_0^{(\hat{k})} \cap \mathcal{S}_m)$. This finishes the proof by contradiction.

Finally, the claim that each point in $\mathcal{X}$ is expressed as a linear combination of points in $\mathcal{X}_0^*$ that are from its own subspace follows directly from Lemma 27.

□

Theorem 51 shows that our algorithm FFS is able to select enough samples from each subspace even if the dataset is imbalanced. It also shows that for each data point in $\mathcal{X}$, the representation vector computed in step 1 of Algorithm 11 is subspace-preserving. Formally, we have established the following

result.

Theorem 52. *Take any $k \geq \sum_{\ell=1}^{n} d_\ell$. Under Assumption 3, the representation vectors $\{\mathbf{c}_j\}_{j=1}^{N}$ in step 1 of Algorithm 11 are subspace-preserving, i.e., c_{ij} is nonzero only if $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are from the same subspace.*

6.4 Experiments

In this section, we demonstrate the performance of ESC for subspace clustering as well as for unsupervised subset selection tasks. The sparse optimization problem (6.4) in step 7 of Algorithm 10 and step 1 of Algorithm 11 are solved by the LASSO version of the LARS algorithm [56] implemented in the SPAMS package [114]. The nearest neighbors in step 2 of Algorithm 11 are computed by the k-d tree algorithm implemented in the VLFeat toolbox [159].

6.4.1 Subspace clustering

We first demonstrate the performance of ESC for subspace clustering on large-scale class-imbalanced databases. These databases are described next.

Databases. We use two publicly available databases. The Extended MNIST (EMNIST) dataset [44] is an extension of the MNIST dataset that contains gray-scale handwritten digits and letters. We take all $190{,}998$ images corresponding to 26 lower case letters, and use them as the data for a 26-class

clustering problem. The size of each image in this dataset is 28 by 28. Following [193], each image is represented by a feature vector computed from a scattering convolutional network [28], which is translational invariant and deformation stable (i.e. it linearizes small deformations). Therefore, these features from EMNIST approximately follow a union of subspaces model.

The German Traffic Sign Recognition Benchmark (GTSRB) [141] contains 43 categories of street sign data with over $50{,}000$ images in total. We remove categories associated with speed limit and triangle-shaped signs (except the yield sign) as they are difficult to distinguish from each other, which results in a final data set of $12{,}390$ images in 14 categories. Each image is represented by a $1{,}568$-dimensional HOG feature [48] provided with the database. The major intra-class variation in GTSRB is the illumination conditions, therefore the data can be well-approximated by a union of subspaces [18].

For both EMNIST and GTSRB, feature vectors are mean subtracted and projected to dimension 500 by PCA and normalized to have unit ℓ_2 norm. Both the EMNIST and GTSRB databases are imbalanced. In EMNIST, for example, the number of images for each letter ranges from $2{,}213$ (letter "j") to $28{,}723$ (letter "e"), and the number of samples for each letter is approximately equal to their frequencies in the English language. In Figure 6.3 we show the number of instances for each class in both of these databases.

Baselines. We compare our approach with SSC-BP to show the effectiveness

of exemplar selection in addressing imbalanced data. To handle large scale data, we use the efficient algorithm in [191] for solving the sparse recovery problem in SSC. For a fair comparison with ESC, we compute an affinity graph for SSC using the same procedure as that used for ESC, i.e., the procedure in Algorithm 11.

We also compare our method with k-means clustering and spectral clustering on the k-nearest neighbors graph, named "Spectral" in the following figures and tables. It is known [80] that Spectral is a provably correct method for subspace clustering. The k-means and k-d trees algorithms used to compute the k-nearest neighbor graph in Spectral are implemented using the VLFeat toolbox [159]. In addition, we compare with three other subspace clustering algorithms SSC-OMP, OLRSC [137] and SBC [2] that are able to handle large-scale data.

We compare these methods with ESC-FFS (Algorithm 11) with λ set to be 150 and 15 for EMNIST and GTSRB, respectively, and t set to be 3 for both databases. We also report the result of ESC-Rand when the exemplars are selected at random from $\mathcal{X}$, i.e., we replace the exemplar selection via FFS in step 1 of Algorithm 11 by selecting k atoms at random from $\mathcal{X}$ to form $\mathcal{X}_0$.

Evaluation metrics. The first metric we use is the clustering accuracy. It measures the maximum proportion of points that are correctly labeled over all possible permutations of the labels. Concretely, let $\{C_1, \cdots, C_n\}$ be the ground-

Figure 6.3: Number of points in each class of EMNIST (left) and GTSRB (right) databases.

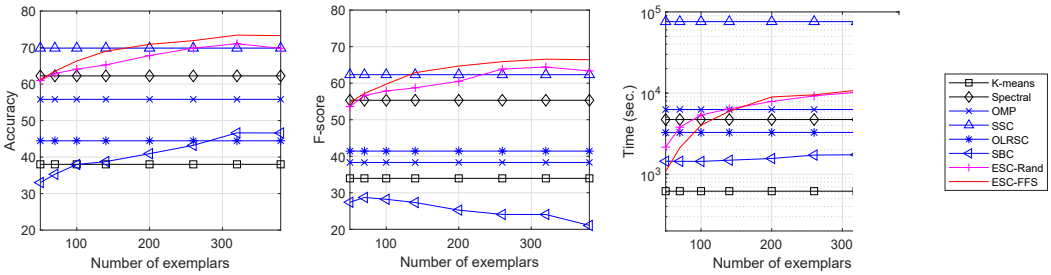
truth partition of the data, $\{G_1, \cdots, G_n\}$ be a clustering result of the same data, $n_{ij} = |C_i \cap G_j|$ be the number of common objects in C_i and G_j, and Π be the set of all permutations of $\{1, \cdots, n\}$. Clustering accuracy is defined as

$$\text{Accuracy} = \max_{\pi \in \Pi} \frac{100}{N} \sum_{i=1}^{n} n_{i,\pi(i)}. \tag{6.23}$$

In the context of classification, accuracy has been known to be biased when the dataset is class imbalanced [26]. For example, if a dataset is composed of 99% of samples from one particular class, then assigning all data points to the same label yields at least 99% accuracy. To address this issue, we also use the F-score averaged over all classes. Let $p_{ij} = n_{ij}/|G_j|$ be the precision and $r_{ij} = n_{ij}/|C_i|$ be the recall. The F-score between the clustering result G_i and the true class C_j is defined as $F_{ij} = \frac{2p_{ij}r_{ij}}{p_{ij}+r_{ij}}$. We report the average F-score given by

$$\text{F-score} = \max_{\pi \in \Pi} \frac{100}{n} \sum_{i=1}^{n} F_{i,\pi(i)}. \tag{6.24}$$

Results on EMNIST. Figure 6.4 shows the results on EMNIST. From left to right, the sub-figures show, respectively, the accuracy, the F-score and the run-



(a) Accuracy **(b)** F-score **(c)** Running time

Figure 6.4: Subspace clustering on images of 26 lower case letters from EMNIST database.

ning time (Y axis) as a function of the number of exemplars (X axis). We can see that ESC-FFS significantly outperforms all methods except SSC in terms of both accuracy and F-score when the number of exemplars is greater than 70.

Recall that in SSC each data point is expressed as a linear combination of all other points. By selecting a subset of exemplars and expressing points using these exemplars, ESC-FFS is able to outperform SSC when the number of exemplars reaches 200. In contrast, ESC-Rand does not outperform SSC by a significant amount, showing the importance of exemplar selection by FFS.

In terms of running time, we see that ESC-FFS is faster than SSC by a large margin. Specifically, ESC-FFS is almost as efficient as ESC-Rand, which indicates that the proposed FFS Algorithm 10 is efficient.

