RICE UNIVERSITY

New Theory and Methods for Signals in Unions of Subspaces

by

Eva L. Dyer

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE

Doctor of Philosophy

APPROVED, THESIS COMMITTEE:

Dr. Richard G. Baraniuk, Chair
Victor E. Cameron Professor of Electrical and Computer Engineering

Dr. Genevera I. Allen
Assistant Professor, Statistics Department

Dr. Farinaz Koushanfar
Associate Professor, Department of Electrical and Computer Engineering

Dr. Ashutosh Sabharwal
Professor, Department of Electrical and Computer Engineering

HOUSTON, TEXAS
SEPTEMBER 2014
I am incredibly grateful that I have had the opportunity to work with a team of brilliant and amazing people over the course of my graduate studies. First and foremost, I would like to acknowledge my advisor Prof. Richard G. Baraniuk, who has provided me with a constant and seemingly relentless stream of new and exciting ideas and problems to work on. Rich has always encouraged me to explore new and crazy ideas and most importantly, think big. Over the course of my graduate studies, he gave me tons of opportunities to learn and grow, both intellectually and as a person.

I am also indebted to Prof. Don H. Johnson whom played a major role in my early years as a graduate student and provided a sounding board for my crazy ideas on neural coding and the inner workings of 'da brain'. Along the way, I also was fortunate to have the chance to work with Prof. Farinaz Koushanfar, who was not only an excellent person to collaborate with, but also a mentor, confidant, and great friend. I would also like to acknowledge Prof. Jacob Robinson for all his help and introducing me to some of the emerging challenges at the interface of big data and neuroscience. I’m excited to continue to work in this field in the future and happy to have had a chance to meet you. I would also like to acknowledge all of my committee members, Farinaz Koushanfar, Genevera Allen, and Ashutosh Sabharwal. All of the members of my committee provided insightful and useful comments on the thesis that
have helped to improve the thesis immensely.

The DSP group has been a vibrant and exciting team to work with. I am so happy that I had the chance to brainstorm and work with so many of the grad students and postdocs in the group. Thank you Jason, Mark, Marco, Chinmay, Drew, Mr. Lan, Arian, Raj, Heather, Momo, Matthew, and Ankit for making my life at Rice, brainstorming sessions, and group meetings fun and enriching. Minh, you were a joy to work with and mentor and I know that you will do amazing in your future. A special thank you goes to Aswin, who played a major role in the early development of the ideas in this thesis. Thank you to Tom and Christoph for making my later years at Rice exciting, both inside and outside of the office. Christoph, you are an incredible mentor and have been instrumental in helping me to develop as a researcher; you made work not feel like work and paper writing fun. Thanks to you, I have learned so much and I am forever grateful. I would also like to thank Farinaz’s students, Mehrdad and Azalia, for being great collaborators and friends.

My experience in Houston wouldn’t have been nearly as fun and rewarding without a number of great friends that I have met along the way. Thank you Lindsey, Jed, Cynthia, Azalia, Manjari, Sam, Rose, David, Gareth, and Shaun for being good friends, giving great career and life advice, and being awesome people. Jed I had so much fun writing my thesis with you, I will forever remember our special months together as thesis buddies!

Last but not least, I would like to thank my family for being supportive of all of my endeavors and being my biggest fans.

When asked as a kid who my hero was, I would always say Albert
Einstein. I’d like to end my acknowledgements with a quote from Einstein that always resonated with me. Throughout my Ph.D., I always reminded myself of the message behind this quote and the importance of imagination and creativity.

“Imagination is more important than knowledge. For knowledge is limited to all we now know and understand, while imagination embraces the entire world, and all there ever will be to know and understand.”

– Albert Einstein
This thesis is dedicated to my mom. I couldn’t have done any of this without your support and love.
Acknowledgements

Abstract

1 Introduction

1.1 Low-dimensional Signal Models

1.2 Unions of Subspaces

1.3 Main Contributions

1.3.1 Part 1: Self-Expressive Decompositions

1.3.2 Part 2: Methods for Subspace Clustering

1.3.3 Part 3: Structure-Aware Distributed Computing

1.4 Motivating Applications

1.5 Organization

2 Low-dimensional Signal Models

2.1 Preliminaries

2.1.1 Vector Spaces

2.1.2 Singular Value Decomposition
2.1.3 Pseudoinverse and Projectors .................................. 18
2.1.4 Notation ............................................................... 19
2.2 Signal Models ............................................................ 19
  2.2.1 Linear Subspace Models .......................................... 20
  2.2.2 Unions of Subspaces .............................................. 21
    A Generative Model for Unions of Subspaces ................. 22
2.3 Metrics for Characterizing Geometric Structure in Data .......... 23
  2.3.1 Coherence and Cumulative Coherence ....................... 23
  2.3.2 Principal Angles ............................................... 24
  2.3.3 Covering Radius ............................................... 26
  2.3.4 Matrix Spark .................................................. 27

3 Sparse Signal Models and Recovery Methods .......................... 30
  3.1 Signal Representation .............................................. 30
  3.2 Redundant Expansions .............................................. 33
    3.2.1 Method of Frames .......................................... 33
  3.3 Sparse Approximation ............................................. 34
    3.3.1 Sparse Signal Models ....................................... 34
    3.3.2 Sparse Approximation Problems ............................ 37
  3.4 Convex Methods for Sparse Recovery .............................. 38
    3.4.1 Sparse Recovery Using the $\ell_1$-norm ................... 38
    3.4.2 Projected Gradient Descent Methods ....................... 39
    Gradient Descent ............................................ 39
    Projected Gradient Descent .................................. 40
    Iterative Soft Thresholding Algorithm ....................... 42
  3.5 Greedy Methods for Sparse Recovery .............................. 43
    OMP Algorithm ............................................... 44
Complexity of OMP ................................................. 45
Batch OMP Algorithm ............................................. 46

3.6 Exact Signal Recovery Conditions .......................... 47
  3.6.1 Sub-dictionaries ............................................ 47
  3.6.2 Exact Recovery Conditions ............................... 47
  3.6.3 Coherence-Based Recovery Conditions .................. 48

4 Matrix Factorization ............................................ 50
  4.1 Matrix Factorization .......................................... 50
    4.1.1 Principal Components Analysis .......................... 50
    4.1.2 Sparse Principal Components Analysis ................ 51
    4.1.3 Dictionary Learning ...................................... 53
  4.2 Column Selection for Matrix Factorization ............... 55
    4.2.1 Column Subset Selection ................................ 55
    4.2.2 Nystrom Method ........................................ 56
    4.2.3 Adaptive Sampling Methods ............................. 56

5 Self-Expressive Sparse Recovery .............................. 58
  5.1 Self-Expressive Sparse Recovery ............................ 58
  5.2 Sparse Self-expressive Decomposition (SEED) ............. 60
    5.2.1 Motivation ............................................... 60
    5.2.2 SEED Method ............................................. 61
    Step 1: Column Subset Selection with oASIS ................ 61
    Step 2: Greedy Sparse Recovery with OMP .................. 62
    5.2.3 Derivation of oASIS ..................................... 62
    Accelerated Sampling ........................................ 64
    5.2.4 Variant of Alg. 2 ........................................ 65
5.3 Outlier Detection with SEED
5.3.1 Motivation
5.3.2 Sparsity-driven Outlier Detection
5.4 Numerical Experiments
5.4.1 Evaluation Setup
5.4.2 Datasets
5.4.3 Generative Model for Synthetic Data
  Constructing Sub-dictionaries
  Coefficient Synthesis
5.4.4 Results
  Results for YF Dataset
  Results for LF Datasets
  Results for HS Datasets
  Results for MS Dataset
  Results for US Dataset

6 Self-Expressive Decompositions for Subspace Clustering
6.1 Subspace Clustering
6.2 Sparse Subspace Clustering
6.3 Greedy Feature Selection for Subspace Clustering
  6.3.1 Greedy Spectral Clustering Approach (SSC-OMP)
  6.3.2 Greedy Consensus Approach
6.4 SEED-Clust Method
  6.4.1 Bipartite Graph Partitioning
  6.4.2 Comparison of SEED and SSC-OMP
6.5 Numerical Experiments
  6.5.1 Clustering Performance of SSC-OMP
Phase Transitions for SSC-OMP ........................................ 96
Comparison of OMP and NN ........................................ 98
Results for YF Dataset ........................................... 101
Results for MS Dataset ........................................... 104
Comparison of Consensus and Spectral Clustering .............. 105

6.5.2 Clustering Performance for SEED ............................ 107
Evaluation Setup .................................................. 108
Results for YF Datasets ........................................... 109
Results for HS Datasets ........................................... 109
Results for MS Dataset ........................................... 113
Results for US Data ................................................ 114

7 Geometric Analysis of Self-Expressive Sparse Recovery 117

7.1 Analysis of Exact Feature Selection ............................ 117
7.1.1 Exact Feature Selection ..................................... 117
Implications of EFS for Subspace Clustering ....................... 118
7.1.2 Complete Reference Set .................................... 119
7.1.3 Sufficient Conditions for EFS ............................... 121
EFS for Disjoint Subspaces ....................................... 123
7.1.4 Geometry Underlying EFS .................................. 124
7.1.5 Connections to Previous Work .............................. 125
Subspace Clustering with BP ...................................... 126
Exact Recovery Conditions for Sparse Recovery ................. 128
7.1.6 EFS for Bounded Unions of Subspaces .................... 128
7.1.7 Proofs ....................................................... 131
Proof of Thm. 3 .................................................. 131
Proof of Thm. 4 .................................................. 133
Proof of Thm. 5 ........................................... 135

7.2 Rank Revealing Property of SEED .......................... 137
  7.2.1 Lower Bound on Sparsity Level ..................... 138
  7.2.2 Upper Bound on Sparsity Level ...................... 139
  7.2.3 Results for Outlier Detection ...................... 139
  7.2.4 Proofs ............................................ 141
    Proof of Thm. 6 ................................... 141
    Proof of Prop. 2 .................................... 142

7.3 Exact Matrix Recovery .................................... 144
  7.3.1 Condition for Exact Matrix Recovery ............. 144
  7.3.2 Proof of Thm. 8 .................................. 146
  7.3.3 Proof of Thm. 9 .................................. 148

7.4 Complexity Analysis of SEED ............................ 149
  7.4.1 Complexity of SEED .................................. 149
  7.4.2 Comparison with Other Methods .................... 150

8 Structure-Aware Distributed Computing ........................ 152
  8.1 Motivation ........................................... 153
    8.1.1 Our Approach ................................... 154
  8.2 RankMap Framework ................................... 157
    8.2.1 Target Applications .............................. 157
    8.2.2 Overview of RankMap ............................ 159
  8.3 Sparse Matrix Decomposition Approach ................. 161
    8.3.1 Sparse Matrix Decomposition via SEED ............. 161
    8.3.2 Impact of Approximation Error on Iterative Updates ... 163
      Bound on Solution Error .............................. 164
      Proof of Thm. 10 .................................. 165
8.3.3 Sparsity Guarantees for Unions of Subspaces . . . . . . . . . . 168
8.3.4 Complexity Analysis . . . . . . . . . . . . . . . . . . . . . . . 169
8.4 Execution Flow of Iterative Updates . . . . . . . . . . . . . . . . 170
  8.4.1 Graph-based Model . . . . . . . . . . . . . . . . . . . . . . . 170
  8.4.2 Matrix-based Model . . . . . . . . . . . . . . . . . . . . . . . 173
8.5 Data Partitioning and Distributed Computing . . . . . . . . . . . . 173
  8.5.1 Distributed Graph Allocation . . . . . . . . . . . . . . . . . . 173
  8.5.2 Distributed Matrix Allocation . . . . . . . . . . . . . . . . . . 175
  8.5.3 Performance Bounds . . . . . . . . . . . . . . . . . . . . . . . 176
8.6 Evaluations . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 178
  8.6.1 Evaluation Setup . . . . . . . . . . . . . . . . . . . . . . . . . 178
    Dataset and Application . . . . . . . . . . . . . . . . . . . . . . 178
    Computing Platform . . . . . . . . . . . . . . . . . . . . . . . . . 179
    Distributed Tools . . . . . . . . . . . . . . . . . . . . . . . . . . 179
  8.6.2 Decomposition Results . . . . . . . . . . . . . . . . . . . . . . 180
    Comparison of Decomposition Methods . . . . . . . . . . . . . . . 180
    Decomposition Error . . . . . . . . . . . . . . . . . . . . . . . . . 181
    Sparsity of Decomposition . . . . . . . . . . . . . . . . . . . . . . 181
  8.6.3 Reconstruction Results . . . . . . . . . . . . . . . . . . . . . . 181
    Reconstruction Method . . . . . . . . . . . . . . . . . . . . . . . . 182
    Reconstruction Performance . . . . . . . . . . . . . . . . . . . . . 182
  8.6.4 Graph vs. Matrix-based Models . . . . . . . . . . . . . . . . . 183
    Runtime Efficiency . . . . . . . . . . . . . . . . . . . . . . . . . . 183
    Memory Efficiency . . . . . . . . . . . . . . . . . . . . . . . . . . 185

9 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 189
  9.1 The Power of Self-Expression . . . . . . . . . . . . . . . . . . . . . 191
9.2 “Data Driven” Sparse Approximation ........................ 191
9.3 Learning Block-Sparse Signal Models .......................... 192
9.4 Beyond Coherence ............................................. 193
9.5 Discriminative Dictionary Learning ............................ 193
9.6 Greedy Feature Selection ....................................... 194

References .................................................................. 196
List of Figures

1.1 Visualizing low-dimensional structure in data. ............... 4
2.1 Examples of low-dimensional signal models for collections of data. ... 20
2.2 Covering radius and deep holes. .................................. 27
2.3 Covering radius in the ambient space. .............................. 28
3.1 Example of a sparse time-frequency representation. ............ 31
3.2 Sparse representation of images with wavelets. .................. 34
3.3 Limitations of orthogonal wavelet decompositions in representing edge content in images. ............................... 35
3.4 Example of a redundant dictionary for images. ................... 35
5.1 Demonstration of rank revealing property of SEED for outlier detection. 68
5.2 Example of hyperspectral signals arising in remote sensing data. ... 72
5.3 Generating unions of subspaces from shift-invariant dictionaries. ... 75
5.4 Components selected from collections of face images. ............. 80
5.5 Factorization error for face image database. ....................... 81
5.6 Factorization error for light field data. ............................ 82
5.7  Factorization error for hyperspectral image data. .......................... 82
5.8  Factorization error for motion segmentation data. ......................... 83
5.9  Factorization error for synthetic union of subspace data corrupted by
     outliers. ................................................................................. 84

6.1  Comparison of the nearest neighbor graph and $\ell_0$-graph for unions of
     illumination subspaces ......................................................... 89
6.2  Visualization of spectral co-clustering matrix for unions of subspaces.  94
6.3  Probability of EFS for different sampling conditions ....................... 97
6.4  Probability of EFS for unions with structured cross-spectra ............. 99
6.5  Phase transitions for OMP and nearest neighbor graphs. ............... 101
6.6  Comparison of subspace affinities for face image database. ............ 102
6.7  Cross-spectra for illumination subspaces .................................. 104
6.8  Classification performance for segmenting two motions. ............... 105
6.9  Classification performance for segmenting three motions. ............. 105
6.10 Probability of subspace recovery ........................................ 106
6.11 Performance of subspace consensus. ...................................... 107
6.12 Normalized cut ratios for face image database. ............................ 110
6.13 Sparse coefficient matrices obtained for hyperspectral image data. . 112
6.14 Affinity matrices for hyperspectral image data. .......................... 112
6.15 Visualization of co-clustering matrix for unions of subspaces with outliers.115

7.1  Geometry underlying EFS from disjoint subspaces. .......................... 123

8.1  Graph decomposition approach underlying RankMap. .................... 157
8.2  Overview of RankMap framework. ......................................... 159
8.3  Distributed graph decomposition architecture. .......................... 171
8.4  Distributed design of graph-based model. ................................. 174
8.5 Distributed design of matrix-based model. . . . . . . . . . . . . . . . 174

8.6 Comparison of error and sparsity for adaptive and random sampling
methods. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 186

8.7 Runtime comparisons for iterative updates for FISTA. . . . . . . . . 187

8.8 Comparison of matrix-based and graph-based architectures. . . . . . 188
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>Classification error and EFS rates for illumination subspaces.</td>
<td>103</td>
</tr>
<tr>
<td>6.2</td>
<td>Comparison of normalized graph cut metrics for hyperspectral image data.</td>
<td>113</td>
</tr>
<tr>
<td>6.3</td>
<td>Comparison of normalized graph cut metrics for motion segmentation data.</td>
<td>113</td>
</tr>
<tr>
<td>6.4</td>
<td>Normalized cut metrics for synthetic data living on a union of five overlapping subspaces.</td>
<td>114</td>
</tr>
<tr>
<td>8.1</td>
<td>Complexity of different decomposition methods.</td>
<td>180</td>
</tr>
<tr>
<td>8.2</td>
<td>Comparison of memory usage (MB).</td>
<td>185</td>
</tr>
</tbody>
</table>
Abstract

New Theory and Methods for Signals in Unions of Subspaces

by

Eva L. Dyer

The rapid development and availability of cheap storage and sensing devices has quickly produced a deluge of high-dimensional data. While the dimensionality of modern datasets continues to grow, our saving grace is that these data often exhibit low-dimensional structure that can be exploited to compress, organize, and cluster massive collections of data.