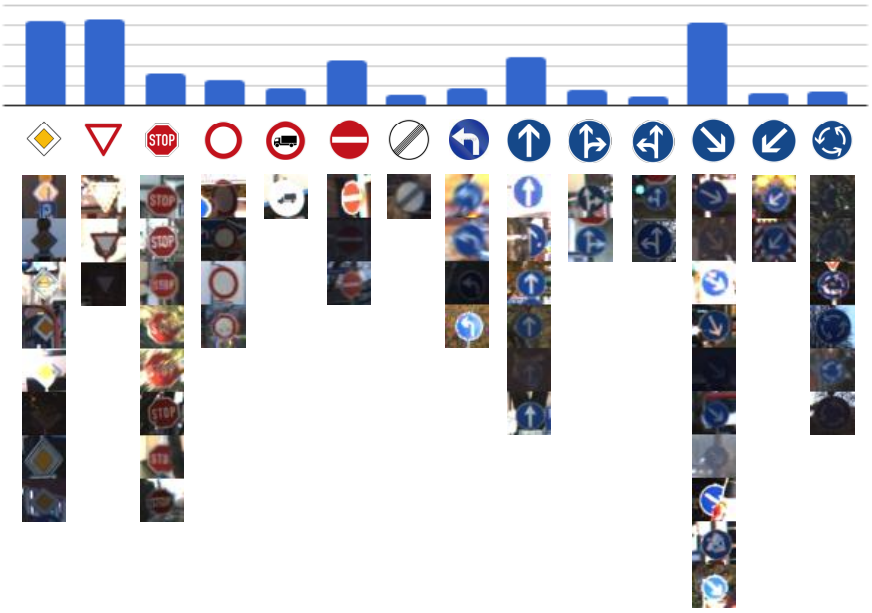
Results on GTSRB. Table 6.1 reports the clustering performance on the GTSRB database. In addition to reporting average performance, we report the standard deviations. The variation in accuracy and F-score across trials is due

to 1) random initializations of the k-means algorithm, which is used (trivially) in the K-means method, and in the spectral clustering step of all other methods, and 2) random dictionary initialization in OLRSC, SBC, ESC-Rand and ESC-FFS.

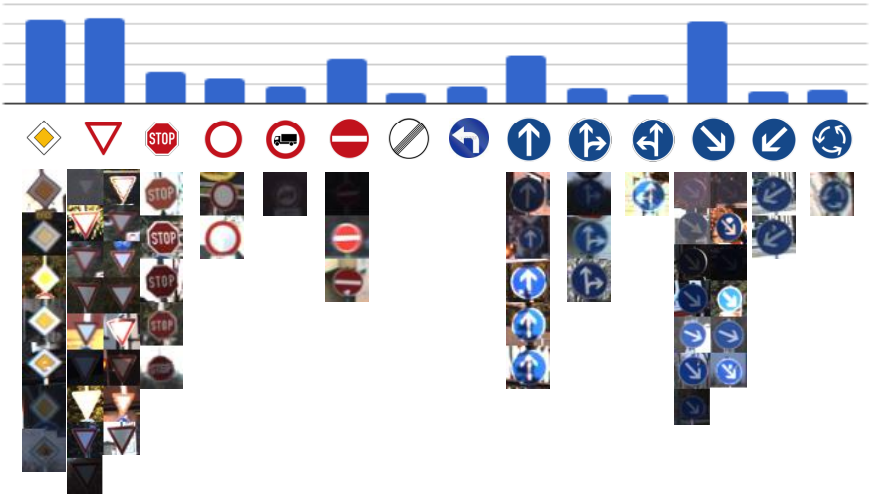
We observe that ESC-FFS outperforms all the other methods in terms of accuracy and F-score. In particular, ESC-FFS outperforms SSC, which in turn outperforms ESC-Rand, thus showing the importance of finding a representative set of exemplars and the effectiveness of FFS in achieving this. In addition, the standard deviation of accuracy and F-score for ESC-Rand are all larger than for ESC-FFS. This indicates that the set of exemplars given by FFS is more robust in giving reliable clustering results than the randomly selected exemplars in ESC-Rand. In terms of running time, ESC-FFS is also competitive.

Table 6.1: Subspace clustering on the GTSRB street sign database. The parameter k is fixed to be 160 for ESC-Rand and ESC-FFS. We report the mean and standard deviation for accuracy, F-score and running time (in sec.) from 10 trials.

Methods	Accuracy	F-score	Time (sec.)
K-means	63.7 ± 3.5	54.4 ± 2.8	$\mathbf{12.2} \pm 0.5$
Spectral	89.5 ± 1.3	79.8 ± 2.5	40.3 ± 0.7
OMP	82.8 ± 0.8	67.8 ± 0.5	22.0 ± 0.2
SSC	92.4 ± 1.1	82.3 ± 2.8	52.2 ± 0.7
OLRSC	71.6 ± 4.3	66.7 ± 4.7	64.9 ± 1.6
SBC	74.9 ± 5.2	72.2 ± 8.5	41.9 ± 0.4
ESC-Rand	89.7 ± 1.6	75.5 ± 4.9	21.5 ± 0.4
ESC-FFS	$\mathbf{93.0} \pm 1.3$	$\mathbf{85.3} \pm 2.5$	25.2 ± 1.2



(a) The 60 exemplars selected by ESC-FFS.



(b) The 60 exemplars selected by ESC-Rand.

Figure 6.5: The exemplars from the GTSRB database selected using ESC-FFS (top) and ESC-Rand (bottom). The frequency of each category within the *entire* database is illustrated via a bar plot above each subfloat.

Visualization of selected exemplars. In Figure 6.5 we show a visualization of the selected exemplars from the GTSRB database. Due to space limitations,

we chose $k = 60$ so that 60 exemplars are selected. The percentage of instances in each category of the dataset is displayed at the top of each subfigure.

From Figure 6.5b we see that the ESC-Rand, which selects exemplars by sampling from the entire dataset uniformly at random, is biased by the imbalanced distribution of the categories in the dataset. For example, the "priority road" sign (the first from the left), the "yield" sign (the second from the left) and the "pass on right side" sign (the third from the right) have more samples in the set of exemplars than the other categories, which is a consequence of these categories having the most samples in the dataset. On the other hand, the seventh and eighth from the left sign types do not have any representatives in the set of exemplars, as they have very few instances in the dataset and therefore have a small chance of being selected by ESC-Rand.

Figure 6.5a shows that ESC-FFS is able to select a more balanced set of exemplars. For example, only 3 instances are selected from the "yield" sign, which is a majority class of the dataset. Moreover, the selected exemplars from each category capture variations in the class, e.g., the 3 exemplars from the "yield" sign category capture three different illumination conditions.

Comparison with additional methods on small scale data. To include a comparison with the recent developed methods ℓ_0-SSC [188] and DD-SSC [182] which cannot handle large scale datasets that we used in our paper, we perform an additional experiment on a small scale dataset that is composed of images

from two categories of the GTSRB database. These two categories have 420 and 2070 images, respectively. For ℓ_0-SSC [188] we use the code provided by the authors. For DD-SSC [182], we use our own implementation following the algorithm description in their paper.

The performance of different methods is reported in Table 6.2. As this dataset is relatively easy, all methods except k-means and DD-SSC produce good clustering results. In terms of runtime, our method is also on par with other scalable subspace clustering methods (i.e. SSC, OLRSC, SBC and OMP), and is orders of magnitude faster than ℓ_0-SSC [188] and DD-SSC [182].

Table 6.2: Subspace clustering on a small subset of the GTSRB street sign database.

Methods	Accuracy	F-score	Time (sec.)
K-means	57.0	54.5	**0.3**
Spectral	95.7	92.3	0.8
OMP	99.2	98.6	0.7
SSC	**99.9**	**99.9**	3.4
OLRSC	**99.9**	**99.9**	1.7
SBC	97.3	94.9	4.3
ℓ_0-SSC	99.6	99.2	602.0
DD-SSC	76.7	67.9	22340.9
ESC-Rand	98.8	97.7	0.5
ESC-FFS	**99.9**	**99.9**	1.1

6.4.2 Unsupervised subset selection

Given a large-scale unlabeled dataset, it is expensive to manually annotate all data. One solution is to select a small subset of data for manual labeling,

and then infer the labels for the remaining data by training a model on the selected subset. In this section, we evaluate the performance of the FFS algorithm as a tool for selecting a subset of representatives for a given dataset. This subset is then subsequently exploited to classify the entire data set.

We use the Extended Yale B face database, which contains images of 38 faces and each of them is taken under 64 different illumination conditions. For this experiment, we create an imbalanced dataset by randomly selecting 10 classes and sampling a subset from each class. The number of images we sample for those 10 classes is 16 for the first 3 classes, 32 for the next 3 classes and 64 for the remaining 4 classes. We first apply FFS to select 100 images from this dataset. Note that during this phase we assume that the ground truth labeling is unknown. We then train three classifiers, the nearest neighbor (NN), sparse representation based classification (SRC) [180] and linear support vector machine (SVM) on the selected images, which is then used to classify all of the images.

We compare FFS with random sampling (Rand), k-centers, K-medoids [130], SMRS [58] and kDPP [89]. For k-centers, we implement the farthest first traversal algorithm (see, e.g. [175]). For K-medoids, we use the function provided by ®Matlab, which employs a variant of the algorithm in [130]. For SMRS and kDPP, we use the code provided by the authors. We set $\lambda = 100$ in FFS.

Table 6.3: Classification from subsets on the Extended Yale B face database. We report the mean and standard deviation for classification accuracy and running time of the subset selection from 50 trials.

Methods	NN	SRC	SVM	Time(sec.)
Rand	69.4 ± 3.2	84.7 ± 2.2	83.7 ± 2.5	$< \mathbf{1e-3}$
k-centers	69.1 ± 3.7	84.9 ± 2.6	83.0 ± 2.8	0.26 ± 0.01
K-medoids	$\mathbf{75.5} \pm 2.8$	86.0 ± 2.1	85.3 ± 2.3	1.5 ± 0.1
kDPP	70.5 ± 3.2	88.3 ± 2.3	87.8 ± 2.1	0.57 ± 0.06
SMRS	69.0 ± 3.1	83.4 ± 2.3	82.1 ± 2.3	3.1 ± 0.2
FFS	67.5 ± 4.0	$\mathbf{91.4} \pm 2.4$	$\mathbf{91.0} \pm 3.0$	0.70 ± 0.08

In Table 6.3 we report the classification accuracy averaged over 50 trials. We can see that the NN classifier works the best with K-medoids, but the performance of NN is worse than SRC and SVM. This is because images of the same face lie approximately in a subspace, and their pairwise distances may not be small. When SRC and SVM are used as classifiers, we can see that our method achieves the best performance.

6.5 Conclusion

We presented a novel approach to subspace clustering for imbalanced and large-scale data. Our method searches for a set of exemplars from the given dataset, such that all data points can be well-represented by the exemplars in terms of a sparse representation cost. Analytically, we showed that the set of exemplars selected by our model has the property that its symmetrized convex hull covers as much of the rays $\{t\boldsymbol{x} : t \geq 0\}$ as possible for all data points

$\boldsymbol{x} \in \mathcal{X}$. In the context of subspace clustering, we proved that our method selects a set of exemplars that is small and balanced, while being able to represent all data points. We also introduced an algorithm for approximately solving the exemplar selection optimization problem. Empirically we demonstrated that our method is effective for subspace clustering and unsupervised subset selection applications.

6.6 Appendix

6.6.1 Proof for Lemma 24

Proof. We divide the proof into two parts.

Part 1. Consider an arbitrary data point $\boldsymbol{x}_j \in \mathcal{X}$. In Part 1 of the proof, we show that $f_\lambda(\boldsymbol{x}_j, \mathcal{X}_0)$ is in the range of $[1 - \frac{1}{2\lambda}, \frac{\lambda}{2}]$, and that the upper bound is achieved, i.e. $f_\lambda(\boldsymbol{x}_j, \mathcal{X}_0) = \frac{\lambda}{2}$ when $\mathcal{X}_0 = \emptyset$. Because the function $f_\lambda(\boldsymbol{x}_j, \cdot)$ is monotone (see Lemma 23), we only need to show that $f_\lambda(\boldsymbol{x}_j, \emptyset) = \lambda/2$ and that $f_\lambda(\boldsymbol{x}_j, \mathcal{X}) = 1 - \frac{1}{2\lambda}$.

First, we have $f_\lambda(\boldsymbol{x}_j, \emptyset) = \lambda/2$ directly from Definition 25.

To show $f_\lambda(\boldsymbol{x}_j, \mathcal{X}) = 1 - \frac{1}{2\lambda}$, let $\boldsymbol{c}_j^* = [c_{1j}^*, \cdots, c_{Nj}^*]$ be any solution to the

optimization problem

$$\arg\min_{\boldsymbol{c}_j} \|\boldsymbol{c}_j\|_1 + \frac{\lambda}{2}\|\boldsymbol{x}_j - \sum_{i=1}^{N} c_{ij}\boldsymbol{x}_i\|_2^2.$$

Let $\boldsymbol{e}_j^* = \boldsymbol{x}_j - \sum_{i=1}^{N} c_{ij}^*\boldsymbol{x}_i$. We have

$$1 = \|\boldsymbol{x}_j\|_2 = \|\boldsymbol{e}_j^* + \sum_{i=1}^{N} c_{ij}^*\boldsymbol{x}_i\|_2 \leq \|\boldsymbol{e}_j^*\|_2 + \sum_{i=1}^{N}(|c_{ij}^*| \cdot \|\boldsymbol{x}_i\|_2) = \|\boldsymbol{e}_j^*\|_2 + \|\boldsymbol{c}_j^*\|_1, \quad (6.25)$$

where we have used the fact that all points in $\mathcal{X}$ have unit ℓ_2 norm. From (6.25), we can derive the following lower bound on $f_\lambda(\boldsymbol{x}_j, \mathcal{X})$:

$$f_\lambda(\boldsymbol{x}_j, \mathcal{X}) = \|\boldsymbol{c}_j^*\|_1 + \frac{\lambda}{2}\|\boldsymbol{e}_j^*\|_2^2 \geq 1 - \|\boldsymbol{e}_j^*\|_2 + \frac{\lambda}{2}\|\boldsymbol{e}_j^*\|_2^2 \geq 1 - \frac{1}{2\lambda}. \quad (6.26)$$

On the other hand, let $\bar{\boldsymbol{c}}_j = [\bar{c}_{1j}, \cdots, \bar{c}_{Nj}]$ be a one-hot vector with the j-th entry, $\bar{c}_{jj}$, being $1 - \frac{1}{\lambda}$ (and all other entries being 0). One can easily verify that $\|\bar{\boldsymbol{c}}_j\|_1 + \frac{\lambda}{2}\|\boldsymbol{x}_j - \sum_{i=1}^{N} \bar{c}_{ij}\boldsymbol{x}_i\|_2^2 = 1 - \frac{1}{2\lambda}$. This shows that the lower bound in (6.26) is achieved by $\bar{\boldsymbol{c}}_j$. Therefore, we have $f_\lambda(\boldsymbol{x}_j, \mathcal{X}) = 1 - \frac{1}{2\lambda}$.

Part 2. In Part 2 we show that the lower bound of $f_\lambda(\boldsymbol{x}_j, \mathcal{X})$ is achieved, i.e. $f_\lambda(\boldsymbol{x}_j, \mathcal{X}_0) = 1 - \frac{1}{2\lambda}$ if and only if $\boldsymbol{x}_j \in \mathcal{X}_0$ or $-\boldsymbol{x}_j \in \mathcal{X}_0$.

For the "if" part, assume that $\boldsymbol{x}_j \in \mathcal{X}_0$ or $\boldsymbol{x}_j \in \mathcal{X}_0$. If $\boldsymbol{x}_j \in \mathcal{X}_0$, we take $\bar{\boldsymbol{c}}_j = [\bar{c}_{1j}, \cdots, \bar{c}_{Nj}]$ as a one-hot vector with the j-th entry being $1 - \frac{1}{\lambda}$. Otherwise, if $-\boldsymbol{x}_j \in \mathcal{X}_0$, we take $\bar{\boldsymbol{c}}_j = [\bar{c}_{1j}, \cdots, \bar{c}_{Nj}]$ to be a one-hot vector with the j-th

entry being $\frac{1}{\lambda} - 1$. In either case, one can easily verify that $\|\bar{\boldsymbol{c}}_j\|_1 + \frac{\lambda}{2}\|\boldsymbol{x}_j - \sum_{i:\boldsymbol{x}_i \in \mathcal{X}_0} \bar{c}_{ij}\boldsymbol{x}_i\|_2^2 = 1 - \frac{1}{2\lambda}$. This implies that $f_\lambda(\boldsymbol{x}_j, \mathcal{X}_0) = 1 - \frac{1}{2\lambda}$.