Signal models such as linear subspace models, remain one of the most widely used models for high-dimensional data; however, in many settings of interest, finding a global model that can capture all the relevant structure in the data is not possible. Thus, an alternative to learning a global model is to instead learn a hybrid model or a union of low-dimensional subspaces that model different subsets of signals in the dataset as living on distinct subspaces.

This thesis develops new methods and theory for learning union of subspace models as well as exploiting multi-subspace structure in a wide range of signal processing and data analysis tasks. The main contributions of this thesis include new methods and theory for: (i) decomposing and subsampling datasets consisting of signals on unions of subspaces, (ii) subspace clustering for learning union of subspace models, and (iii) exploiting multi-subspace structure in order accelerate distributed computing and signal processing on massive collections of data. I demonstrate the utility of the proposed methods in a number of important imaging and computer vision applications including: illumination-invariant face recognition, segmentation of hyperspectral remote sensing data, and compression of video and light field data arising in 3D scene modeling and analysis.
Chapter 1

Introduction

1.1 Low-dimensional Signal Models

Cameras are getting smaller and cheaper, the amount of data we can store on a hard drives is increasing, and computing devices are getting more portable and commonplace. With the rapid development and availability of storage and sensing devices and the widespread use of the internet, it has become increasingly easy to snap a picture or video and upload it to the web. As a result, we are in the midst of an explosion of digital content that has become extremely difficult to compress, organize, and extract information from. To put this in perspective, it is estimated that each minute 48 hours of video is uploaded to YouTube, 3,000 images are added to FlickR, and 684,000 pieces of content are uploaded to Facebook.

While an extremely potent example of “big data”, web-based content is just one example of how our digital universe is rapidly expanding. New data collection methods in neuroscience that aim to map living neural circuits, remote sensing data from satellites and unmanned aerial vehicles, and novel assays of genetic and cellular material are a few other instances of how technology has contributed to data getting bigger across many different domains.
Although the size of modern datasets is rapidly expanding, the intrinsic dimensionality of many of these datasets is typically much lower due to similarities and shared features between data entries. *Dimensionality reduction* methods exploit any low-dimensional structure present in a collection of data to compress, organize, and partition data according to its underlying structure. One approach used to reduce the dimensionality of collections of data is to learn a low-dimensional model that captures structure present across the collection rather than forming independent representations for each signal in the collection. Once a low-dimensional model is learned from the data, this model can be used to more efficiently organize, partition, and reduce the dimensionality of the collection.

To make this idea precise, consider a collection of $N$ signals each of $M$ dimensions (a collection of $N$ images with $M$ pixels for instance). Each signal (image) can be thought of as a single “point” in the ambient space, $\mathbb{R}^M$. Thus, if we consider a collection of $N$ such signals, which we stack into a matrix $X \in \mathbb{R}^{M \times N}$, then this data matrix generates a collection of $N$ points in $\mathbb{R}^M$. The key insight is that when the signals are correlated or exhibit redundancies, the points generated by the collection will not be spread across all of $\mathbb{R}^M$ but instead lie in a small subset of the space. See Fig. 1.1 for a demonstration of this concept, where in this example, each point corresponds to an image of a subject’s face under a different illumination condition which collectively occupy a linear subspace of $\mathbb{R}^M$.

When a collection of data exhibits low-dimensional structure, then a low-dimensional model that is well matched to this structure can be used to exploit this fact in subsequent processing. Linear subspace models are one of the most widely used signal models for collections of high-dimensional data, with applications throughout signal processing, machine learning, and the computational sciences. The widespread use of subspace models is due in part to their simplicity but also due to the fact that
Figure 1.1: Visualizing low-dimensional structure in data. Each point along the plane corresponds to an image of a subject’s face under a distinct illumination condition.

Principal components analysis (PCA) [1] provides a closed-form and computationally efficient solution to the problem of finding an optimal low-rank approximation to a collection of data.

1.2 Unions of Subspaces

While linear subspace models are powerful models for high-dimensional data, in many settings of interest, finding a global model that provides a compact representation of the geometric structures present across an entire dataset is often not possible. Thus, an alternative to learning a global model is to instead learn a hybrid model or a union of low-dimensional subspaces that encapsulate the structure present in the dataset. For instance, if we revisit our example in Fig. 1.1 and consider a collection of images of multiple subject’s faces under different illumination conditions, the structure of the
resulting collection can be more compactly modeled by a union of low-dimensional subspaces, where images of each subject’s face lie on a different subspace.

Unions of subspaces provide a compact geometric model for a wide range of datasets including: collections of images of objects under different illumination conditions [2], motion trajectories of point-correspondences arising from different objects [3], and a wide range of structured sparse and block-sparse signals [4, 5, 6, 7]. Unions of subspace models have also been used in the decoding of neural data collected from the motor cortex at different points in time [8]. After learning a hybrid model, then this model can be employed for denoising [9, 10], inpainting [9], and to form a more compact (sparse) representation of image data [11].

Formally, we say that a signal \( \mathbf{x} \in \mathbb{R}^M \) lies on a union of \( p \) subspaces of \( \mathbb{R}^M \), when \( \mathbf{x} \in \mathcal{U} = \bigcup_{i=1}^{p} S_i \) and \( S_i \) is a linear subspace of \( \mathbb{R}^M \). To learn \( \mathcal{U} \) from a collection of \( N \) signals in \( \mathbb{R}^M \), denoted by \( \mathbf{X} \in \mathbb{R}^{M \times N} \), we assume that the columns of \( \mathbf{X} \) are “spread” along each of the \( p \) subspaces in the union, i.e., there exists at least \( k \) linearly independent vectors that span each \( k \)-dimensional subspace in \( \mathcal{U} \). To demonstrate the geometry of hybrid models and unions of subspaces, we point the reader to Fig. 2.1, where we provide a pictorial example of different linear and nonlinear models for collections of high-dimensional data.

Unions of subspaces provide a natural extension to single subspace models; however, the application of unions of subspaces to data analysis and signal processing has been limited thus far. One of the main reasons why hybrid models haven’t yet gained traction is due to the fact that learning hybrid models is challenging. This is due to the fact that in order to learn a hybrid model for a collection of data, both \textit{subspace clustering}—or clustering points in accordance with their subspace membership—and \textit{subspace estimation} must be performed jointly. Nevertheless, if we can sift through the points in the dataset and identify subsets of points that lie along or near the same
subspace, then a local subspace estimate\(^1\) formed from any such set is guaranteed to coincide with one of the true subspaces present in the dataset [12, 13].

Another major reason why hybrid models have yet to gain traction is due the fact that existing methods for modeling and processing data that admit multi-subspace structure must be stitched together one piece at a time. First, subspace clustering is performed, a hybrid model is learned after clustering data into different subspaces, and finally different dimensionality reduction methods can be applied to data based upon this learned model. In contrast, PCA provides a natural way in which to model and process low rank data that can be well-approximated by a linear subspace as it provides a means by which the data can be compressed, clustered, and modeled via a linear subspace. To date, no unified approaches have been proposed for both dimensionality reduction and clustering of datasets that exhibit multi-subspace structure.

In order to fully exploit this unique signal structure in large collections of data, new methods and theory are required to provide a similar holistic approach for processing multi-subspace data.

### 1.3 Main Contributions

The aim of this thesis is to take a step towards providing a unified approach for signal processing and data analysis tasks for signals that admit multi-subspace structure. The main idea underlying the methods and theory developed over the course of the thesis is to decompose or factorize the data in such a way that the resulting decomposition can be used for dimensionality reduction, subspace clustering, and to learn a hybrid model. To achieve this goal, we develop methods for forming sparse factorizations of data that rely on “self expression” or sparsely representing signals

\(^1\)A local subspace estimate is a low-rank approximation formed from a subset of points in the ensemble, rather than from the entire collection of data.
in a collection of data with respect to a small subset of the dataset. We demonstrate that in theory and in practice, the proposed methods provide a provable strategy for dimensionality reduction, clustering, and learning a union of subspace model from datasets that admit multi-subspace structure. An important aspect of our approach is that it is efficient and scalable and thus provides an alternative to other sparse matrix factorization methods that are prohibitive for extremely large collections of data. Moreover, the proposed methods for self-expressive decompositions also admit a rich geometric analysis and guarantees regarding the performance of these methods. In addition to providing new methods and theory for self-expressive decompositions (SEED) of big datasets, we show how this same factorization approach can be used to exploit multi-subspace structure to accelerate large-scale distributed processing of massive datasets in a number of important machine learning applications.

We now provide a brief summary of the main contributions of this thesis which we organize into three main parts: (i) dimensionality reduction, (ii) subspace clustering, and (iii) distributed computing with SEED.

1.3.1 Part 1: Self-Expressive Decompositions

In the first part of the thesis, we introduce new algorithms and theory for sparse self-expressive decompositions and demonstrate the utility of SEED for dimensionality reduction. We introduce a novel approach for sparse matrix decomposition that forms a sparse representation of the data in terms of a small but intelligently selected subset of the data. To form an efficient decomposition of a large collection of data, we develop a novel sampling algorithm, which we dub oASIS (Accelerated Sequential Incoherence Sampling) which is not only computationally efficient but also leads to guarantees regarding the linear independence of the samples/columns selected via this method. This property of oASIS enables us to develop guarantees regarding our
ability to exactly recover a low rank matrix of rank $r$ with only $r$ samples from the dataset. Thus, this approach achieves the same rate of dimensionality reduction as PCA for low rank matrices. By coupling oASIS with greedy sparse recovery methods, we obtain an efficient decomposition of data that lie on union of subspaces.

In addition to demonstrating that we obtain an exact recovery of the underlying data, we also develop guarantees that the resulting decomposition will be sparse. We provide theory that when the data lie on unions of separated subspaces, this can be sufficient to guarantee that the resulting decomposition is rank revealing. This means that the decomposition of multi-subspace data that lie on at most $k$-dimensional subspaces can be guaranteed to be $k$-sparse, i.e., contain at most $k$ non-zeros. Thus, the factorization provided by our methods only require storage (number of non-zeros) proportional to the subspace dimension, rather than the rank of the data (which can be much higher for unions of subspaces). This rank revealing property of sparse self-expression also leads to a provable strategy for outlier segmentation that ensures that our methods are robust to outliers and thus, outliers do not amplify the size/storage of the decomposition.

The specific contributions of this part of the thesis are provided below.

- In Sec. 5.1, we introduce a greedy sparse recovery strategy for self-expression that decomposes a collection of data in terms of other signals in the same collection.

- In Sec. 5.2, we introduce a novel method for sparse matrix factorization that we dub sparse SELF-Expressive Decomposition or SEED. This method consists of two steps: (i) adaptively sampling columns from a matrix and (ii) computing sparse representations of the remaining data in terms of the columns selected in the first step.

- In Sec. 5.2.2, we introduce a novel approach for column subset selection that is
based upon sequentially selecting columns that are incoherent from previously sampled columns, called oASIS. We analyze the performance of oASIS in Sec. 7.3.2 and provide guarantees that this method will select a set of \( r \) linearly independent columns from a rank \( r \) matrix in exactly \( r \) iterations. Using this property of oASIS, we prove that this sampling method provides an efficient and provable strategy for exact matrix recovery of low rank matrices.

- In Sec. 7.2, we develop sufficient conditions that are required to produce efficient sparse decompositions where the sparsity of each signal is bounded by the subspace dimension of the underlying subspace structures. This guarantees that the storage and compression of the decompositions is bounded by the subspace dimension which can provide sufficiently more compression and reduction for multi-subspace data than linear subspace methods such as PCA.

- In Sec. 5.3, we develop a method for outlier detection that is based upon the sparsity level of each signal computed via SEED. We show that when the dataset consists of a low rank subspace and incoherent outlier points, this method can be used for provable outlier detection. In Sec. 7.2.3, we provide sufficient conditions regarding the geometric structure of the underlying low rank component and outlier points required to ensure the outlier detection method leads to provable outlier detection.

The proposed approach for self-expressive decomposition couples recent results where sparse self-expression used to cluster data into multiple subspaces (subspace clustering) [14, 15, 16, 17] with column sampling approaches used for low rank matrix approximation [18, 19, 20, 21]. Thus, the contributions of this thesis wed two disparate but highly related areas of study and highlights links between new results in both areas. We believe that this study will open up a new and fruitful discussion between both fields of study in the future.
1.3.2 Part 2: Methods for Subspace Clustering

The second major contribution of this thesis is the development of methods and theory for *subspace clustering* or the segmentation of a collection of data in accordance with the subspace membership of the points. Subspace clustering can be used to segment data that admit multi-subspace structure and is also an essential first step required to learn a union of subspace model for a dataset. In the latter case, one clusters the data based upon their subspace membership and then a low rank model is learned for each subset of points by using PCA or a robust variant.

In Chap. 6, we develop new methods for subspace clustering that are built upon the decompositions introduced in the first part of the thesis and thus, we show that SEED decompositions provide a unified approach for both dimensionality reduction and clustering. We demonstrate how a SEED can be used in conjunction with both consensus and spectral clustering-based methods for segmenting the data; in theory and numerical experiments, we demonstrate the advantages of greedy sparse recovery methods such as orthogonal matching pursuit [22] and our subsampling-driven approach for subspace clustering.