For the "only if" part, we assume that $f_\lambda(\boldsymbol{x}_j, \mathcal{X}_0) = 1 - \frac{1}{2\lambda}$. Let $\boldsymbol{c}_j^*$ be any solution to the optimization problem

$$\arg\min_{\boldsymbol{c}_j} \|\boldsymbol{c}_j\|_1 + \frac{\lambda}{2}\|\boldsymbol{x}_j - \sum_{i:\boldsymbol{x}_i \in \mathcal{X}_0} c_{ij}\boldsymbol{x}_i\|_2^2,$$

and let $\boldsymbol{e}_j^* = \boldsymbol{x}_j - \sum_{i:\boldsymbol{x}_i \in \mathcal{X}_0} c_{ij}^* \boldsymbol{x}_i$. Note that from the optimality of $\boldsymbol{c}_j^*$, we know that $c_{ij}^* = 0$ for all $i \in \{1, \cdots, N\}$ such that $\boldsymbol{x}_i \notin \mathcal{X}_0$. We have

$$\begin{aligned} 1 = \|\boldsymbol{x}_j\|_2 = \|\boldsymbol{e}_j^* + \sum_{i:\boldsymbol{x}_i \in \mathcal{X}_0} c_{ij}^* \boldsymbol{x}_i\|_2 &\leq \|\boldsymbol{e}_j^*\|_2 + \| \sum_{i:\boldsymbol{x}_i \in \mathcal{X}_0} c_{ij}^* \boldsymbol{x}_i\|_2 \\ &\leq \|\boldsymbol{e}_j^*\|_2 + \sum_{i:\boldsymbol{x}_i \in \mathcal{X}_0} (|c_{ij}^*| \cdot \|\boldsymbol{x}_i\|_2) = \|\boldsymbol{e}_j^*\|_2 + \|\boldsymbol{c}_j^*\|_1. \end{aligned} \tag{6.27}$$

From (6.27), we can establish the following lower bound on $f_\lambda(\boldsymbol{x}_j, \mathcal{X}_0)$:

$$f_\lambda(\boldsymbol{x}_j, \mathcal{X}_0) = \|\boldsymbol{c}_j^*\|_1 + \frac{\lambda}{2}\|\boldsymbol{e}_j^*\|_2^2 \geq 1 - \|\boldsymbol{e}_j^*\|_2 + \frac{\lambda}{2}\|\boldsymbol{e}_j^*\|_2^2 \geq 1 - \frac{1}{2\lambda}. \tag{6.28}$$

Since we have $f_\lambda(\boldsymbol{x}_j, \mathcal{X}_0) = 1 - \frac{1}{2\lambda}$ by assumption, it follows that equality is achieved for all inequalities in (6.28) and (6.27). In particular, by requiring that the equality is achieved for the last inequality in (6.28) we get

$$\|\boldsymbol{e}_j^*\|_2 = \frac{1}{\lambda}. \tag{6.29}$$

Moreover, by requiring that equality is achieved for all inequality in (6.27), we get

$$\| \sum_{i:\boldsymbol{x}_i \in \mathcal{X}_0} c_i^* \boldsymbol{x}_i \|_2 = \| \boldsymbol{c}_j^* \|_1 = 1 - \frac{1}{\lambda}. \tag{6.30}$$

Now, let us define $\mu_0 := \max_{i:\boldsymbol{x}_i \in \mathcal{X}_0} |\langle \boldsymbol{x}_j, \boldsymbol{x}_i \rangle|$. The goal in the rest of the proof is to show that $\mu_0 = 1$. Note that

$$\frac{1}{\lambda^2} = \| \boldsymbol{e}_j^* \|_2^2 = \| \boldsymbol{x}_j - \sum_{i:\boldsymbol{x}_i \in \mathcal{X}_0} c_{ij}^* \boldsymbol{x}_i \|_2^2 = 1 - 2 \cdot \langle \boldsymbol{x}_j, \sum_{i:\boldsymbol{x}_i \in \mathcal{X}_0} c_{ij}^* \boldsymbol{x}_i \rangle + (1 - \frac{1}{\lambda})^2, \tag{6.31}$$

where we have used (6.29) in the first equality and (6.30) in the third equality. For the second term on the right hand side of (6.31), we have

$$\langle \boldsymbol{x}_j, \sum_{i:\boldsymbol{x}_i \in \mathcal{X}_0} c_{ij}^* \boldsymbol{x}_i \rangle \;=\; \sum_{i:\boldsymbol{x}_i \in \mathcal{X}_0} c_{ij}^* \langle \boldsymbol{x}_j, \boldsymbol{x}_i \rangle \;\leq\; \| \boldsymbol{c}_j^* \|_1 \cdot \mu_0 \;\leq\; (1 - \frac{1}{\lambda}) \cdot \mu_0, \tag{6.32}$$

where we have used (6.30) in the last inequality. Continuing with (6.31), we have

$$\begin{aligned} & \frac{1}{\lambda^2} \geq 1 - 2\mu_0 \cdot (1 - \frac{1}{\lambda}) + (1 - \frac{1}{\lambda})^2 \\ \Longrightarrow\; & \frac{1}{\lambda^2} \geq 1 - 2\mu_0 \cdot (1 - \frac{1}{\lambda}) + 1 - \frac{2}{\lambda} + \frac{1}{\lambda^2} \\ \Longrightarrow\; & 0 \geq -2\mu_0 (1 - \frac{1}{\lambda}) + 2(1 - \frac{1}{\lambda}) \\ \Longrightarrow\; & 0 \geq 2 \cdot (1 - \frac{1}{\lambda}) \cdot (1 - \mu_0). \end{aligned} \tag{6.33}$$

Note that λ takes value in the range $(1, \infty)$ (see Definition 25). Therefore, from

(6.33) we get $\mu_0 = \max_{i:\boldsymbol{x}_i \in \mathcal{X}_0} |\langle \boldsymbol{x}_j, \boldsymbol{x}_i \rangle| \geq 1$. Since both $\boldsymbol{x}_j$ and $\boldsymbol{x}_i$ have unit ℓ_2 norm, we can conclude that $\mu_0 = 1$. This implies that there exists $\boldsymbol{x}_i \in \mathcal{X}_0$ such that either $\boldsymbol{x}_j = \boldsymbol{x}_i$ or $\boldsymbol{x}_j = -\boldsymbol{x}_i$.

□

Chapter 7

Conclusions

This thesis attempted to develop theory and algorithms for sparse methods in the applications of subspace classification and subspace clustering.

The first part of this thesis provided an extensive study of the subspace-preserving recovery theory, which extends upon canonical sparse recovery theories established in the area of compressed sensing. By identifying key geometric quantities associated with data in low-dimensional subspaces, we derived conditions for instance and universal subspace-preserving recovery with clear geometric interpretations. By working with a random data modeling, we further derived conditions for instance and universal subspace-preserving recovery that reveal the effect of data parameters such as subspace dimension, ambient space dimension and number of sample points. We showed that these theoretical analysis can be applied to provide justifications for the success of

existing multi-subspace learning methods such as SRC, SSC and EnSC.

The second part of this thesis focused on the development of practical subspace clustering algorithms for real world applications. Since real datasets are usually large-scale, corrupted with outliers, and imbalanced across classes, we developed several algorithmic techniques that can deal with such cases. The effectiveness of our developed techniques are verified in the clustering of several real world image databases.

Bibliography

[1] Maryam Abdolali, Nicolas Gillis, and Mohammad Rahmati. Scalable and robust sparse subspace clustering using randomized clustering and multilayer graphs. *arXiv preprint arXiv:1802.07648*, 2018.

[2] A. Adler, M. Elad, and Y. Hel-Or. Linear-time subspace clustering via bipartite graph modeling. *IEEE Transactions on Neural Networks and Learning Systems*, 26(10):2234 – 2246, 2015.

[3] M. Aharon, M. Elad, and A. M. Bruckstein. K-SVD: an algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, 2006.

[4] Akram Aldroubi, Keaton Hamm, Ahmet Bugra Koku, and Ali Sekmen. Cur decompositions, similarity matrices, and subspace clustering. *arXiv preprint arXiv:1711.04178*, 2017.

[5] Akram Aldroubi, Ali Sekmen, Ahmet Bugra Koku, and Ahmet Faruk

Cakmak. Similarity matrix framework for data from union of subspaces. *Applied and Computational Harmonic Analysis*, 2017.

[6] David Alonso-Gutierrez. On the isotropy constant of random convex sets. *Proceedings of the American Mathematical Society*, 136(9):3293–3300, 2008.

[7] Jason Altschuler, Aditya Bhaskara, Gang Fu, Vahab Mirrokni, Afshin Rostamizadeh, and Morteza Zadimoghaddam. Greedy column subset selection: New bounds and distributed algorithms. In *International Conference on Machine Learning*, pages 2539–2548, 2016.

[8] C. Archambeau, N. Delannay, and M. Verleysen. Mixtures of robust probabilistic principal component analyzers. *Neurocomputing*, 71(7–9):1274–1282, 2008.

[9] Andreas Argyriou, Rina Foygel, and Nathan Srebro. Sparse prediction with the k-support norm. In *Neural Information Processing Systems*, pages 1466–1474, 2012.

[10] E. Arias-Castro, G. Chen, and Gilad Lerman. Spectral clustering based on local linear approximations. *Electron. J. Statist.*, 5:1537–1587, 2011.

[11] Ery Arias-Castro, Gilad Lerman, and Teng Zhang. Spectral cluster-

ing based on local pca. *The Journal of Machine Learning Research*, 18(1):253–309, 2017.

[12] Sanjeev Arora, Rong Ge, Ravindran Kannan, and Ankur Moitra. Computing a nonnegative matrix factorization–provably. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 145–162. ACM, 2012.

[13] F. Bach, R. Jenatton, J. Mairal, and G. Obozinski. Optimization with sparsity-inducing penalties. *Journal Foundations and Trends in Machine Learning*, 4(1):1–106, 2012.

[14] L. Bako. Identification of switched linear systems via sparse optimization. *Automatica*, 47(4):668–677, 2011.

[15] Keith Ball. An elementary introduction to modern convex geometry. In *in Flavors of Geometry*, pages 1–58. Univ. Press, 1997.

[16] Richard Baraniuk. Compressive sensing. *IEEE Signal Processing Magazine*, 24(4):118–121, 2007.

[17] Richard Baraniuk, Mark Davenport, Ronald DeVore, and Michael Wakin. A simple proof of the restricted isometry property for random matrices. *Constructive Approximation*, 28(3):253–263, 2008.

[18] R. Basri and D. Jacobs. Lambertian reflection and linear subspaces.

IEEE Transactions on Pattern Analysis and Machine Intelligence, 25(2):218–233, 2003.

[19] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[20] Alexei Borodin. Determinantal point processes. *arXiv preprint arXiv:0911.1153*, 2009.

[21] T.E. Boult and L.G. Brown. Factorization-based segmentation of motions. In *IEEE Workshop on Motion Understanding*, pages 179–186, 1991.

[22] C. Boutsidis, M. W. Mahoney, and P. Drineas. An improved approximation algorithm for the column subset selection problem. In *Proceedings of SODA*, pages 968–977, 2009.

[23] P. S. Bradley and O. L. Mangasarian. k-plane clustering. *Journal of Global Optimization*, 16(1):23–32, 2000.

[24] S. Brazitikos, A. Giannopoulos, P. Valettas, and B.H. Vritsiou. *Geometry of Isotropic Convex Bodies:*. Mathematical Surveys and Monographs. American Mathematical Society, 2014.

[25] S. Brin and L. Page. The anatomy of a large-scale hypertextual web

search engine. *Computer Networks and ISDN Systems*, 30:107–117, 1998.

[26] Kay Henning Brodersen, Cheng Soon Ong, Klaas Enno Stephan, and Joachim M Buhmann. The balanced accuracy and its posterior distribution. In *Pattern recognition (ICPR), 2010 20th international conference on*, pages 3121–3124. IEEE, 2010.

[27] A.M. Bruckstein, D.L. Donoho, and M. Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM Review*, 51(1):34–81, 2009.

[28] Joan Bruna and Stephane Mallat. Invariant scattering convolution networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1872–1886, August 2013.

[29] D. Burago, Y. Burago, and S. Ivanov. *A Course in Metric Geometry*. Graduate Studies in Mathematics, vol.33. American Mathematical Society, Providence, 2001.

[30] T. Tony Cai and Anru Zhang. Sparse representation of a polytope and recovery of sparse signals and low-rank matrices. *IEEE Transactions on Information Theory*, 60(1):122–132, 2014.

[31] E. Candès. The restricted isometry property and its implications for com-

pressed sensing. *Comptes Rendus Mathematique*, 346(9-10):589–592, 2008.

[32] E. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis. *Journal of the ACM*, 58, 2011.

[33] E. Candès and M. Wakin. An introduction to compressive sampling. *IEEE Signal Processing Magazine*, 25(2):21–30, 2008.

[34] Emmanuel Candès. Compressive sampling. *Proceedings of the International Congress of Mathematicians*, 2006.

[35] Emmanuel Candès and Terence Tao. Decoding by linear programming. *IEEE Trans. on Information Theory*, 51(12):4203–4215, 2005.

[36] Volkan Cevher and Andreas Krause. Greedy dictionary selection for sparse representation. *IEEE Journal of Selected Topics in Signal Processing*, 5(5):979–988, 2011.

[37] T.F. Chan. Rank revealing qr factorizations. *Lin. Alg. and its Appl.*, 88-89:67–82, 1987.

[38] Ling-Hua Chang and Jwo-Yuh Wu. An improved rip-based performance guarantee for sparse signal recovery via orthogonal matching pursuit. *IEEE Transactions on Information Theory*, 60(9):5702–5715, 2014.

[39] G. Chen and G. Lerman. Spectral curvature clustering (SCC). *International Journal of Computer Vision*, 81(3):317–330, 2009.

[40] S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM J. Sci. Comput.*, 20:33–61, 1998.

[41] Yeshwanth Cherapanamjeri, Prateek Jain, and Praneeth Netrapalli. Thresholding based efficient outlier robust pca. *arXiv preprint arXiv:1702.05571*, 2017.

[42] Tat-Jun Chin, Yang Heng Kee, Anders Eriksson, and Frank Neumann. Guaranteed outlier removal with mixed integer linear programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5858–5866, 2016.

[43] F. Chung. Spectral graph theory. In *CBMS Regional Conference Series in Mathematics*, volume 92. American Mathematical Society and Conference Board of the Mathematical Sciences, 1997.

[44] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. Emnist: an extension of mnist to handwritten letters. *arXiv preprint arXiv:1702.05373*, 2017.

[45] Yang Cong, Junsong Yuan, and Ji Liu. Sparse reconstruction cost for abnormal event detection. In *The 24th IEEE Conference on Computer*

Vision and Pattern Recognition, CVPR 2011, Colorado Springs, CO, USA, 20-25 June 2011, pages 3449–3456, 2011.

[46] Yang Cong, Junsong Yuan, and Jiebo Luo. Towards scalable summarization of consumer videos via sparse dictionary selection. *IEEE Transactions on Multimedia*, 14(1):66–75, 2012.

[47] Hallard T Croft, Richard K Guy, and Kenneth J Falconer. *Unsolved problems in geometry*. Springer, 1991.

[48] N Dalal and B Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[49] Abhimanyu Das and David Kempe. Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. *arXiv preprint arXiv:1102.3975*, 2011.

[50] M. A. Davenport and M. B. Wakin. Analysis of orthogonal matching pursuit using the restricted isometry property. *IEEE Transactions on Information Theory*, 56(9):4395–4401, 2010.