In Chap. 7, we develop new theory and geometric insights into the behavior of self-expressive sparse recovery from union of subspaces. In particular, we develop guarantees that describe when the sparse representation of a signal in a particular subspace will only consist of signals from the same subspace, a property that refers to as *exact feature selection* (EFS). When EFS occurs for all of the signals that lie in a subspace, this ensures that no false edges will link signals from different subspaces in the resulting subspace affinity graph—this property is important for guaranteeing that subspace clustering is possible with both consensus and spectral-clustering approaches.

The specific contributions of this part of the thesis are provided below.
• In Sec. 6.3, we introduce a spectral clustering- and consensus-based strategy for clustering data with the representations obtained via greedy recovery methods developed in Chap. 5.

• In Sec. 6.4.1, we provide a spectral clustering approach for clustering low-dimensional factorizations of data from their sparse representations of the data based upon a small reference set of signals that are selected via oASIS. This co-clustering approach simultaneously determines the labels for selected and unselected signals from the dataset by posing this graph clustering problem as a bi-partite graph partitioning problem and thus can cluster data with respect to a lower-dimensional representation.

• In Sec. 7.1, we analyze greedy feature selection strategies and provide sufficient conditions for OMP to produce sparse representations that consist of other points from the same subspace. This property of greedy feature selection is essential for ensuring that both our spectral clustering and consensus method will reliably detect subspace structures in the dataset.

### 1.3.3 Part 3: Structure-Aware Distributed Computing

The third major contribution of this thesis is the development of a data-aware pipeline for distributed processing of massive datasets that admit multi-subspace structure. We demonstrate how sparse self-expressive decompositions provide a scalable strategy for decomposing massive datasets that admit multi-subspace structure and how this decomposition can be utilized to perform distributed matrix computations on the dataset.

The specific contributions of this part of the thesis are as follows:

• In Chap. 8, we propose the RankMap framework, a suite of domain-specific
transformations and distributed interfaces that enable executing a large set of popular knowledge extraction algorithms on big and dense data. Based upon learning the domain data, dependency structure, and the properties of the underlying hardware, RankMap provides an optimized resource-aware mapping of the computation onto the distributed setting.

- In Sec. 8.3, we introduce a novel scalable transformation which maps a dense and structured data matrix onto two matrices which contain much fewer number of non-zeroes. The resulting decomposition is a building block for several important computational applications that exploit data dependencies.

- In Sec. 8.4, we propose an efficient mapping of the iterative computation on the decomposed data onto the processing fabric. It considers the physical limits and flexibility of the underlying processing architecture. Two different models (graph-based and matrix-based) to represent and carry out iterative computations on the transformed domain data have been developed. Each of these models have their own advantages and would return different performances depending on the domain data and its sparsity level.

- In Sec. 8.5, we implement distributed interfaces for each of the graph-based and matrix based models and develop a highly efficient partitioning method with a constant runtime. Performance bounds are provided on the memory usage, computation cost, and communication overhead as a function of the domain data dependency structure.

- In Sec. 8.6, we perform proof-of-concept evaluations of RankMap on a real-world massive light field reconstruction application as well as various large-scale synthetic datasets. The results demonstrate up to 2 orders of magnitude improvement in memory usage and runtime speed.
We evaluate our optimized C++ implementation of the RankMap framework on the Amazon Elastic Cloud (EC2) computing service and the IBM iDataPlex computer cluster. Our experiments utilize up to 244 cores on 12 large computing nodes.

1.4 Motivating Applications

To demonstrate the utility of the methods developed over the course of this thesis, we provide extensive numerical simulations on both synthetic and real-world datasets arising in numerous imaging and computer vision applications. The motivating applications and real-world datasets that we consider over the course of the thesis are described below.

- **Face Recognition Under Varying Illumination Conditions**: Recognition of faces from images collected under varying illumination conditions is an important but challenging problem in computer vision because often the photometric variability present in images renders many approaches based upon Euclidean and other standard distance measures ineffective. We study the application of SEED for dimensionality reduction of face images and also for clustering these images in order to perform face recognition.

- **Light Field Data Compression**: In numerous imaging and video applications, small patches of an image collected either over time (video) or from slightly different viewing angles/positions (camera arrays) can be modeled as living on a low-dimensional subspace. Such datasets more generally are referred to as *light field data* because multi-camera arrays or video sequences convey information about how the light passes in every direction and point in space and/or time. After acquiring a light field data from a multi-camera array or light field camera (e.g., the Lytro camera), a number of post-processing tasks can be performed
to refocus the image at different depths to zoom in on different features in the image. We study the application of SEED for dimensionality reduction (compression) and distributed processing of massive light field datasets.

- **Motion Segmentation:** Motion segmentation is an important yet challenging problem in computer vision where one aims to segment different rigid body motion trajectories from one another directly from video sequences. Each trajectory may correspond to the motion of a different object or even the motion introduced from the camera. It can be shown that rigid body motion arising from point correspondences in multiple affine views live on a 5-dimensional affine hyperplane embedded in the ambient dimension. Thus, when multiple rigid body motions are combined within the field of view, the problem of motion segmentation boils down to learning subspaces from point correspondences and then segmenting the data in accordance with these learned hyperplanes.

- **Clustering and Compression of Hyperspectral Image Data:** Remote sensing data collected from satellite and unmanned aerial vehicles (UAVs) is an extreme example of big multi-dimensional data. In contrast to standard image sensors that acquire information in three spectral bands (red, green, and blue), hyperspectral imaging systems split the spectrum into hundreds of bands and thus can be used to characterize the chemical composition and materials of the underlying scene. In a number of applications, being able to detect even small differences in the chemical composition of crops or vegetation is important for tracking and monitoring agriculture production, yield, crop health, and potentially detect early signs of infestation or other trends and interactions between different insect populations with crops. We study the application of SEED for compression and clustering of hyperspectral image data arising in remote sensing applications in order to segment a scene based upon the different materials
present in the field-of-view.

1.5 Organization

We now provide a detailed roadmap of the organization of the thesis.

Chapter 2. The goal of this Chapter is to provide background on low-dimensional signal models. We begin with preliminaries and notation, review low-dimensional signal models, and introduce relevant definitions and metrics for characterizing geometric structure in data.

Chapter 3. The goal of this Chapter is to review sparse signal models, sparse recovery methods, and theory for exact recovery of sparse signals with convex optimization and greedy recovery methods.

Chapter 4. The goal of this Chapter review approaches and methods for matrix factorization and graph decompositions.

Chapter 5. The goal of this Chapter is to introduce our approach to sparse self-expressive decompositions. We introduce greedy sparse recovery methods for self-expression, a novel algorithm for column subset selection (oASIS), and a sparsity-driven method for outlier detection. To demonstrate the utility of our self-expression for matrix factorization, we provide numerical experiments that highlight the advantages of oASIS for dimensionality reduction and matrix recovery.

Chapter 6. The goal of this Chapter is to demonstrate the utility of self-expressive decompositions and greedy sparse recovery for subspace clustering. We introduce new algorithms for subspace clustering and hybrid linear modeling and then evaluate these methods for clustering numerous real-world and synthetic datasets.
Chapter 7. The goal of this Chapter is to develop a number of theoretical results for the methods developed in Chap. 5-6. We introduce a new geometric analysis for exact feature selection for subspace clustering that provides insight into the geometric conditions on subspaces in the data required to ensure we can reliably recover subspaces with greedy sparse recovery. We show that self-expressive decompositions are rank revealing and the sparsity of the decomposition is bounded by the maximum subspace dimension; in contrast, outliers can be guaranteed to admit a dense a representation and we use this fact to prove a result for the outlier detection strategy presented in Chap. 5. In addition, we analyze the performance of oASIS for exact matrix recovery and show that this sampling method provides a highly efficient strategy for column subset selection.

Chapter 8. The goal of this Chapter is to introduce a novel data-aware distributed processing pipeline that exploits multi-subspace structure in data. The methods presented in this Chapter couple self-expressive decompositions with a number of new methods for graph partitioning and distributed computing to accelerate large-scaled distributed matrix computation important in many signal processing and machine learning applications.

Chapter 9. The goal of this Chapter is to discuss the results, contributions, and implications of the methods and theory developed over the course of the thesis.
2.1 Preliminaries

Before proceeding, we quickly provide notation and review preliminary concepts that will be relevant to the results developed over the course of this thesis.

2.1.1 Vector Spaces

In this thesis, we will treat signals as vectors in a $N$-dimensional Euclidean space, denoted by $\mathbb{R}^N$. We will denote the standard inner product between two vectors in $\mathbb{R}^N$ as

$$\langle x, y \rangle = x^T y = \sum_{i=1}^{N} x(i)y(i).$$

The standard $p$-norm is defined as

$$\|x\|_p = \left( \sum_{j=1}^{n} |x(j)|^p \right)^{1/p},$$

where $p \geq 1$. The $\ell_0$ quasi-norm of a vector $x$, $\|x\|_0$ is defined as the number of non-zero elements in $x$. 
The key matrix norms that we will employ in our subsequent analysis include:

- The matrix norm $\|A\|_{1,1}$ equals the maximum $\ell_1$-norm across all the columns in $A$.
- The spectral norm $\|A\|_{2,2}$ equals the maximum singular value of $A$.
- The Fröbenius norm $\|A\|_{F} = \sum_{i,j} A_{ij}^2$ and is equivalent to the $\ell_2$-norm of the singular values of $A$.

### 2.1.2 Singular Value Decomposition

Recall that a rank $r$ matrix $X \in \mathbb{R}^{M \times N}$ can be expressed as

$$A = \sum_{i=1}^{r} \sigma(i) u_i v_i^T,$$

where $u_i$ and $v_i$ are the $i^{th}$ left and right singular vectors of $A$ respectively and both sets of principal vectors $\langle u^{(i)}, u^{(j)} \rangle = 0, \forall i \neq j$. Stacking the left singular vectors into the columns of $U$ and the right singular vectors into the columns of $V$, we can write the singular value decomposition (SVD) of $A$ as $A = U\Sigma V^T$.

### 2.1.3 Pseudoinverse and Projectors

We denote the right pseudoinverse of a matrix $A$ as $A^+$, where $AA^+ = I$. If $A = U\Sigma V^T$ then $A^+ = V\Sigma^+ U^T$, where we obtain $\Sigma^+$ by taking the reciprocal of the entries in $\Sigma$, leaving the zeros in their place, and taking the transpose of this matrix. An orthonormal basis (ONB) is known to satisfy the following two properties, $\Phi_i^T \Phi_i = I_k$, and $\text{range}(\Phi_i) = S_i$, where $I_k$ is the $k \times k$ identity matrix. The orthogonal projector onto the subspace spanned by the sub-matrix $X_S$ is defined as $P_S = X_S X_S^+$. We will
denote the orthogonal projection of a matrix $A$ onto the span of the sub matrix $X_S$ as $\pi_{X_S}(A)$ and when appropriate, we will write this projection as $\pi_S(A)$ for short.

### 2.1.4 Notation

Throughout this thesis, we will write vectors $x$ in bold lowercase script and matrices $A$ in bold uppercase script. Let $A_{ij}$ denote the $(i, j)$ entry of the matrix $A$. For a matrix $A \in \mathbb{R}^{M \times N}$ and an associated index set $S \subset \{1, 2, \ldots, N\}$, where $|S| = L < N$, we write the submatrix corresponding to selecting the columns in $A$ indexed by the set $S$ as $A_S \in \mathbb{R}^{M \times L}$. We use the notation $A_{-S}$ to denote the set of columns from $A$ excluding the set $S$ and $A_{-i}$ denotes all columns of $A$ but the $i^{th}$ column. The support of a vector $x$, often written as $\text{supp}(x)$, is the set containing the indices of its non-zero coefficients. Let $[A \; B]$ denote the columnwise concatenation of the matrices $A$ and $B$. We represent the element-wise product of matrices $A$ and $B$ as $A \circ B$, and $\text{colsum}(A)$ denotes a row vector containing the sums of the columns of $A$. When describing algorithms, we use “Matlab” style indexing in which $G(i, j)$ denotes the $(i, j)$th entry of the matrix $G$, and $G(:, j)$ denotes its $j^{th}$ column. Let $I_k$ denote a $k \times k$ identity matrix which is a diagonal matrix with ones on its diagonal.

### 2.2 Signal Models

In this section, we review a number of low-dimensional signal models for collections of signals. In Fig. 2.1, we provide a visualization of different types of low-dimensional geometric models for data. On the left, we show linear and nonlinear global signal models and on the right we show mixtures or unions of linear (top) and nonlinear (bottom) signal models.
2.2.1 Linear Subspace Models

Linear subspace models are one of the most widely used signal models for collections of high-dimensional data, with applications throughout signal processing, machine learning, and the computational sciences. This is due in part to the simplicity of linear models but also due to the fact that principal components analysis (PCA) provides a closed-form and computationally efficient solution to the problem of finding an optimal low-rank approximation to a collection of data (an ensemble of signals in $\mathbb{R}^n$). More formally, if we stack a collection of $d$ vectors (points) in $\mathbb{R}^n$ into the columns of $Y \in \mathbb{R}^{n \times d}$, then PCA finds the best rank-$k$ estimate of $Y$ by solving

\[
\text{(PCA)} \quad \min_{X \in \mathbb{R}^{d \times N}} \sum_{i=1}^{d} \sum_{j=1}^{N} (Y_{ij} - X_{ij})^2 \quad \text{subject to} \quad \text{rank}(X) \leq k, \quad (2.1)
\]
where \( X_{ij} \) is the \((i,j)\) entry of \( X \).

The best rank-\( k \) approximation of \( A \) is given by \( A_k = U_k \Sigma_k V_k^T \), where \( U_k \in \mathbb{R}^{m \times k} \) and \( V_k \in \mathbb{R}^{N \times k} \) are the truncated singular vectors (first \( k \) columns of \( U \) and \( V \)) of \( A \) and \( \Sigma_k \in \mathbb{R}^{k \times k} \) contains the first \( k \) singular values of \( A \) along its diagonal. The rank equals the number of nonzero singular values or \( \text{rank}(A) = \|\sigma\|_0 \). The spectral norm of the approximation of \( A \) by its leading \( k \) singular vectors is \( \|A - A_k\|_2 = \sigma(k + 1) \). The truncated SVD also provides the solution to principal components analysis (PCA), which seeks to find a \( k \)-dimensional subspace that best approximates \( A \) in the least-squares sense, i.e., find an orthonormal basis (ONB) \( U \in \mathbb{R}^{m \times k} \) that minimizes \( \|A - UU^T A\|_F \). Thus, the subspace spanned by \( U \) can be interpreted as the best \( k \)-dimensional linear subspace approximation to \( A \), in the least-squares sense.