[51] Christine De Mol, Ernesto De Vito, and Lorenzo Rosasco. Elastic-net regularization in learning theory. *Journal of Complexity*, 25(2):201–230, 2009.

[52] Chris Ding, Ding Zhou, Xiaofeng He, and Hongyuan Zha. R_1-pca: rotational invariant l_1-norm principal component analysis for robust subspace factorization. In *Proceedings of the 23rd international conference on Machine learning*, pages 281–288. ACM, 2006.

[53] D. L. Donoho and M. Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ_1 minimization. *Proceedings of National Academy of Sciences*, 100(5):2197–2202, 2003.

[54] David L. Donoho. Neighborly polytopes and sparse solution of underdetermined linear equations. *Technical Report, Stanford University*, 2005.

[55] Eva L. Dyer, Aswin C. Sankaranarayanan, and Richard G. Baraniuk. Greedy feature selection for subspace clustering. *Journal of Machine Learning Research*, 14(1):2487–2517, 2013.

[56] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *Annals of Statistics*, 32(2):407–499, 2004.

[57] E. Elhamifar, G. Sapiro, and R. Vidal. Finding exemplars from pairwise dissimilarities via simultaneous sparse recovery. In *Neural Information Processing and Systems*, 2012.

[58] E. Elhamifar, G. Sapiro, and R. Vidal. See all by looking at a few: Sparse

modeling for finding representative objects. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[59] E. Elhamifar and R. Vidal. Sparse subspace clustering. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2790–2797, 2009.

[60] E. Elhamifar and R. Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2765–2781, 2013.

[61] Z. Fan, J. Zhou, and Y. Wu. Multibody grouping by inference of multiple subspaces from high-dimensional data using oriented-frames. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(1):91–105, 2006.

[62] Yuqiang Fang, Ruili Wang, and Bin Dai. Graph-oriented learning via automatic group sparsity for data analysis. In *IEEE International Conference on Data Mining*, pages 251–259, 2012.

[63] Yuqiang Fang, Ruili Wang, Bin Dai, and Xindong Wu. Graph-based learning via auto-grouped sparse regularization and kernelized extension. *IEEE Transactions on Knowledge and Data Engineering*, 27(1):142–154, 2015.

[64] 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*, pages 1801 –1807, 2011.

[65] Miroslav Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czech. Math. J.*, 25:619–633, 1975.

[66] Mario Figueiredo, Robert Nowak, and Stephen Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE Journal of Selected Topics in Signal Processing*, 1(4):586–597, 2007.

[67] 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*, 26:381–395, 1981.

[68] S. Foucart and H. Rauhut. *A Mathematical Introduction to Compressive Sensing*. Applied and Numerical Harmonic Analysis. Springer New York, 2013.

[69] Komei Fukuda. Frequently asked questions in polyhedral computation. 2004.

[70] Robert G Gallager. *Stochastic processes: theory for applications*. Cambridge University Press, 2013.

[71] A.S. Georghiades, P.N. Belhumeur, and D.J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. 23(6):643–660, 2001.

[72] Jennifer A Gillenwater, Alex Kulesza, Emily Fox, and Ben Taskar. Expectation-maximization for learning determinantal point processes. In *NIPS*, pages 3149–3157, 2014.

[73] A. Goh and R. Vidal. Segmenting motions of different types by unsupervised manifold clustering. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–6, 2007.

[74] E. Grave, G. Obozinski, and F. Bach. Trace lasso: a trace norm regularization for correlated designs. In *Neural Information Processing Systems*, 2011.

[75] L. Grenie and G. MOLENI. Inequalities for the beta function. *Math. Inequal. Appl.*, 18(4):1427–1442, 2015.

[76] Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. 2007.

[77] A. Gruber and Y. Weiss. Multibody factorization with uncertainty and

missing data using the EM algorithm. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume I, pages 707–714, 2004.

[78] T. Hastie and P. Simard. Metrics and models for handwritten character recognition. *Statistical Science*, 13(1):54–65, 1998.

[79] Jun He, Yue Zhang, Jiye Wang, Nan Zeng, and Hanyong Hao. Robust k-subspaces recovery with combinatorial initialization. In *IEEE International Conference on Big Data*, pages 3573–3582. IEEE, 2016.

[80] Reinhard Heckel and Helmut Bölcskei. Robust subspace clustering via thresholding. *IEEE Transactions on Information Theory*, 61(11):6320–6342, 2015.

[81] J. Ho, M. H. Yang, J. Lim, K.C. Lee, and D. Kriegman. Clustering appearances of objects under varying illumination conditions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 11–18, 2003.

[82] W. Hong, J. Wright, K. Huang, and Y. Ma. Multi-scale hybrid linear models for lossy image representation. *IEEE Transactions on Image Processing*, 15(12):3655–3671, 2006.

[83] A. Jalali, Y. Chen, S. Sanghavi, and H. Xu. Clustering partially observed graphs via convex optimization. In *International Conference on Machine Learning*, number 1001-1008, 2011.

[84] Pan Ji, Mathieu Salzmann, and Hongdong Li. Efficient dense subspace clustering. In *2014 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 461–468. IEEE, 2014.

[85] Hao Jiang, Daniel P Robinson, René Vidal, and Chong You. A non-convex formulation for low rank subspace clustering: algorithms and convergence analysis. *Computational Optimization and Applications*, 70(2):395–418, 2018.

[86] Bangti Jin, Dirk Lorenz, and Stefan Schiffler. Elastic-net regulariztion: error estimates and active set methods. *Inverse Problems*, 25(11), 2009.

[87] 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*, 1(4):606–617, 2007.

[88] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, 2009.

[89] Alex Kulesza and Ben Taskar. k-dpps: Fixed-size determinantal point processes. In *ICML*, pages 1193–1200, 2011.

[90] Abhishek Kumar, Vikas Sindhwani, and Prabhanjan Kambadur. Fast conical hull algorithms for near-separable non-negative matrix factor-

ization. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 231–239, 2013.

[91] Hanjiang Lai, Yan Pan, Canyi Lu, Yong Tang, and Shuicheng Yan. Efficient k-support matrix pursuit. In *European Conference on Computer Vision*, pages 617–631, 2014.

[92] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278 – 2324, 1998.

[93] Paul Leopardi. A partition of the unit sphere into regions of equal area and small diameter. *Electronic Transactions on Numerical Analysis*, 25(12):309–327, 2006.

[94] PAUL LEOPARDI. Diameter bounds for equal area partitions of the unit sphere. *Electronic Transactions on Numerical Analysis*, 35:1–16, 2009.

[95] G. Lerman and T. Zhang. Robust recovery of multiple subspaces by geometric ℓ_p minimization. *Annals of Statistics*, 39(5):2686–2715, 2011.

[96] Gilad Lerman and Tyler Maunu. Fast, robust and non-convex subspace recovery. *arXiv preprint arXiv:1406.6145*, 2014.

[97] Gilad Lerman and Tyler Maunu. An overview of robust subspace recovery. *arXiv preprint arXiv:1803.01013*, 2018.

[98] Gilad Lerman, Michael B McCoy, Joel A Tropp, and Teng Zhang. Robust computation of linear models by convex relaxation. *Foundations of Computational Mathematics*, 15(2):363–410, 2015.

[99] David Asher Levin, Yuval Peres, and Elizabeth Lee Wilmer. *Markov chains and mixing times*. American Mathematical Soc., 2009.

[100] C.-G. Li, C. You, and R. Vidal. Structured sparse subspace clustering: A joint affinity learning and subspace clustering framework. *IEEE Transactions on Image Processing*, 26(6):2988–3001, 2017.

[101] Jun Li, Yu Kong, and Yun Fu. Sparse subspace clustering by learning approximation ℓ_0 codes. In *Proc. of the AAAI Conf. on Artif. Intell*, pages 2189–2195, 2017.

[102] Xingguo Li and Jarvis Haupt. Identifying outliers in large matrices via randomized adaptive compressive sampling. *IEEE Transactions on Signal Processing*, 63(7):1792–1807, 2015.

[103] Z. Lin, M. Chen, L. Wu, and Yi Ma. The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices. *arXiv:1009.5055v2*, 2011.

[104] Zhouchen Lin, Risheng Liu, and Zhixun Su. Linearized alternating di-

rection method with adaptive penalty for low rank representation. In *Neural Information Processing Systems*, 2011.

[105] John Lipor, David Hong, Dejiao Zhang, and Laura Balzano. Subspace clustering using ensembles of k-subspaces. *arXiv preprint arXiv:1709.04744*, 2017.

[106] G. Liu, Z. Lin, S. Yan, J. Sun, and Y. Ma. Robust recovery of subspace structures by low-rank representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):171–184, 2013.

[107] G. Liu, Z. Lin, and Y. Yu. Robust subspace segmentation by low-rank representation. In *International Conference on Machine Learning*, pages 663–670, 2010.

[108] Guangcan Liu, Huan Xu, and Shuicheng Yan. Exact subspace segmentation and outlier detection by low-rank representation. In *AISTATS*, pages 703–711, 2012.

[109] Guangcan Liu and Shuicheng Yan. Latent low-rank representation for subspace segmentation and feature extraction. In *IEEE International Conference on Computer Vision*, pages 1615–1622, 2011.

[110] C. Lu, Z. Lin, and S. Yan. Correlation adaptive subspace segmentation

by trace lasso. In *IEEE International Conference on Computer Vision*, pages 1345–1352, 2013.

[111] C-Y. Lu, H. Min, Z-Q. Zhao, L. Zhu, D-S. Huang, and S. Yan. Robust and efficient subspace segmentation via least squares regression. In *European Conference on Computer Vision*, pages 347–360, 2012.

[112] Y. Ma, H. Derksen, W. Hong, and J. Wright. Segmentation of multivariate mixed data via lossy coding and compression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(9):1546–1562, 2007.

[113] Y. Ma, A. Y. Yang, H. Derksen, and R. Fossum. Estimation of subspace arrangements with applications in modeling and segmenting mixed data. *SIAM Review*, 50(3):413–458, 2008.

[114] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online learning for matrix factorization and sparse coding. *The Journal of Machine Learning Research*, 11:19–60, 2010.

[115] Michael McCoy, Joel A Tropp, et al. Two proposals for robust pca using semidefinite programming. *Electronic Journal of Statistics*, 5:1123–1160, 2011.

[116] Brian McWilliams and Giovanni Montana. Subspace clustering of high

dimensional data: a predictive approach. *Data Mining and Knowledge Discovery*, 28(3):736–772, 2014.

[117] Jingjing Meng, Hongxing Wang, Junsong Yuan, and Yap-Peng Tan. From keyframes to key objects: Video summarization by representative object proposal selection. In *CVPR*, pages 1039–1048, 2016.

[118] Kaushik Mitra, Ashok Veeraraghavan, and Rama Chellappa. Analysis of sparse regularization based robust regression approaches. *IEEE Transactions on Signal Processing*, 61(5):1249–1257, 2013.

[119] Qun. Mo and Song. Li. New bounds on the restricted isometry constant δ_{2k}. *Applied and Computational Harmonic Analysis*, 31(3):460–468, 2011.

[120] Qun Mo and Yi Shen. A remark on the restricted isometry property in orthogonal matching pursuit. *IEEE Transactions on Information Theory*, 58(6):3654–3656, 2012.

[121] HDK Moonesignhe and Pang-Ning Tan. Outlier detection using random walks. In *2006 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06)*, pages 532–539. IEEE, 2006.

[122] HDK Moonesinghe and Pang-Ning Tan. Outrank: a graph-based out-

lier detection framework using random walk. *International Journal on Artificial Intelligence Tools*, 17(01):19–36, 2008.

[123] B. Nasihatkon and R. Hartley. Graph connectivity in sparse subspace clustering. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2137–2144, 2011.

[124] Balas Kausik Natarajan. Sparse approximate solutions to linear systems. *SIAM journal on computing*, 24(2):227–234, 1995.

[125] S. A. Nene, S. K. Nayar, and H. Murase. Columbia object image library (COIL-100). *Technical Report CUCS-006-96*, 1996.

[126] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization, second edition*. World Scientific, 2006.

[127] Isabella Novik. A tale of centrally symmetric polytopes and spheres. *arXiv:1711.09310*, 2017.

[128] Yannis Panagakis and Constantine Kotropoulos. Elastic net subspace clustering applied to pop/rock music structure analysis. *Pattern Recognition Letters*, 38:46–53, 2014.

[129] Dohyung Park, Constantine Caramanis, and Sujay Sanghavi. Greedy subspace clustering. In *Neural Information Processing Systems*, 2014.

[130] Hae-Sang Park and Chi-Hyuck Jun. A simple and fast algorithm for k-medoids clustering. *Expert systems with applications*, 36(2):3336–3341, 2009.

[131] Y. Pati, R. Rezaiifar, and P. Krishnaprasad. Orthogonal matching pursuit: recursive function approximation with application to wavelet decomposition. In *Asilomar Conference on Signals, Systems and Computation*, 1993.

[132] Xi Peng, Lei Zhang, and Zhang Yi. Scalable sparse subspace clustering. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 430–437, 2013.

[133] Q. Qu, J. Sun, and J. Wright. Finding a sparse vector in a subspace: Linear sparsity using alternating directions. In *Advances in Neural Information Processing Systems*, pages 3401–3409, 2014.

[134] Mostafa Rahmani and George Atia. Coherence pursuit: Fast, simple, and robust principal component analysis. *arXiv preprint arXiv:1609.04789*, 2016.

[135] 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*, 32(10):1832–1845, 2010.

[136] Richard Serfozo. *Basics of applied stochastic processes*. Springer Science & Business Media, 2009.

[137] Jie Shen, Ping Li, and Huan Xu. Online low-rank subspace clustering by basis dictionary pursuit. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 622–631, 2016.

[138] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[139] M. Soltanolkotabi and E. J. Candès. A geometric analysis of subspace clustering with outliers. *Annals of Statistics*, 40(4):2195–2238, 2012.

[140] M. Soltanolkotabi, E. Elhamifar, and E. J. Candès. Robust subspace clustering. *Annals of Statistics*, 42(2):669–699, 2014.

[141] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, (0):–, 2012.

[142] T. Tao. *Topics in Random Matrix Theory*. Graduate studies in mathematics. American Mathematical Society, 2012.

[143] Henk C Tijms. *A first course in stochastic models*. John Wiley and Sons, 2003.

[144] Tomasz Tkocz. An upper bound for spherical caps. *The American Mathematical Monthly*, 119(7):606–607, 2012.

[145] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography. *International Journal of Computer Vision*, 9(2):137–154, 1992.

[146] L. Fejes Toth. On covering a spherical surface with equal spherical caps (in hungarian). *Matematikai Fiz. Lapok*, (50):40–46, 1943.

[147] Panagiotis A. Traganitis and Georgios B. Giannakis. Sketched subspace clustering. *IEEE Transactions on Signal Processing*, 2017.

[148] J. Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE Transactions on Information Theory*, 50(10):2231–2242, Oct. 2004.

[149] J. A. Tropp. Algorithms for simultaneous sparse approximation. part ii: Convex relaxation. *Signal Processing, Special Issue on Sparse approximations in signal and image processing*, 86:589–602, 2006.

[150] Joel A Tropp, Anna C Gilbert, and Martin J Strauss. Algorithms for simultaneous sparse approximation. part i: Greedy pursuit. *Signal Processing*, 86(3):572–588, 2006.

[151] M. C. Tsakiris and R. Vidal. Filtrated algebraic subspace clustering. *SIAM Journal on Imaging Sciences*, 10(1):372–415, 2017.

[152] M. C. Tsakiris and R. Vidal. Dual principal component pursuit. *Journal of Machine Learning Research*, 18(19):1–50, 2018.

[153] M. C. Tsakiris and R. Vidal. Theoretical analysis of sparse subspace clustering with missing entries. *arXiv:1801.00393*, 2018.

[154] M.C. Tsakiris and R. Vidal. Dual principal component pursuit. In *ICCV Workshop on Robust Subspace Learning and Computer Vision*, pages 10–18, 2015.