### 2.2.2 Unions of Subspaces

In many cases, a linear subspace model is sufficient to characterize the intrinsic structure of an ensemble; however, in many emerging applications, a single subspace is not enough to capture the low-dimensional structure of the dataset. Instead, we can consider a generalization of a low rank model to the case where different subsets of signals in \( A \) lie on different low dimensional subspaces. Ensembles ranging from collections of images taken of objects under different illumination conditions [23], motion trajectories of point-correspondences [3], to structured sparse and block-sparse signals [6] are all well-approximated by a union of low-dimensional subspaces or a union of affine hyperplanes. Union of subspace models have also found utility in the classification of signals collected from complex and adaptive systems at different instances in time, e.g., electrical signals collected from the brain’s motor cortex [8].

Formally, we say that a signal \( x \in \mathbb{R}^M \) lies on a union of \( p \) subspaces of \( \mathbb{R}^M \) when
\( x \in \mathcal{U} \), where
\[
\mathcal{U} = \bigcup_{i=1}^{p} \mathcal{S}_i,
\]
and \( \mathcal{S}_i \) is a linear subspace of \( \mathbb{R}^M \). When a collection of data \( \mathbf{X} \) lies on an union of subspaces, where \( k_i = \dim(\mathcal{S}_i) \), the \( \dim(\bigcup_{i=1}^{p} \mathcal{S}_i) \leq \sum_{i=1}^{p} k_i \). Equality is attained when the subspaces are all \textit{independent}. A pair of subspaces are considered to be \textit{disjoint} if they only intersect at the origin and the minimum \textit{principal angle} between the subspaces is greater than zero. We provide a thorough review of principal angles in Sec. 2.3.2.

**A Generative Model for Unions of Subspaces**

We now introduce a generative model for unions of subspaces, where we refer to the set of points sampled from a particular subspace as a “subspace cluster”. Given a set of \( p \) subspaces of \( \mathbb{R}^M \), \( \{\mathcal{S}_1, \ldots, \mathcal{S}_p\} \), we generate a “subspace cluster” by sampling \( N_i \) points from the \( i^{th} \) subspace \( \mathcal{S}_i \) of dimension \( k_i \leq k \). Let \( \tilde{\mathcal{X}}_i \) denote the set of points that lie in the \( i^{th} \) subspace cluster and let \( \tilde{\mathcal{X}} = \bigcup_{i=1}^{p} \tilde{\mathcal{X}}_i \) denote the union of \( p \) subspace clusters. Let \( \mathcal{X}_{-i} = \mathcal{X} \setminus \mathcal{X}_i \) denote the set of points in \( \mathcal{X} \) with the points in the \( i^{th} \) subspace cluster \( \mathcal{X}_i \) excluded. Let \( \mathbf{X} = [\mathbf{X}_1 \ \mathbf{X}_2 \ \cdots \ \mathbf{X}_p] \) be a data matrix containing the points drawn from the union of \( p \) subspaces, where each point in \( \mathcal{X}_i \) is stacked into the columns of \( \mathbf{X}_i \in \mathbb{R}^{M \times N_i} \). The columns of \( \mathbf{X}_i \) can be expanded \( \mathbf{X}_i = \Phi_i \mathbf{C}_i \) in terms of an ONB \( \Phi_i \in \mathbb{R}^{M \times k_i} \) that spans \( \mathcal{S}_i \) and subspace coefficients \( \mathbf{C}_i = \Phi_i^T \mathbf{X}_i \). Let \( \mathbf{X}_{-i} \) denote the matrix containing the points in \( \mathbf{X} \) with the sub-matrix \( \mathbf{X}_i \) excluded.
2.3 Metrics for Characterizing Geometric Structure in Data

We now introduce metrics that aim to characterize the geometric structure amongst signals within a collection of data.

2.3.1 Coherence and Cumulative Coherence

The mutual coherence $\mu(A, B)$ between two matrices with normalized columns, $A \in \mathbb{R}^{M \times N_1}$ and $B \in \mathbb{R}^{M \times N_2}$, is defined as the maximum entry of $A^T B$. Likewise, the coherence of a matrix $A$ with normalized columns is the maximum off-diagonal entry of the Gram matrix $A^T A$. We provide a formal definition of the maximum coherence for later reference.

**Definition 1 (Maximum coherence)** The maximum coherence of a matrix $A$ with normalized columns is defined as

$$\mu(A) = \max_{i \neq j} |(A^T A)_{ij}|.$$

**Definition 2 (Cumulative coherence)** The cumulative coherence of a dictionary of unit-norm atoms $D = \{a_i\}_{i=1}^N$, is defined as

$$\mu_1(k) = \max_{|\Lambda| \leq k} \max_{i \notin \Lambda} \sum_{j \in \Lambda} |\langle a_i, a_j \rangle|.$$

Whereas the maximum coherence describes the maximum amount of coherence that exists between two atoms in the dictionary, the cumulative coherence measures the accumulation of coherence between a fixed atom and $k$ other atoms in the dictionary. Thus, the coherence provides a bound for the cumulative coherence, $\mu_1(k) \leq k\mu$, where $\mu$ denotes the maximum coherence of the dictionary.
2.3.2 Principal Angles

To characterize the “distance” between pairs of subspaces in the ensemble, the principal angles between subspaces will prove useful. The first principal angle $\theta_0$ between subspaces $S_1$ and $S_2$ of dimension $k_1$ and $k_2$ is defined as the smallest angle between a pair of unit vectors $(u_1, v_1)$ drawn from $S_1 \times S_2$. The vector pair $(u_1^*, v_1^*)$ that attains this minimum is referred to as the first set of principal vectors. Formally, the first principal angle is defined as follows.

**Definition 3 [Principal Angle]** The first principal angle between subspaces $S_1$ and $S_2$ of dimension $k_1$ and $k_2$ respectively, is the smallest angle between a pair of unit vectors $(u_1^*, v_1^*)$ drawn from $S_1 \times S_2$,

$$\theta_1 = \min_{u \in S_1, v \in S_2} \text{arccos} \langle u, v \rangle \quad \text{subject to} \quad \|u\|_2 = 1, \|v\|_2 = 1. \tag{2.2}$$

The second principal angle is defined much like the first, except that the second set of principal vectors that define the second principal angle are required to be orthogonal to the first set of principal vectors $(u_1^*, v_1^*)$. The remaining principal angles are defined recursively in this way. The sequence of $k = \min(k_1, k_2)$ principal angles, $\theta_1 \leq \theta_2 \leq \cdots \leq \theta_k$, is non-decreasing and all of the principal angles lie between $[0, \pi/2]$.

The recursive definition above provides insight into what the principal angles/vectors tell us about the geometry underlying a pair of subspaces; in practice, however, the principal angles are not computed in this recursive manner. Rather, a computationally efficient way to compute the principal angles between two subspaces $S_1$ and $S_2$ is to first compute the singular values of the matrix $G = \Phi_1^T \Phi_2$, where $\Phi_1 \in \mathbb{R}^{M \times k_1}$ is an ONB that spans subspace $S_1$. Let $G = U \Sigma V^T$ denote the SVD of $G$ and let $\sigma \in [0, 1]^k$ denote the singular values of $G$, where $k = \min(k_1, k_2)$ is the minimum
dimension of the two subspaces. The \( m^{th} \) smallest principal angle \( \theta_m \) is related to the \( m^{th} \) largest entry of \( \sigma \) via the following relationship, \( \cos(\theta_m) = \sigma(m) \). For our subsequent discussion, we will refer to the singular values of \( G \) as the *cross-spectra* of the subspace pair \((S_1, S_2)\). In the sequel, we will also write the cross-spectra for the subspace pair \((S_i, S_j)\) as \( \sigma_{ij} \).

A pair of subspaces is said to be *disjoint* if the minimum principal angle is greater than zero. Non-disjoint or intersecting subspaces are defined as subspaces with minimum principal angle equal to zero. The dimension of the intersection between two subspaces is equivalent to the number of principal angles equal to zero or equivalently, the number of entries of the cross-spectra that are equal to one. We define the *overlap* between two subspaces as the \( \text{rank}(G) \) or equivalently, \( q = \|\sigma\|_0 \), where \( q \geq \dim(S_1 \cap S_2) \).

Whereas the mutual coherence provides a measure of the similarity between a pair of unit norm vectors that are contained in two discrete sets \( X_1 \) and \( X_2 \), the cosine of the minimum principal angle provides a measure of the similarity between all pairs of unit norm vectors that lie in the span of \( S_1 \times S_2 \). For this reason, the cosine of the first principal angle provides an upper bound on the mutual coherence. The following upper bound is in effect for each pair of subspace clusters in the ensemble:

\[
\mu(X_1, X_2) \leq \cos(\theta_1),
\]

(2.3)

where \( X_1 \) and \( X_2 \) contain the signals in \( X \) that lie in \( S_1 \) and \( S_2 \) respectively and \( \theta_1 \) is the minimum principal angle between \( S_1 \) and \( S_2 \).
2.3.3 Covering Radius

To measure how densely each subspace is covered by data points that lie in its span, we will study the covering radius of a set of normalized points relative to the subspace they span.

**Definition 4 (Covering Radius)** The covering radius of the set of $N$ unit-norm points $\mathcal{X} = \{x_i\}_{i=1}^N$ along subspace $\mathcal{S}$ is defined as

$$\text{cover}(\mathcal{X}, \mathcal{S}) = \max_{u \in \mathcal{S}} \min_{x \in \mathcal{X}} d(u, x),$$

where $d(u, x) = \sqrt{1 - |\langle u, x \rangle|}$.

The covering radius of the normalized subspace cluster $\mathcal{X}$ can be interpreted as the size of the largest open ball that can be placed in the set of all unit norm vectors that lie in the span of $\mathcal{S}$, without touching a point in the set $\mathcal{X}$. We will also write the covering radius as $\text{cover}(D, \mathcal{S})$, where $D$ is a matrix with normalized columns.

Let $(u^*_1, x^*_1)$ denote a pair of points that attain the maximum covering diameter for $\mathcal{X}_1$; $u^*_1$ is referred to as a deep hole in $\mathcal{X}_1$ along $\mathcal{S}_1$. The covering radius can be interpreted as the sine of the angle between the deep hole $u^*_1 \in \mathcal{S}_1$ and its nearest neighbor $x^*_1 \in \mathcal{X}_1$. We show the geometry underlying the covering radius in Fig. 2.2.

A related quantity is the inradius of the set $\mathcal{X}$, or the cosine of the angle between a point in $\mathcal{X}$ and any point in $\mathcal{S}$ that attains the covering radius. The relationship between the covering radius and inradius $r(D, \mathcal{S})$ is given by

$$r(D, \mathcal{S}) = \sqrt{1 - \text{cover}(D, \mathcal{S})^2}.$$  \hfill (2.4)

A geometric interpretation of the inradius is that it measures the distance from the origin to the maximal gap in the antipodal convex hull of the columns of $D$ that lie
Figure 2.2: Covering radius of points in a normalized subspace. The interior of the antipodal convex hull of points in a normalized subspace—a subspace of $\mathbb{R}^n$ mapped to the unit $\ell_2$-sphere—is shaded. The vector in the normalized subspace (unit circle) that attains the covering radius (deep hole) is marked with a star: when compared with the convex hull, the deep hole coincides with the maximal gap between the convex hull and the set of all vectors that live in the normalized subspace.

along $S$. The geometry underlying the covering radius and the inradius is displayed in Fig. 2.2.

2.3.4 Matrix Spark

The spark provides a worst-case measure of the number of points required to form a linearly dependent set, for all possible subsets of columns in the matrix. Note that since this property of the data is independent of the specific sparse recovery algorithm used, this proposition is of broad interest for the application of both convex optimization based algorithms and greedy methods as we propose here.
Figure 2.3:  *Covering radii in the ambient space.* A union of two intersecting 2D subspaces in $\mathbb{R}^3$. The covering radius between two points in a single subspace is shown and the point that attains the covering radius (deep hole) is also highlighted, the subspace intersection is surrounded by an $\epsilon$-tube.

While the spark provides valuable intuition into the geometry of a signal ensemble, in practice, computing the spark of a matrix requires a combinatorial search. For instance, to test whether the spark of $D$ is less than or equal to $k$, we must search over all $\binom{N}{k}$ submatrices of $D$. Instead, we can bound the spark by the coherence. Intuitively, if the columns of a matrix contain incoherent columns, then even large submatrices of $D$ will be full rank or equivalently, the matrix will have a large spark. The following lemma [24] makes this idea precise.

**Lemma 1** [24] Let $D$ be a $M \times N$ matrix with normalized columns and Gram matrix
$G = D^T D$. If the maximum cumulative coherence

$$\max_{|\Lambda| = r} \sum_{i \in \Lambda} |G_{ij}| < 1, \; \forall j = \{1, \ldots, N\}$$

then $r \leq \text{spark}(D)$.

A corollary of Lemm. 1 is obtained by assuming a uniform distribution of the off-diagonal entries of $G$.

**Corollary 1 [24]** Let $\mu$ denote the maximum coherence between the columns of $D$. The $\text{spark}(D) \geq \mu^{-1}$.

In contrast to the rank, which tells us the largest set of linearly independent vectors in a matrix, the spark of a matrix tells us the smallest number of columns that we can select before any set of columns are linearly dependent.
In this Chapter, we provide background on signal representation, sparse approximation, sparse recovery methods, and the geometry underlying exact recovery conditions for sparse recovery methods.

### 3.1 Signal Representation

In many applications, it is advantageous to “re-represent” a signal of interest in a new space or domain. Take for example a time-domain signal composed of two sinusoidal components, as shown on the left in Fig. 3.1. Upon inspection, it appears that signal is composed of a number of periodic sinusoidal components, however, determining the precise frequencies that this signal consists of would be a nontrivial task if we were to stay in the time domain. If we instead take the Fourier transform of the signal, the frequency content becomes glaringly obvious. This idea is demonstrated in Fig. 3.1.

To find a signal’s frequency representation, we project our input signal $\mathbf{y} \in \mathbb{R}^M$ onto a collection of $M$ basis vectors, $\mathcal{D} = \{\mathbf{d}_m\}_{m=0}^{M-1}$, where $\mathbf{d}_m(k) = e^{-j2\pi km/N}$ for $k = \{0, \ldots, M - 1\}$. We will refer each of the elements that comprise our representation as an *atom* and a collection of these atoms as a *dictionary*. To simplify notation,
we will drop our dependence on $k$ and denote the $m^{th}$ atom in the dictionary as $d_m$. Generally speaking, a dictionary is defined to be a finite collection of $N$ unit-norm atoms, where $M \leq N$. In the case of the DFT, all of the atoms in the representation are orthogonal and forms an orthonormal basis for $\mathbb{R}^M$, where $M = N$.

Upon stacking each of these atoms into the rows of the DFT analysis matrix $D^*$, we can premultiply our signal $y$ to obtain a new representation of the signal, $x = D^*y$. The analysis matrix may be written as,

$$D^* = \frac{1}{\sqrt{M}} \begin{pmatrix}
1 & 1 & 1 & \ldots & 1 \\
1 & \omega & \omega^2 & \ldots & \omega^{(M-1)} \\
1 & \omega^2 & \omega^4 & \ldots & \omega^{2(M-1)} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & \omega^{M-1} & \omega^{2(M-1)} & \ldots & \omega^{(M-1)(M-1)}
\end{pmatrix}, \quad (3.1)$$

where $\omega$ is the $M^{th}$ root of unity and is written as $\omega = e^{-j2\pi/M}$.