[155] M.C. Tsakiris and R. Vidal. Filtrated spectral algebraic subspace clustering. In *ICCV Workshop on Robust Subspace Learning and Computer Vision*, pages 28–36, 2015.

[156] Michael Tschannen and Helmut Bolcskei. Noisy subspace clustering via matching pursuits. *arXiv:1612.03450*, 2016.

[157] P. Tseng. Nearest q-flat to m points. *Journal of Optimization Theory and Applications*, 105(1):249–252, 2000.

[158] R. van Handel. *Probability in High Dimension*. ORF 570 Lecture Notes. Princeton University, 2014.

[159] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org/`, 2008.

[160] Roman Vershynin. Lectures in geometric functional analysis. 2009.

[161] R. Vidal. Subspace clustering. *IEEE Signal Processing Magazine*, 28(3):52–68, March 2011.

[162] R. Vidal and P. Favaro. Low rank subspace clustering (LRSC). *Pattern Recognition Letters*, 43:47–61, 2014.

[163] R. Vidal, Y. Ma, and S. Sastry. Generalized Principal Component Analysis (GPCA). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(12):1–15, 2005.

[164] R. Vidal, Y. Ma, and S. Sastry. *Generalized Principal Component Analysis*. Springer Verlag, 2016.

[165] R. Vidal, R. Tron, and R. Hartley. Multiframe motion segmentation with missing data using PowerFactorization, and GPCA. *International Journal of Computer Vision*, 79(1):85–105, 2008.

[166] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[167] Hongxing Wang, Yoshinobu Kawahara, Chaoqun Weng, and Junsong Yuan. Representative selection with structured sparsity. *Pattern Recognition*, 63:268–278, 2017.

[168] Yin Wang, Caglayan Dicle, Mario Sznaier, and Octavia Camps. Self scaled regularized robust regression. In *Proceedings of the IEEE Con-*

ference on Computer Vision and Pattern Recognition, pages 3261–3269, 2015.

[169] Yining Wang, Yu-Xiang Wang, and Aarti Singh. Graph connectivity in noisy sparse subspace clustering. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pages 538–546, 2016.

[170] Yu-Xiang Wang and Huan Xu. Noisy sparse subspace clustering. In *International Conference on Machine Learning*, pages 89–97, 2013.

[171] Yu-Xiang Wang and Huan Xu. Noisy sparse subspace clustering. *Journal of Machine Learning Research*, 17(12):1–41, 2016.

[172] Yu-Xiang Wang, Huan Xu, and Chenlei Leng. Provable subspace clustering: When LRR meets SSC. In *Neural Information Processing Systems*, 2013.

[173] Zhaowen Wang, Jianchao Yang, Nasser Nasrabadi, and Thomas Huang. A max-margin perspective on sparse representation-based classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1217–1224, 2013.

[174] Jinming Wen, Zhengchun Zhou, Jian Wang, Xiaohu Tang, and Qun Mo. A

sharp condition for exact support recovery with orthogonal matching pursuit. *IEEE Transactions on Signal Processing*, 65(6):1370–1382, 2017.

[175] David P Williamson and David B Shmoys. *The design of approximation algorithms*. Cambridge university press, 2011.

[176] J. Wright, A. Ganesh, S. Rao, Y. Peng, and Y. Ma. Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. In *NIPS*, 2009.

[177] 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*, 31(2):210–227, Feb. 2009.

[178] John Wright and Yi Ma. Dense error correction via ℓ^1-minimization. *IEEE Transactions on Information Theory*, 56(7):3540–3560, 2010.

[179] John Wright, Allen Y. Yang, Arvind Ganesh, S. Shankar Sastry, and Yi Ma. Robust face recognition via sparse representation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(2):210–227, 2009.

[180] S. J. Wright, R. D. Nowak, and M. A. T. Figueiredo. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57:2479–2493, 2009.

[181] Shijie Xiao, Wen Li, Dong Xu, and Dacheng Tao. Falrr: A fast low rank

representation solver. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4612–4620, 2015.

[182] Bo Xin, Yizhou Wang, Wen Gao, and David Wipf. Building invariances into sparse subspace clustering. *IEEE Transactions on Signal Processing*, 66(2):449–462, 2018.

[183] Huan Xu, Constantine Caramanis, and Sujay Sanghavi. Robust pca via outlier pursuit. In *Advances in Neural Information Processing Systems*, pages 2496–2504, 2010.

[184] J. Yan and M. Pollefeys. A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate. In *European Conference on Computer Vision*, pages 94–106, 2006.

[185] A. Y. Yang, S. Rao, and Y. Ma. Robust statistical estimation and segmentation of multiple subspaces. In *Workshop on 25 years of RANSAC*, 2006.

[186] Allen Y Yang, Zihan Zhou, Arvind Ganesh Balasubramanian, S Shankar Sastry, and Yi Ma. Fast ℓ_1-minimization algorithms for robust face recognition. *IEEE Transactions on Image Processing*, 22(8):3234–3246, 2013.

[187] Jian Yang, Lei Zhang, Yong Xu, and Jing-Yu Yang. Beyond sparsity:

The role of l_1-optimizer in pattern classification. *Pattern Recognition*, 45(3):1104–1118, 2012.

[188] Yingzhen Yang, Jiashi Feng, Nebojsa Jojic, Jianchao Yang, and Thomas S Huang. ℓ_0-sparse subspace clustering. In *European Conference on Computer Vision*, pages 731–747, 2016.

[189] C. You, C. Donnat, D. Robinson, and R. Vidal. A divide-and-conquer framework for large-scale subspace clustering. In *Asilomar Conference on Signals, Systems and Computers*, 2016.

[190] C. You, C. Li, D. Robinson, and R. Vidal. A scalable exemplar-based subspace clustering algorithm for class-imbalanced data. In *European Conference on Computer Vision*, 2018.

[191] C. You, C.-G. Li, D. Robinson, and R. Vidal. Oracle based active set algorithm for scalable elastic net subspace clustering. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3928–3937, 2016.

[192] C. You, D. Robinson, and R. Vidal.

[193] C. You, D. Robinson, and R. Vidal. Scalable sparse subspace clustering by orthogonal matching pursuit. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3918–3927, 2016.

[194] C. You, D. Robinson, and R. Vidal. Provable self-representation based

outlier detection in a union of subspaces. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4323–4332, 2017.

[195] C. You and R. Vidal. Geometric conditions for subspace-sparse recovery. In *International Conference on Machine learning*, pages 1585–1593, 2015.

[196] Lei Zhang, Meng Yang, and Xiangchu Feng. Sparse representation or collaborative representation: Which helps face recognition? In *IEEE International Conference on Computer Vision*, pages 471–478, 2011.

[197] Lei-Hong Zhang, Wei Hong Yang, and Li-Zhi Liao. On an efficient implementation of the face algorithm for linear programming. *Journal of Computational Mathematics*, 31:335–354, 2013.

[198] T. Zhang, A. Szlam, and G. Lerman. Median k-flats for hybrid linear modeling with many outliers. In *Workshop on Subspace Methods*, pages 234–241, 2009.

[199] T. Zhang, A. Szlam, Y. Wang, and G. Lerman. Hybrid linear modeling via local best-fit flats. *International Journal of Computer Vision*, 100(3):217–240, 2012.

[200] Teng Zhang and Gilad Lerman. A novel m-estimator for robust pca. *The Journal of Machine Learning Research*, 15(1):749–808, 2014.

[201] Yinqiang Zheng, Shigeki Sugimoto, and Masatoshi Okutomi. Deterministically maximizing feasible subsystem for robust model fitting with unit norm constraint. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1825–1832. IEEE, 2011.

[202] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, 67:301–320, 2005.

Vita

Chong You was born in Beijing, China in 1987. He received his undergraduate degree in Electrical Engineering and a double degree in Applied Mathematics from Peking University in 2009. He received his M.S. degree in Electrical Engineering from Peking University in 2012. His research interest lies in the intersection of machine learning, numerical optimization, signal processing and computer vision for modern data analysis. His doctoral research focuses on developing provable algorithms to uncover low-dimensional structures in high-dimensional datasets.