Figure 3.1: Example of a sparse time-frequency representation. On the left is the time-domain representation of a signal consisting of three sinusoidal components and on the right is the frequency representation of the same signal.
When D forms an orthonormal basis, we may obtain an exact reconstruction of our signal \( y \) by synthesizing our signal from the same \textit{atoms} that we used to analyze the signal. Writing this out,

\[
y = \sum_{m=0}^{N-1} \langle d_m, y \rangle d_m
\]

and also in matrix notation,

\[
y = Dx = DD^*y
\]

Since the DFT is an orthonormal expansion, this implies that \( \langle d_m, d_n \rangle = 0, \forall m \neq n \), and 1 otherwise. Due to this orthogonality condition, the matrix product \( DD^* \) is simply the \( M \times M \) identity matrix. Hence, multiplying the signal by \( DD^* \) will give us back exactly \( y \).

Just as \( D^* \) was called our \textit{analysis matrix} (because it allows us to analyze the signal in the new representation), \( D \) is called the \textit{synthesis matrix} because it allows us to synthesize the signal from a set of \textit{atoms} and their corresponding coefficients. Since we are working in \( \mathbb{R}^M \), \( D \) and \( D^* \) are simply the transpose of one another (in complex spaces, they are the conjugate transpose of one another). We have seen that in the case of an orthogonal decomposition, the analysis coefficients obtained by projecting the signal onto the \textit{analysis matrix} are exactly the coefficients used in our expansion. In general, this is not necessarily true. In particular, when redundancy is introduced into the system, the analysis coefficients of an expansion give an idea of which \textit{atoms} the energy of the signal lies within but may not be used to synthesize the signal again. Whereas the Gram matrix \( D^T D = I \) in the case of an orthonormal system, for a \textit{synthesis matrix} with coherent columns, the Gram matrix does not equal the identity. Rather, the Gram matrix provides us with a way to visualize the coherence between all \textit{atoms} in our representation.
3.2 Redundant Expansions

As we pointed to at the end of Sec. 3.1, representing a signal as a linear combination of its analysis coefficients and their corresponding atoms is only possible in the case of an orthonormal expansion. When the number of atoms $N$ is larger than the dimension of the signal, i.e., $M > N$, the resulting representation is considered to be overcomplete. In this case, we must solve an overdetermined system of equations and hence, $D$ no longer provides a unique representation of signals in $\mathbb{R}^M$.

While redundant expansions are no longer unique, there are many cases with which introducing redundancy within a signal expansion may be beneficial. For example, in Fig. 3.3, we show the ringing (modulation) effects in the wavelet representation of oriented edges due to the limitations introduced by using an orthogonal expansion with 2D atoms oriented at $0^\circ$, $45^\circ$, and $90^\circ$. By introducing redundancy into the representation (see Fig. 3.4), we may not only minimize ringing effects but also produce a more compact representation than in the case of the typical orthogonal expansion.

3.2.1 Method of Frames

In order to find a representation of $x$ with respect to an overcomplete dictionary, we must constrain our search in some way. One idea for how to constrain this overdetermined system of equations is to find the coefficient vector with minimum $\ell_2$-norm. This approach is known as the method of frames.

To find the coefficient vector with minimum $\ell_2$-norm, we solve the following optimization problem,

$$\min_x \|x\|_2 \quad \text{subject to} \quad y = Dx. \quad (3.2)$$

The minimizer of this problem is given by the projection of the signal onto the pseudo-inverse of $D$. We can write the resulting coefficient vector as $x = D^\dagger y$, where $D^\dagger = \cdots$
Figure 3.2: Sparse representation of images with wavelets. On the left, we show the cameraman test image and on the right, we show its wavelet decomposition across multiple scales. Notice that most of the coefficients are very small (blue). This sparse representation enables us to compress the original image by simply throwing away all of the coefficients below a particular threshold.

\[(D^*D)^{-1}D^*\]. Accordingly, we can synthesize our original signal as \( y = Dx = DD^\dagger y \). The matrix product \( DD^\dagger = I \) which allows for perfect reconstruction, however, the Gram matrix \( DD^* \) may be far from the identity.

3.3 Sparse Approximation

We now introduce sparse signal models and numerous sparse approximation problems.

3.3.1 Sparse Signal Models

In Sec. 3.1, we saw that one of the advantages of representing a signal in a new space/domain is the ease with which we may extract information from the new representation. This is typically true when the new representation is more compact than
Figure 3.3: *Limitations of orthogonal wavelet decompositions in representing edge content in images.* The wavelet decomposition shown in Figure (3.2), zoomed in. In a standard 2D wavelet representation, wavelets are oriented at $0^\circ$, $45^\circ$, and $90^\circ$, shown on the right from top to bottom, respectively. Looking closely at the wavelet representation of the two prominent oriented edges in the image, we see modulatory effects due to the limitations in the representation’s ability to efficiently represent oriented edges at arbitrary angles.

Figure 3.4: An example of a redundant representation that provides a compact representation of edges.
the representation of the signal in its original form. This notion of compactness can be made precise by measuring the sparsity of a signal with the $\ell_0$ quasi-norm, which simply counts the number of non-zero coefficients in the signal. In our example above, the sparsity of the original signal $\|y\|_0$ is large (close to $N$), as the signal takes on the value of zero at a small number of points. In contrast, the sparsity of the frequency representation $\|x\|_0 = 3$. Hence, this particular transformation results in a substantial reduction in the amount of information (bits) that must be stored to represent the signal. Most compression schemes, such as JPEG-2000 (DWT) and MP3 (DFT) rely on this fact to achieve high compression rates.

In both cases, compression is achieved by computing the analysis coefficients of the signal with respect to a given dictionary, sorting them, and throwing away coefficients that take on small or zero values. An example of the wavelet decomposition of an image is shown in Fig. 3.2 above. As you can see, most of the coefficients in the expansion are very small (cool colors) and only very few coefficients are significant (warm colors).

Sparse representations may also be exploited in the context of denoising. If we assume that our original signal is corrupted with Gaussian noise, we may decompose our signal in the normal way and perform the same thresholding operation that we employed for compressing the signal. If the noise is spread across all the elements in our representation, this thresholding operation has the effect of throwing away all of the noise, as well as small coefficients that correspond to wavelets that are unnecessary to represent our original signal. The idea that unwanted components (noise) in our signal must be spread across the atoms in our expansion will resurface in a later section with which we will refer to this property as the incoherence between the signal components and the atoms in our expansion.
3.3.2 Sparse Approximation Problems

The goal of sparse recovery is to find the minimum number of atoms within an over-complete dictionary $D \in \mathbb{R}^{m \times N}$ that may be linearly combined to obtain a representation of an input signal $y \in \mathbb{R}^m$, where the $i^{th}$ column of $D$, $\|d_i\|_2 = 1$, $\forall i = \{1, \ldots, \ell\}$ and $m \ll N$. Formally, the sparsest representation of $y \in \mathbb{R}^n$ with respect to $D$ is given by

$$x^* = \arg \min_{x \in \mathbb{R}^N} \|x\|_0 \text{ subject to } \|y - Dx\|_2 \leq \delta, \quad (3.3)$$

where $\|w\|_0$ counts the number of non-zeroes in its argument and $\delta$ is a user specified parameter that controls the amount of error in the approximation. Let $S = \text{supp}(x^*)$ denote the support of $x$ and $x^*(j)$ denote the contribution of the $j^{th}$ atom $d_j$ to the representation of $y$. By penalizing representations that require a large number of non-zero coefficients, the resulting representation will be sparse.

Sparsity has proven to be an essential ingredient in the development of efficient and, in some cases, unique solutions to a number of fundamental problems in signal processing and machine learning, from compression and denoising of signals [25, 26] to compressive sensing [27, 28], morphological components analysis [29, 30] and sparse-representation based classification [31]. In all of these settings, sparse recovery methods, e.g., $\ell_1$-minimization [25, 32] or greedy pursuits [22], are employed to recover signals that admit sparse representations in a fixed and pre-specified basis or over-complete dictionary.

The problem of finding a sparse approximation to a signal can be formulated in any of the following ways.

1. Find the sparsest representation of the target signal.

\[(P0) \quad \min_x \|x\|_0 \text{ subject to } y = Dx.\]

2. Given a target signal, find the sparsest coefficient vector that represents the
signal with a particular amount of error.

\begin{align*}
\text{(Error)} \quad \min_x \|x\|_0 \quad \text{subject to} \quad \|y - Dx\|_2^2 \leq \epsilon.
\end{align*}

3. For all coefficient vectors with a fixed sparsity level, find the one with the smallest amount of error.

\begin{align*}
\text{(Sparse)} \quad \min_x \|y - Dx\|^2_2 \quad \text{subject to} \quad \|x\|_0 \leq \delta.
\end{align*}

4. Given a target signal, find a coefficient vector that balances sparsity and approximation error.

\begin{align*}
\text{(Tradeoff)} \quad \min_x \frac{1}{2}\|y - Dx\|_2^2 + \lambda\|x\|_0.
\end{align*}

It has been shown that solving any of these formulations directly is a NP-hard problem. This is in part due to the combinatorial complexity required to conduct an exhaustive search over all possible K-sparse signals (in the case of Sparse) or conducting this search to find the smallest K (to solve P0). Therefore, we must resort to approximations or relaxations of these problems in order to ensure a sparse approximation can be obtained in polynomial time. We will now go over two different types of approaches to numerically solve these problems. The first class of methods are known as greedy pursuits. We hope that the intuition behind these types of procedures will give us some insight into how sparse approximations may be obtained from overcomplete dictionaries.

\section{3.4 Convex Methods for Sparse Recovery}

\subsection{3.4.1 Sparse Recovery Using the $\ell_1$-norm}

Methods for sparse recovery fall broadly into one of two classes, convex optimization-based approaches and greedy pursuit methods. In this section, we describe the first class of methods transform the non-convex objective function in (Exact) into a convex
objective by replacing the $\ell_0$ penalty with the $\ell_1$ norm. This relaxation results in a formulation which is known to as Basis Pursuit (BP),

\[
(BP) \quad \min_x \|x\|_1 \quad \text{subject to} \quad y = Dx.
\]

If we relax the equality constraint in (BP) by allowing some amount of $\ell_2$ approximation error (Error), this results in convex formulations known as basis pursuit denoising (BPDN) [25]. For later reference, we supply the BPDN objective function below,

\[
(BPDN) \quad \min_x \|x\|_1 \quad \text{subject to} \quad \|y - Dx\|_2 < \kappa,
\]

where $\kappa$ is a parameter that is selected based upon the amount of noise in the data.

If we relax (Sparse), this results in a convex formulation known as the LASSO [32]. In this case, our objective can be expressed as

\[
(LASSO) \quad \min_x \frac{1}{2}\|y - Dx\|_2 \quad \text{subject to} \quad \|x\|_1 \leq \tau,
\]

where $\tau$ is determined by the user based upon the assumed sparsity of the signal.

### 3.4.2 Projected Gradient Descent Methods

In this section, we review iterative first order methods for solving a number of convex optimization problems.

**Gradient Descent**

Recall that to solve the following unconstrained minimization problem,

\[
x^* = \arg\min_x f(x),
\]

\[
(3.4)
\]
the gradient descent algorithm requires the following simple iterative update

\[ x_t = x_{t-1} - \eta_t \nabla f(x_{t-1}), \]

where \( f(x) \) is a differentiable convex function that maps \( x \in \mathbb{R}^n \rightarrow \mathbb{R} \), \( x_t \) is the estimate of \( x^* \) at the \( t \)th time step, \( \eta_t \) is the step size at the \( t \)th time step, and \( \nabla f(x_t) \) is the gradient of \( f(x) \) evaluated at \( x_t \). Assuming that \( f(x) \) is twice-differentiable, and the gradient of \( f(x) \) is Lipschitz with constant \( L > 0 \), i.e., \( \| \nabla f(x) - \nabla f(y) \|_2 \leq L \| x - y \|_2 \), \( \forall x, y \), then one can show that

\[ f(x_t) - f^* \leq \frac{\| x_0 - x^* \|_2^2}{2 \eta t}, \]

where \( f^* = f(x^*) \) is the objective evaluated at the minimizer of Equation (3.4) \( x^* \), and \( \eta_t < 1/L \). Thus, asymptotic convergence to the optimal estimate is guaranteed, as long as the step size at each iteration \( \eta_t \) is sufficiently small.

**Projected Gradient Descent**

Gradient descent cannot be used on problems with non-differentiable cost functions or constrained minimization problems with non-smooth constraint sets. Examples of non-differentiable functions that are used as regularizers for problems with differentiable cost functions \( g(x) = \|x\|_1 \) (\( \ell_1 \)-norm used to promote sparse representations), indicator function \( g(x) = 1_C \), where \( C \) is a convex set and \( g(x) = \infty \) when \( x \notin C \).

To solve such problems, *projected gradient descent* (PGD) can be used in place of gradient descent. In this case, to solve the following problem

\[ \min_x f(x) + \lambda g(x), \]
where $f(x)$ is differentiable and convex (as before) and $g(x)$ is non-differentiable, generalized gradient descent considers minimizing a quadratic approximation of $f(x)$ at $x_0$ (as in gradient descent) added to its non-differentiable counterpart, $g(x)$. The estimate at the $t^{th}$ time step is given by

$$
x_{t+1} = \arg \min_x P(x_t - \eta \nabla f(x_t)),
$$

where $P_\lambda(x) = \arg \min_z \|x - z\|_2 + \lambda g(z)$.

We now provide a list of different non-differentiable functions and their corresponding proximal operator.

- **Soft Thresholding**: The proximal operator for $g(x) = \|x\|_1$ is a soft-thresholding function that sets all entries below the threshold $\lambda$ to zero and shrinks all remaining entries towards zero. To be precise, the soft-thresholding operator is defined as follows

$$
P_\lambda(x(i)) = \begin{cases} 
0 & \text{if } |x(i)| \leq \lambda \\
x(i) & \text{if } x(i) > \lambda \\
x(i) & \text{if } x(i) < \lambda.
\end{cases}
$$

- **Non-negativity Constraint**: The proximal operator for $g(x) = 1_C$, where $C = \{x : x \geq 0\}$ is defined as

$$
P(x(i)) = \begin{cases} 
x(i) & \text{if } x(i) > 0 \\
0 & \text{otherwise}.
\end{cases}
$$
• **Hard Thresholding:** The hard thresholding operator is defined as follows

\[
P_\lambda(x(i)) = \begin{cases} 
0 & \text{if } |x(i)| \leq \lambda \\
 x(i) - \lambda & \text{if } x(i) > \lambda \\
 x(i) + \lambda & \text{if } x(i) < \lambda.
\end{cases}
\] (3.8)

Projected gradient descent with the hard thresholding operator is guaranteed to converge to a local minimum of the sparse coding objective (P0), where \( f(x) = \|y - Dx\|_2^2 \) and \( g(x) = 1_C \) is an indicator function for the non-convex set \( C = \{x : \|x\|_0 \leq k\} \), where \( \|x\|_0 \) returns the number of nonzero entries in the vector \( x \).

All of the objective functions listed above can be interpreted via their respective thresholding operations. Conversely, for a given thresholding function, we can often map this thresholding operation back to its respective cost in the underlying optimization problem.

**Iterative Soft Thresholding Algorithm**

The iterative soft thresholding algorithm is a PGD-based method for solving the following sparse recovery objective

\[
x^* = \arg\min_x \|y - Dx\|_2^2 + \lambda \|x\|_1,
\] (3.9)

where \( \lambda \) is a regularization parameter that controls the tradeoff between the cost function given by \( f(x) = \|y - Dx\|_2^2 \) and the non-differentiable regularizer given by \( g(x) = \|x\|_1 \).
The update equation for solving (3.9) is given by the following equation

\[ x^t = P_\lambda(x^{t-1} - \eta D^T(y - Dx^{t-1})) , \]  \hspace{1cm} (3.10)

where we have substituted \( \nabla f(x^{t-1}) = D^T(y - Dx^{t-1}) \) into the PGD update in Eq. 3.5 and \( P_\lambda(x) \) is the soft-thresholding function. This update results in a first-order method known as iterative soft thresholding (IST) [33]. A fast variant of this method known as FISTA is introduced in [34].

If we examine the update rule for the BPDN problem in Eq. 3.9, observe that we can write the update for the \( i \)th variable at the \( t \)th time step as follows:

\[ x^t(i) = P_\lambda[(1 - \eta)x^{t-1}(i) + \eta y^T d_i - \eta \sum_{j \neq i} G_{ji} x^{t-1}(j)] , \]  \hspace{1cm} (3.11)

where \( G_{ji} = d_j^T d_i \) represents the similarity between the \( i \) and \( j \) column of \( D \).

### 3.5 Greedy Methods for Sparse Recovery

The second class of sparse recovery methods employ greedy pursuit strategies to find an approximate solution to either Sparse or Error. Examples of greedy pursuit strategies include matching pursuit (MP) [35], an improved variant called orthogonal matching pursuit (OMP) [22] (Alg. 1), or subspace pursuits such as CoSaMP [36]. Greedy methods work by selecting atoms iteratively, subtracting the contribution of each selected atom from the current signal residual. This selection process is then repeated until a stopping criterion is satisfied: either a target sparsity \( \|x\|_0 = k \) is reached, or the residual magnitude becomes smaller than a pre-specified value.
OMP Algorithm

The OMP algorithm proceeds as follows. At the first step, we find the point that is closest to the input signal $y$ in terms of its angular distance. If we consider this greedy selection for an input signal $y$, we will select the atom from $\mathcal{D} = \{d_j\}_{j=1}^{N}$ that maximizes the following expression

$$j^* = \arg \max_j |\langle y, d_j \rangle|.$$  

After finding the atom that is most correlated with the input signal, we place this index in our feature set $\Lambda = j^*$ and update the residual by removing the projection of $y$ onto $d_j^*$. To be precise, we set the residual to

$$r = y - \langle y, d_j^* \rangle y.$$  

After computing the residual, we will look for the next atom that is maximally correlated with the signal residual,

$$j^* = \arg \max_j |\langle r, d_j \rangle|.$$  

This atom is added to the feature set, $\Lambda = j^* \cup \Lambda$, and the residual is computed by projecting $y$ into the space orthogonal to the subspace spanned by the points in the current feature set. Formally, at the $m^{th}$ step of the algorithm, the residual is computed as

$$r = y - \pi_\Lambda(y),$$  

where $\pi_\Lambda(y)$ denotes the orthogonal projection of $y$ onto the subspace spanned by the submatrix $D_\Lambda$. This greedy selection procedure is repeated $k$ times or until the norm of the signal residual drops below a certain pre-specified threshold.
**Algorithm 1: Orthogonal Matching Pursuit (OMP)**

**Input:** Input signal $x$, dictionary $D$ containing $L$ unit-norm vectors in its columns, the maximum sparsity level $k$, and error tolerance $\epsilon$.

**Output:** Support set $\Lambda$ and $s$-dimensional coefficient vector $v$, where $|\Lambda| = s \leq k$.

**Initialize:** Set the residual $r$ to the input signal $r = x$.
1. Select the column of $D$ that is maximally correlated with $r$ and add it to $\Lambda$
   $$\Lambda \leftarrow \Lambda \cup \arg \max_{j=1,\ldots,N} |\langle d_j, r \rangle|.$$ 
2. Update the residual $r = x - D_\Lambda D_\Lambda^\dagger x$.
3. Repeat steps (1–2) until the norm of the residual $\|r\| \leq \epsilon$ or $|\Lambda| = k$.
4. Return the support set $\Lambda$ and coefficients $v = D_\Lambda^\dagger x$.

Greedy pursuits will serve as the algorithmic framework for our subsequent studies. For this reason, we also detail OMP in Alg. 1 for later reference. Note that this algorithm can be accelerated by using a progressive Cholesky update to compute the pseudoinverse $D_\Lambda^\dagger$ in Step 2 as a new column of $D$ is added to the approximation.

**Complexity of OMP**

In our complexity analysis of OMP, we will assume that a progressive Cholesky update is used to update the pseudoinverse in Step 2 of Alg. 1. For large dictionaries, the complexity of OMP is dominated by computing $D^T r$, which we denote by $T_D$. When we have an explicit representation of $D \in \mathbb{R}^{M\times N}$ then $T_D = 2MN$. At the $n^{th}$ iteration, we must update both the Cholesky decomposition of $G_{\Lambda,\Lambda}$ and the residual, which require $(n-1)^2$ and $2n^2$ operations, respectively. Assuming that the maximum sparsity level $k$ is provided, the complexity of OMP is given by

$$T_{OMP} \approx k^3 + kT_D.$$ 

Thus, when we have an explicit representation of $D$, $T_{OMP} \approx k^3 + kMN$. 
Batch OMP Algorithm

In order to compute the sparse representation of a large set of signals with respect to the same dictionary, the batch OMP method [37] can be employed to accelerate this computation. The idea behind the batch OMP algorithm is to replace the computation of the residual $r$ in Step 2 of the OMP algorithm and its multiplication by $D^T$ with a lower cost computation of $D^T r$. To make this precise, let $G_D = D^T D$, $a = D^T r$, $a^0 = D^T x$, and $\Lambda$ be the current support set. In order to update $a$ at each iteration of OMP, the batch OMP algorithm computes:

$$a = a^0 - G_{\Lambda} (G_{\Lambda,\Lambda})^{-1} a^0_{\Lambda}.$$

Thus, we can pre-compute $G$ and $a^0$ for each signal and then apply the update formula to $a$ for each signal. Additional acceleration is achieved by using a progressive Cholesky factorization to update the inverse Gram matrix $(G_{\Lambda,\Lambda})^{-1}$ as a new column of $D$ is added at each iteration.

The only drawback of updating $D^T r$ rather than the residual is that the norm of the residual (error) cannot be directly computed. In order to use the batch-OMP algorithm to perform error-constrained sparse recovery, an iterative update of the residual error can be used instead. Let $\epsilon^n = \|r^n\|_2^2$ equal the squared error of the residual at the $n$th iteration, $x^n$ be the sparse representation at the $n$th step, and $\delta^n = (x^n)^T G x^n$. The update for the error is given by the following equation

$$\epsilon^n = \epsilon^{n-1} - \delta^n + \delta^{n-1}.$$

With both updates for $a$ and $\epsilon$, the batch OMP algorithm can be used for error-constrained greedy recovery.
3.6 Exact Signal Recovery Conditions

In this section, we will develop further insight into the connections between the dictionary and the behavior of sparse recovery methods.

3.6.1 Sub-dictionaries

A sub-dictionary of $D$ may be expressed as $D_{\Lambda}$, where $\Lambda$ is an index set containing the indices of a subset of columns of $D$. We will often be interested in identifying a sub-dictionary consisting of atoms that take on non-zero coefficients and analyzing this sub-dictionary. Often, the Gram matrix of a sub-dictionary will be well-conditioned but the entire matrix will have large condition number. The maximum and minimum singular values of $DD^*$ provides an estimate of how much $D$ can stretch or compress any signal in $\mathbb{R}^M$. The ratio of maximum to minimum singular values could be very large for the entire dictionary but relatively small for most sub-dictionaries of a particular size.

3.6.2 Exact Recovery Conditions

Given a signal $y$ that can be expressed as a linear combination of atoms belonging to the index set $\Lambda$, both OMP and BP will succeed in recovering the correct atoms and coefficients used to synthesize $y$ provided that

$$\max_{i \notin \Lambda} \|D_{\Lambda}^i d_i\|_1 < 1. \quad (3.12)$$

This condition is known as the Exact Recovery Condition (ERC) [38].

The orthogonal projection of an atom outside of the correct support set $d_m$ onto
the columns of the sub-dictionary $D_{\Lambda}$ is given by

$$P_{\Lambda}d_m = D_{\Lambda}D_{\Lambda}^\dagger d_m.$$ 

The coefficient vector used to synthesize this projection is given by $D_{\Lambda}^\dagger d_m$. Hence, the quantity

$$ERC(\Lambda) = 1 - \|D_{\Lambda}^\dagger d_m\|_1$$

measures how far $P_{\Lambda}d_m$ lies from the antipodal convex hull of the atoms in $\Lambda$.

### 3.6.3 Coherence-Based Recovery Conditions

Although the exact recovery condition provides some intuition about when a signal can be uniquely recovered from a certain sub-dictionary of $D \in \mathbb{R}^{M \times N}$, to guarantee that all $k$-sparse signals can be uniquely recovered, we must ensure that all sub-dictionaries of size $k$ satisfy the condition that $ERC(\Lambda) > 0$. In practice, the exact recovery condition is impossible to check because it requires evaluating $ERC(\Lambda)$ for all $\binom{N}{k}$ sub-dictionaries of size $k$. The explicit conditions for exact recovery in (3.12) can be translated into general recovery conditions by studying the coherence and cumulative coherence of the synthesis dictionary [38].

In [39], Gribonval and Nielsen derive a sufficient condition for exact recovery that depends on set-wise cumulative coherence over the support set and the complement set. In subsequent discussions, we will refer to the set-wise cumulative coherence over the support set as the intra-support coherence, $\mu_1(\Lambda)$, and the coherence over the complement set as the inter-support coherence, $\mu_1(\Lambda^c)$, of the dictionary.

**Theorem 1** [39] *Let a signal be synthesized from atoms in the sub-dictionary $D_{\Lambda}$.***
If the Neumann ERC,

$$\mu_1(\Lambda) + \mu_1(\Lambda^c) = \max_{n \in \Lambda} \sum_{m \in \Lambda, m \neq n} |\langle d_m, d_n \rangle| + \max_{n \in \Lambda^c} \sum_{m \in \Lambda} |\langle d_m, d_n \rangle| < 1, \quad (3.13)$$

holds then OMP will recover the atoms in $\Lambda$ exactly.

These recovery conditions are valid only for a particular support set $\Lambda$; if we seek success support recovery with OMP over all $k$-sparse combinations of atoms from $D$, we can instead bound the inter- and intra-support coherence by the cumulative coherence of the dictionary. Recall that the cumulative coherence defined as

$$\mu_1(k) \equiv \max_{|\Lambda|=k} \max_{m \in \Lambda^c} \sum_{n \in \Lambda} |\langle d_n, d_m \rangle| \leq k\mu. \quad (3.14)$$

**Theorem 2** [38] Let $\mu_1(k)$ be the cumulative coherence of a dictionary $D$. Exact support recovery is guaranteed for all $k$-sparse signals with both OMP and BP if

$$\mu_1(k - 1) + \mu_1(k) < 1.$$

Finally, if we simplify the cumulative coherence in terms of the maximum coherence, $k\mu + (k - 1)\mu < 1$. This implies that $k < \frac{1}{2}(1 + \mu^{-1})$. This condition implies that ERC holds whenever the coherence $\mu$ is upper bounded by twice the inverse of the sparsity level $k$. 

In this Chapter, we provide background on matrix factorization as well as alternating and approximate methods for matrix factorization that can be utilized in large-scale learning.

## 4.1 Matrix Factorization

The idea underlying matrix factorization is to decompose $X$ into at least two factor matrices, where $X = UV$. In the case where $X$ is of rank equal to $r$, the SVD decomposes $X = U\Sigma V^T$, where $U$ and $V$ contain $r$ orthogonal factors and $\Sigma$ is a diagonal matrix containing the singular values of $X$.

### 4.1.1 Principal Components Analysis

The objective underlying principal components analysis (PCA) can be expressed as a rank constrained minimization problem:

$$
\minimize_{Y \in \mathbb{R}^{M \times N}} \|X - Y\|_F \quad \text{subject to} \quad \text{rank}(Y) \leq k. \quad (4.1)
$$
A closed-form solution to (PCA) is provided by the truncated SVD decomposition, where the best rank $k$ approximation to $X$ is given by $\hat{X} = U_k \Sigma_k V_k$, $U_k \in \mathbb{R}^{m \times k}$ contains the $k$ leading left singular vectors of $X$, $\Sigma_k \in \mathbb{R}^{k \times k}$ is a diagonal matrix containing the $k$ leading singular values of $X$ along its diagonal, and $V_k \in \mathbb{R}^{k \times N}$ contains the leading $k$ right singular vectors of $X$.

Principal components analysis can be interpreted in a number of different ways. Clearly, PCA can be considered as a factorization of $X$ that minimizes the least-squares error between $X$ and its best rank $k$ approximation. Alternatively, one can interpret PCA as finding the best $k$-dimensional subspace that approximates the range of $X$ in the least-squares sense. Lastly, if we appropriately scale and normalize the columns of $X$, the principal components given by the columns of $U$ capture the directions of maximum variance in the data.

4.1.2 Sparse Principal Components Analysis

While the SVD provides a computationally efficient strategy for finding a low rank approximation to a collection of data (to solve PCA), in many cases, it is natural to enforce additional structure into the principal components and/or coefficients of the decomposition. One of the most natural types of structure that one may impose on the left or right factors ($U$ and $V$) is that they are sparse or have small $\ell_0$-norm. Finding a sparse decomposition of the data in terms of a set of uncorrelated or orthogonal factors is referred to as sparse PCA or SPCA for short.

Most methods for finding sparse principal components (PCs) proceed by iteratively finding a single PC at a time. To find the first sparse PC, the goal is to find an approximate solution to the following non-convex objective function:

$$\max_v v^T \Sigma v - \gamma \|v\|_0,$$  \hspace{1cm} (4.2)
where $\Sigma = X^T X$. Alternatively, the $\ell_0$ penalty on $v$ can be replaced with the $\ell_1$-norm, which is known to produce sparse solutions [32, 25]. After finding the first sparse PC, a variety of deflation methods can be used to remove the contribution of the first PC from the data; this procedure is repeated to find the next PC until a specified number of PCs are computed. See [40] for a thorough review and comparison of different deflation methods for SPCA.

To place these approaches in the context of our approach for sparse matrix factorization, we can express the SPCA objective as follows:

$$\min_{U \in \mathbb{R}^{M \times L}, V \in \mathbb{R}^{L \times N}} \|X - UV\|_F + \gamma \|V\|_0$$  \hspace{1cm} (4.3)

In our numerical experiments, we compare the performance of the proposed method with a generalized power method [41] developed to solve the $\ell_0$ objective above. A set of sparse principal components (where $U$ admits sparse structure) can be computed by replacing $X$ with its transpose and solving the same SPCA objective function for $X^T$. In our experiments on collections of face images in Fig. 5.4.4, we show the principal vectors learned for $X$ (dense factors $U$ with sparse coefficients $V$) and $X^T$ (sparse factors $U$ with dense coefficients $V$).

A number of different approaches have been developed to solve SPCA (and its variants) [42, 41, 43]. Due to the fact that the SPCA objective is non-convex, most approaches for finding sparse PC’s utilize an alternating minimization strategy where the objective is minimized by updating a single component in $U$ (holding $V$ fixed) and then updating the corresponding component in $V$ (holding $U$ fixed).
4.1.3 Dictionary Learning

The aim of dictionary learning (DL) is to learn a set of basis vectors or atoms that provide sparse representations for a set of training examples [44, 45]. DL has been shown to provide sizable gains in compression, denoising, and signal separation, in comparison to using a fixed dictionary designed according to some mathematical desiderata.

Typically, one poses the problem of dictionary learning as a convex optimization problem, where the total mean-square error of the representation learned basis is traded off with the sparsity of each training signal with respect to the dictionary. To make this precise, let \( x_i \in \mathbb{R}^N \) be a single training example and a collection of \( N \) training examples be written as \( X \in \mathbb{R}^{M \times N} \). We will call the overcomplete dictionary that we wish to learn, \( U \in \mathbb{R}^{M \times L} \), where \( M \leq L \). Using the \( \ell_0 \)-norm to count the number of entries of a vector, the goal is to learn a dictionary \( U \) that admits a representation of each training example that is less than \( K_0 \) sparse. If we write each example as \( x_i = Uv_i \) then we wish to find a dictionary that minimizes the total error in the representation of each signal, subject to the constraint that \( \|v_i\|_0 \leq K_0 \) for all \( i \in \{1, \ldots, N\} \).

An error-constrained version of the DL objective may be expressed as follows.

\[
(DL) \quad \min_{U \in \mathbb{R}^{M \times L}, V \in \mathbb{R}^{L \times N}} \|V\|_0 \quad \text{subject to} \quad \frac{1}{N} \|X - UV\|_2 \leq \delta, \tag{4.4}
\]

where \( \|V\|_0 \) counts the total number of non zeros in its matrix argument and \( \delta \) is a user set parameter that constrains the amount of error in the factorization.

This objective is convex with respect to \( U \) and with respect to \( V \), but it is not jointly convex with respect to both \( U \) and \( V \). To get around this fact, an alternating minimization approach is taken. What this amounts to is fixing \( U \) and solving the
objective to find $V$ and then fixing $V$ and solving the objective to compute $U$. This alternating minimization procedure is repeated until a stopping criterion is met.

There are two common approaches in accomplishing this two-stage procedure. The first approach, known as the Maximum-Likelihood (ML) formulation was originally proposed by Olshausen and Field for the purpose of learning a dictionary that admits sparse representations of natural images. Instead of minimizing the total error in the representation of each training example, they take one step in the direction of the gradient, with respect to $U$. Each iteration of the iterative ML approach for DL consists of two steps.

- **Sparse coding step**: Solve a constrained $\ell_1$-minimization problem for each signal $x_i$ with respect to the current dictionary.

$$\min_{v_i} \|v_i\|_1 \quad \text{s.t.} \quad \|x_i - Uv_i\|_2^2 \leq \epsilon \quad \forall \ i. \quad (4.5)$$

- **Dictionary update step**: At the $m^{th}$ iteration, the dictionary is updated according to the following update rule:

$$U_m = U_{m-1} - \eta \nabla U_{m-1}, \quad \text{where} \quad \nabla U_{m-1} = V^T(X - U_{m-1}V). \quad (4.6)$$

The second approach for dictionary learning is a greedy variant of the DL algorithm above known as KSVD [45], where the sparse coding step is solved via a greedy pursuit (typically OMP). The complexity of updating the $m^{th}$ atom of the dictionary is dominated by computing the first left singular vector of a $M \times R$ error residual matrix $E_S$, where $E = X - UV$ and $E_S$ is a sub-dictionary containing the columns corresponding to the $R$ examples that use the $m^{th}$ atom.
4.2 Column Selection for Matrix Factorization

In this Section, we provide background of column selection-based methods for matrix factorization. The idea behind column selection-based methods is to simply select a subset of columns from a matrix and then compute a factorization based upon these sampled columns.

4.2.1 Column Subset Selection

Identifying a set of $k$ columns that provide the best approximation to a collection of $N$ signals of $M$ dimensions $\mathbf{X} \in \mathbb{R}^{M \times N}$ is referred to as the column subset selection problem (CSS). Here, we consider the case where $\mathbf{X}$ is overcomplete and $M \ll N$. The CSS problem can be formulated as follows.

\[(\text{CSS}) \quad \min_{|S|=L} \| \mathbf{X} - \pi_S(\mathbf{X}) \|_F, \quad (4.7)\]

where $\pi_S(\mathbf{X})$ denotes the orthogonal projection of $\mathbf{X}$ onto the range of $\mathbf{X}_S$. In words, the CSS objective aims to find the best set of $L$ columns from $\mathbf{X}$ (indexed by the set $S$) that provide the best approximation to $\mathbf{X}$ in the least-squares sense. When $\mathbf{X}$ is a rank $k$ matrix, then all sub matrices $\mathbf{X}_S \in \mathbb{R}^{M \times L}$ of $\mathbf{X}$ that consist of linearly independent columns will produce the same approximation error.

A number of methods rely on column subset selection (CSS) to form approximate (and in some cases more interpretable) low rank factorizations and SVDs [46, 19, 20, 47, 18]. All of these methods, can be thought of as a randomized algorithms to solve the non-convex CSS optimization problem stated above. In the case of CUR decompositions [18], both the rows and columns of a data matrix are sampled to find a low rank factorization of the data.
4.2.2 Nyström Method

The Nyström method [48, 49] provides a computationally efficient method for finding a low rank approximation to a PSD matrix using only a small subset of its columns. The key assumption underlying the Nyström method is that the entire dataset can be represented in terms of a small subset of the data. The Nyström method has been used successfully to approximate versions of the Gram and kernel matrices used in machine learning applications from kernel-based classification (kernel SVM) to nonlinear dimensionality reduction (manifold learning).

Consider an $N \times N$ PSD matrix $G$ of rank $r$. Suppose we sample a subset $\Lambda$ of the columns of $G$ of size $|\Lambda| = k \geq r$. Let $C_k = G_{\Lambda}$ denote the $N \times k$ matrix of sampled columns. Let $W_k$ denote the $k \times k$ matrix of elements who’s row and column indices lie in $\Lambda$. The Nyström approximation of $G$ is defined as

$$G \approx \tilde{G}_k = C_k W_k^T C_k^T. \tag{4.8}$$

An approximate singular value decomposition (SVD) of $G$ can then be obtained from the SVD of the smaller matrix $W_k$. Since $W$ is $k \times k$, this computation is much faster than the full $N \times N$ SVD of $G$, where the complexity of the SVD step reduces from $O(N^3)$ to $O(k^3)$ with $k \ll N$.

4.2.3 Adaptive Sampling Methods

The performance of CSS-based approaches and the Nyström method depend heavily on how the columns of $G$ are selected. Non-adaptive methods simply select columns of $G$ at random, either with or without replacement [46]. However, much better performance is obtained in practice using adaptive methods that choose a highly efficient representation of the data.
One popular sampling strategy computes a eigenvalue decomposition of $G$ to produce *Leverage scores* [19], which measures the influence that a column of $G$ has on the best rank-$r$ approximation to $G$. To be precise, let $G = U \Sigma U^T$ denote the eigenvalue decomposition of $G$ and $U_1 \in \mathbb{R}^{N \times r}$ denote the leading $r$ eigenvectors of $G$. The leverage score of the $j$th column of $G$ equals

$$\ell_j = \|(U_1)_j\|_2,$$

where $(U_1)_j$ denotes the $j$th row of $U_1$. After computing the leverage scores, columns of $G$ are sampled with probability proportional to their leverage score.

Another popular strategy for column selection, which we refer to as *Sequential Error-Based Selection (SES)*, selects columns of $G$ that are poorly approximated by the current Nyström approximation [20, 47, 50]. To be precise, let $S$ denote the current set of columns selected at the $m$th iteration and let $E = G - \pi_S(G)$ be the residual error obtained by projecting $G$ onto the subspace spanned by the current set of columns $G_S$. SES selects one (or more) column with probability proportional to the norm of each column of the current residual $E$.

An excellent review of sampling methods for the Nyström method is provided in [51, 52] and comparison of spectral and Frobenius error bounds for different column selection methods is provided in [51].
5.1 Self-Expressive Sparse Recovery

In standard applications of sparse recovery, one considers the representation of a signal in terms of a pre-specified basis or dictionary consisting of atoms that have been learned or designed according to some mathematical desiderata. More recently however, sparse approximation has been used to compute a representation of a collection of signals in terms of other signals from the same collection. We refer to representations formed “from within” the data as self-expressive representations.

Recently, the idea of self-expression has been coupled with sparse recovery in the context of multi-class classification [53, 54] and also to reveal subspace structures from large collections of data in an unsupervised setting [55, 56, 15]. The key insight underlying these approaches is that when self-expressive sparse recovery is performed over collections of data living on unions of subspaces, signals that lie in (or near) the same low-dimensional subspace will utilize one another in their sparse representations.

In the aforementioned applications of self-expressive sparse recovery, a sparse representation of each signal is computed with respect to all of the signals in the dataset. Here, we consider a more general formulation of this problem where one computes a
sparse representation of a signal in the dataset with respect to a subset of the signals in the dataset. Formally, we consider the sparse representation of a collection of \( N \) signals \( X \in \mathbb{R}^{M \times N} \) each of dimension \( M \), in terms of a reference set containing other signals from the same collection. The sparse representation of the \( i \)th point \( x_i \in \mathbb{R}^M \) in terms of the signals indexed by the reference set \( S \subset \{1, \ldots, N\} \) of size \( L \) is given by

\[
\mathbf{v}^* = \arg \min_{\mathbf{v} \in \mathbb{R}^L} \| \mathbf{v} \|_0 \quad \text{subject to} \quad x_i = \sum_{j \in S, j \neq i} \mathbf{v}(j)x_j, \quad (5.1)
\]

where \( \| \mathbf{v} \|_0 \) counts the number of non-zeroes in its argument and \( \mathbf{v}(j) \in \mathbb{R} \) denotes the contribution of the \( j \)th signal \( x_j \) to the representation of \( x_i \). Let \( \Lambda^{(i)} = \text{supp}(\mathbf{v}^*) \) denote the subset of points selected to represent the \( i \)th point and \( \mathbf{v}(j) \) denote the contribution of the \( j \)th signal to the self-expressive representation of \( x_i \). By penalizing representations that require a large number of non-zero coefficients, the resulting representation will be sparse.

In general, finding the sparsest representation of a signal has combinatorial complexity; thus, sparse recovery methods such as basis pursuit (BP) [25] or OMP [22] are employed to obtain approximate solutions to (5.1). In the sequel, we will utilize greedy methods to solve (5.1) and analyze the performance of OMP for this task. In addition to the fact that OMP is a lower-complexity alternative than convex optimization-based approaches such as BP and LASSO, another advantage of greedy methods is that they provide a natural way to either supply an error criterion to compute an approximation solution to Error or supply the maximum sparsity level to compute an approximation solution to Sparse.
5.2 Sparse Self-expressive Decomposition (SEED)

This Section introduces SElf-Expressive Decomposition (SEED), a novel approach for sparse matrix decomposition.

5.2.1 Motivation

While PCA is a powerful and widely used approach for dimensionality reduction, when applied to nonlinear data, PCA leads to poor performance when used as a pre-processing step for clustering/identification tasks [18]. To overcome these issues, a number of data-aware approaches for dimensionality reduction such as sparse PCA [57, 42, 41], generalized PCA [58], independent components analysis (ICA) [59], and dictionary learning (DL) [60, 45], have been proposed to accommodate various types of nonlinear structure in data.

In practice, enforcing additional structure into a factorization of the data often leads to better clustering and classification performance, however, methods to enforce additional structure are inherently non-convex and are not guaranteed to find bases that are aligned with the geometric structures in the data. For this reason, the learned components (basis vectors) used to represent the data consist of mixtures of true samples from the data and are therefore difficult to interpret.

Rather than learning a basis to encapsulate structure in data, an alternative strategy is to simply using the data to represent itself. We refer to techniques that decompose data in terms of points from the same dataset as self-expressive decompositions. This idea of ‘self-expression’ has been utilized successfully in the context of dimensionality reduction and low rank matrix factorization, where a small subset of data points are selected and then used as a basis for representing the remaining data [18]. In a separate but related line of work, self-expression is used to compute a sparse decomposition of a dataset $\mathbf{X} = \mathbf{XV}$, in terms of all of the remaining signals in
the dataset. This approach leads to provable guarantees for clustering data into its underlying nonlinear structures [61, 14, 15].

5.2.2 SEED Method

The SEED algorithm computes a sparse factorization \( \hat{X} = DV \) in two steps. First, we adaptively subsample columns from \( X \) to form a small reference set which can be used as the building blocks for the remaining columns. Second, we form a sparse representation of the remaining dataset with respect to this subset of columns. These two steps are summarized below.

**Step 1: Column Subset Selection with oASIS**

The first step of SEED is to select a subset of columns from the dataset \( X \) to represent the range of the data as efficiently as possible. To compute an efficient decomposition of the dataset, we introduce a method called Accelerated Sequential Incoherent Selection (oASIS) which selects columns from \( X \) based upon the improvement of the Nyström approximation to the Gram matrix of the data \( G = X^T X \). A detailed derivation of the selection criterion underlying oASIS is provided below in Sec. 5.2.3. This method returns an index set \( S \), corresponding to the \( L \) selected columns. From this sample set, we form the submatrix \( X_S \) and then normalize all of the columns to obtain a reference set, \( D \in \mathbb{R}^{M \times L} \). We detail the oASIS algorithm for column selection in Alg. 3 and provide a complexity analysis of oASIS in Sec.7.4.

In Sec. 7.3.2, we show that the reference set returned by oASIS is guaranteed to consist of linearly independent columns. Based upon this property of the oASIS algorithm for column selection with the Gram matrix, we show in Thm. 8 that this is sufficient to guarantee that after \( r \) steps of the algorithm, we can obtain exact matrix recovery for rank \( r \) matrices. Thus, oASIS is capable of achieving perfect recovery
of low rank matrices with the same efficiency as PCA. Remarkably, in a number of real world examples we find that oASIS achieves exact matrix recovery with the same number of samples as the number of factors required by PCA. Thus, in theory and in practice, we find that oASIS provides a powerful strategy for approximating the range of a matrix and also leads to better clustering results than state-of-the-art methods such as SSC.

Step 2: Greedy Sparse Recovery with OMP

The second step of SEED is to compute the sparse representation of the columns of $X$ in terms of the reference set $D$ obtained in Step 1. To do this, we employ a greedy sparse recovery procedure, either OMP or Batch OMP. Our final factorization of the data is given by $\tilde{X} = DV$, where $V$ contains sparse representations of the columns of $X$.

5.2.3 Derivation of oASIS

As discussed in Sec. 4.2.2, the Nyström method approximates a low-rank matrix using only a subset of its columns. In practice, the quality of the approximation is highly dependent on which columns are chosen. The oASIS method optimizes the Nyström approximation by sequentially choosing columns using a greedy column subset selection. The method selects columns one at a time, on each iteration selecting a column that has a large impact on the resulting approximation.

Unfortunately, it is impossible to predict how much each column will impact $\tilde{G}$ without examining all of the candidate columns. However, we show below that it is possible to calculate the impact of a column on the upper-left block of $\tilde{G}$ even before that column has been selected. This quantity is a lower bound on how much the new column will affect the Nyström approximation as a whole.
Let \( \tilde{G}_k \) denote the \( k\)-column Nyström approximation of \( G \). Recall that the matrix \( G \) can be written as \( X^T X \), and that \( X_k \) denotes the first \( k \) columns of \( X \). The upper-left \((k + 1) \times (k + 1)\) block of \( G \) can be written in terms of the matrix \( X_{k+1} \). Let \( b_{k+1} \) contain the first \( k \) entries in column \( G(:, k+1) \), and \( d_{k+1} \) denotes the diagonal entry \( G(k + 1, k + 1) \). the upper left \((k + 1) \times (k + 1)\) block of \( \tilde{G}_k \) can be written in terms of \( X_{k+1}^T X_{k+1} \) as follows:

\[
\begin{bmatrix}
X_{k+1}^T X_{k+1} & b_{k+1} \\
\end{bmatrix}
\begin{bmatrix}
W_k^T \\
0 \\
0 \\
\end{bmatrix}
\begin{bmatrix}
X_{k+1}^T X_{k+1} & b_{k+1} \\
\end{bmatrix} =
\begin{bmatrix}
X_{k+1}^T X_{k+1} & b_{k+1} \\

b_{k+1}^T & d_{k+1} \\
\end{bmatrix}
\begin{bmatrix}
b_{k+1}^T & W_k^T b_{k+1} \\
\end{bmatrix}.
\tag{5.2}
\]

If the Nyström approximation \( \tilde{G}_k \) is an accurate approximation to \( G \), then the upper left block (5.2) of \( \tilde{G}_k \) should match the upper-left block of \( G \). The discrepancy between these two blocks is given by \( |d_{k+1} - b_{k+1}^T W_k^T b_{k+1}| \). In the event that \( d_{k+1} - b_{k+1}^T W_k^T b_{k+1} = 0 \), the matrix \( W_k^T \) behaves similarly to the matrix \( W_k \), and the new column does not impact the upper-left block of \( \tilde{G}_k \). However, if \( |d_{k+1} - b_{k+1}^T W_k^T b_{k+1}| \) is large, then this column is guaranteed to have a substantial impact on the Nyström approximation.

This observation is used on each iteration of oASIS to choose an influential column. On each iteration of the method, we calculate the influence of each unsampled column using the formula

\[
\Delta_i = d_i - b_i^T W_k^T b_i.
\]

We then select the un-sampled column that maximizes \( |\Delta_i| \) and add it to the Nyström approximation. The algorithm terminates when \( |\Delta_i| \) falls below a threshold set by the user.
Accelerated Sampling

A naive implementation of oASIS is inefficient because each step requires a matrix inversion to form $W_{k+1}^\dagger$ in addition to calculating the errors $\Delta_i$. Fortunately, this can be done efficiently by updating the results from the previous step using block matrix inversion formulas and rank-1 updates. The resulting algorithm is dubbed Accelerated Sequential Incoherent Selection (oASIS) as it performs the incoherent selection strategy described above with a highly efficient sequence of rank-1 updates. We give a brief derivation of the algorithm here.

We first consider the calculation of $W_{k+1}^\dagger$ after column $k+1$ is added to the approximation. Using the notation of Section 5.2.3, we let $b_k$ denote the first $k$ rows of column $G(:, k+1)$ and $d_k$ denote its diagonal. Using a block inversion formula, we get

$$W_{k+1}^{-1} = \begin{bmatrix} W_k & b_{k+1}b_{k+1}^T & d_{k+1} \end{bmatrix}^{-1} = \begin{bmatrix} W_k^{-1} + s_{k+1}q_{k+1}q_{k+1}^T, & -s_{k+1}q_{k+1} \\ -s_{k+1}q_{k+1}^T, & s_{k+1} \end{bmatrix}$$

(5.3)

where $s_{k+1} = (d_{k+1} - b_{k+1}^TW_{k+1}^{-1}b_{k+1})^{-1} = \Delta_{k+1}^{-1}$ is the (scalar valued) Schur complement and $q_{k+1} = W_k^{-1}b_{k+1}$ is a column vector. This update formula allows $W_{k+1}^{-1}$ to be formed by updating $W_k^{-1}$, and only requires inexpensive vector-vector multiplication. Note that $W_{k+1}$ is invertible as long as $\Delta_{k+1}$ (the Schur complement) is non-zero, which is guaranteed by our sampling rule: the algorithm terminates if $\Delta_{k+1} = 0$ in which case our approximation is exact.

We now consider the calculation of $\Delta_i = d_i - b_i^TW_i^\dagger b_i$ for all $i$. Note that on step $k$ of the method, we have $C_k^T = [b_1, b_2, \cdots, b_N]$. We can evaluate all values of $b_i^TW_k^\dagger b_i$ simultaneously by computing the entry-wise product of $C_k$ with the matrix $R_k = W_k^{-1}C_k^T$ and then summing the resulting columns. If we have already formed $C_k$ and $R_k$ on iteration $k$, the matrix $R_{k+1} = W_{k+1}^{-1}C_{k+1}^T$ needed on the next iteration
is obtained by applying (5.3) to \( C_{k+1}^T \) to get

\[
R_{k+1} = W_{k+1}^{-1} C_{k+1}^T = W_{k+1}^{-1} \begin{bmatrix} C_k^T \\ c_{k+1}^T \end{bmatrix} = \begin{bmatrix} R_k + s_{k+1} q_{k+1} (q_k^T c_k^T - c_{k+1}^T) \\ s_{k+1} (-q_k^T c_k^T + c_{k+1}^T) \end{bmatrix}. 
\] (5.4)

Equation (5.4) forms \( R_{k+1} \) by updating the matrix \( R_k \) from the previous iteration. The update requires only matrix-vector and vector-vector products. The application of this fast update rule to perform incoherent sampling yields Algorithm 3.

5.2.4 Variant of Alg. 2

In Alg. 2, the sparse representation of the entire dataset \( X \) is computed in terms of the reference set \( D \) to form the factorization, \( \tilde{X} = DV \). If we reorder \( X = [X_S \ X_{-S}] \), such that all of the selected columns are placed into a block \( X_S \) and the remaining signals \( X_{-S} \) are stacked together, then equivalently, we can write the sparse matrix \( V = [V_S \ V_{-S}] \), where \( V_{-S} \) contains the sparse representations of the unsampled signals in its columns. In this case, \( V_S \) is simply a diagonal matrix containing the norm of the columns \( X_S \) in its diagonal entries.

We now introduce a variant of this algorithm which we will utilize in later applications of SEED. This variant modifies the way in which we compute the sparse representation of the sampled signals \( X_S \). Rather than simply setting \( V_S \) to be a diagonal matrix containing the norm of the sampled signals \( X_S \), we solve the following SSC objective:

\[
V_S = \arg \min_{W \in \mathbb{R}^{L \times L}} \|W\|_0 \text{ subject to } \|X_S - DW\|_F \leq \epsilon, \ \text{diag}(W) = 0. \quad (5.5)
\]
Algorithm 2: Sparse SELF-Expressive Decomposition (SEED)

**Input:** A dataset $X \in \mathbb{R}^{M \times N}$, the maximum number of columns to select $L$, termination criterion for oASIS $\delta$, termination criteria for OMP (target sparsity $k$ and/or approximation error $\epsilon$).

**Output:** A reference set $D \in \mathbb{R}^{M \times L}$ and sparse coefficient matrix $V \in \mathbb{R}^{L \times N}$.

**Step I. Column Subset Selection:** Select $L$ columns via oASIS (Alg. 3) and normalize the selected columns to form $D \in \mathbb{R}^{M \times L}$.

**Step II. Greedy Sparse Recovery:** Solve OMP for each column of $X$ with respect to $D$ and stack the result into the corresponding column of $V \in \mathbb{R}^{L \times N}$.

Algorithm 3 Accelerated Sequential Incoherent Selection (oASIS)

**Inputs:** The Gram matrix $G = X^T X$.
- The diagonal elements of $G$, stored in $d$.
- The maximum number of columns to sample, $L$.
- The number of columns to initialize the algorithm $k < L$.
- A non-negative stopping tolerance $\delta$.

**Initialize:** Choose a vector $\Omega \in [1, N]^K$ of $k$ random starting indices.

- $C_k = G(:, \Omega)$
- $W_k^{-1} = G(\Omega, \Omega)^{-1}$
- $R_k = W_k^{-1} C_k^T$

**while** $k < L$ **do**

- $\Delta = d - \text{colsum}(C_k \circ R_k)$
- $i = \arg \max_{j \notin \Omega} |\Delta(j)|$
- **if** $|\Delta(i)| < \delta$ **then**

  - **return**

  - **end if**

- $b_k = G(\Omega, i)$
- $d_{k+1} = d(i)$
- $s_{k+1} = 1/\Delta(i)$
- $q_{k+1} = R(:, i)$
- $C_{k+1} = [C_k, G(:, i)]$

- Form $W_{k+1}$ using Eq. (5.3)
- Update $R_{k+1}$ using Eq. (5.4)
- $k \leftarrow k + 1$
- $\Omega \leftarrow \Omega \cup \{i\}$

**end while**
5.3 Outlier Detection with SEED

Now, we describe how the decomposition provided by SEED can be used to segment low rank structures from outliers.

5.3.1 Motivation

An interesting property of self-expressive decompositions is that the degree of sparsity provides information about the geometry of the reference set and how it is matched to the remaining signals in the collection. For instance, when a collection of signals lie on a union of subspaces and we sample enough signals from each subspace to span each low-dimensional subspace, then the resulting sparse decomposition provided by SEED reveals the rank of each subspace. In particular, the sparsity level of each signal in a $k$-dimensional subspace is bounded by $k$. Thus, we can use this rank revealing property of SEED to determine whether a signal lies on one of the low-dimensional subspaces in the ensemble (in which case, we say its an inlier) or whether the signal is an outlier.

As a motivating example, we point the reader to Fig. 5.3.1, where we show an example of the sparse representations formed when our reference set contains a complete sampling of points from two different subspaces (left) and only one of two subspaces (right). On the right, we observe that all the signals in the subspace that isn’t yet covered by the reference set admit dense representations. Of course this idea only really works for sparse representations formed via an error-constrained version of OMP (Error), because then we allow the sparsity to shrink as small as possible. By allowing the sparsity level to vary, rather than fixing the sparsity as in (Sparse), the resulting sparse coefficients can in some cases provide a great deal more information than their sparsity-constrained counterpart.
Figure 5.1: *Demonstration of rank revealing property of SEED for outlier detection.* The sparsity level (top row) and sparse coefficient matrices (bottom row) for SEED with (a) oASIS sampling ($L = 160$), (b) random sampling ($L = 160$), and (c) SPCA ($L = 60$).

5.3.2 Sparsity-driven Outlier Detection

An interesting property of our sparsity-driven decomposition method is that the sparsity level of the representations obtained by SEED can be used to develop a provable method for outlier detection. In order to perform outlier detection, the sparse coefficients $\mathbf{V}_S$ are obtained by solving the SSC objective in (5.5). To compute both $\mathbf{V}_S$ and $\mathbf{V}_{-S}$, we utilize batch OMP to solve (Error) by providing an error tolerance $\epsilon$ to the algorithm.

Once we compute a sparse factorization $\hat{\mathbf{X}} = \mathbf{D}\mathbf{V}$, then our goal is to use the number of nonzeros in each column of $\mathbf{V}$ (sparsity level) to determine whether a column of $\mathbf{V}$ is an outlier or an inlier: when a column of $\mathbf{V}$ is dense we declare it an outlier and when $\mathbf{V}$ is sufficiently sparse, we declare it an inlier. We detail this outlier detection method in Alg. 4 and provide a sufficient condition that guarantees this method will reliably detect all outliers in Thm. 7.

In general, determining an appropriate threshold $\tau$ to segment low rank structures
Algorithm 4: Outlier Detection with SEED

Input: Sparse matrix $V \in \mathbb{R}^{L \times N}$ and a threshold $\tau$.

Output: An index set $\Lambda$ containing the indices of all outliers.

For all $j = 1 : N$, compute the sparsity $\|v_j\|_0$ of each column of $V$. If $\|v_j\|_0 > \tau$, declare $x_j$ an outlier and add $j$ to the index set $\Lambda \leftarrow j \cup \Lambda$.

From outliers can be challenging. However, in practice, the distribution of signal sparsity tends to admit a bi- or multi-modal distribution. See the top of Fig. 5.3.1 for an example of the sparsity of signals living in a union of 20-dimensional subspaces corrupted with incoherent outlier points. In this example, we set $\epsilon = 0.3$, so for all of the signals that lie on 20-dimensional subspaces, we observe that when we sample $L = 160$ columns, the sparsity level is around $k = 2$; in contrast, all of the incoherent outliers admit a sparsity level of around $k = 10$. Alternatively, instead of setting the threshold $\tau$ explicitly, the k-means algorithm can be employed to automatically segment the data based upon their sparsity level.

5.4 Numerical Experiments

5.4.1 Evaluation Setup

To evaluate the performance of SEED, we compute a $L$-dimensional sparse factorization of the data $\tilde{X} = DV$, where we form $D \in \mathbb{R}^{M \times L}$ by utilizing: (i) oASIS sampling, (ii) sequential error selection (SES) [46], (iii) leverage score sampling (Lev) [18], and (iv) uniform random sampling (Random). Both leverage score sampling and SES are detailed in Sec. 4.2.3.

In addition, we also compare the performance of SEED with a generalized power method for SPCA [41] for solving the $\ell_0$ problem in Eq. 4.2. In essence, this method applies a PGD-based power method to compute the eigenvectors of the square covariance matrix $\Sigma = X^T X$ (or $XX^T$) and then uses a hard-thresholding function to
enforce the $\ell_0$ penalty in the objective. To enforce sparsity in the right factor matrix, we solve Eq. 4.2 for $\Sigma = X^T X \in \mathbb{R}^{N \times N}$, to obtain a set of $L$ sparse principal components $Z \in \mathbb{R}^{N \times L}$ and then compute the corresponding $M \times L$ (typically dense) left factor matrix as $D = YZ$. In all of our experiments, the regularization parameter applied to the $\ell_0$ penalty in Eq. 4.2 is set to $\gamma = 0.1 \gamma_{\text{max}}$, where $\gamma_{\text{max}}$ is computed by the theoretical upper bound. This choice of regularization parameter is selected to tradeoff between the $\ell_0$ and $\ell_2$ penalty with weights equal to 0.1 and 0.9 respectively, i.e., when $\gamma = 0.5 \gamma_{\text{max}}$ this places equal weight on the $\ell_0$ and $\ell_2$ terms. We select this value of regularization parameter because in preliminary experiments, we found that this value of $\gamma$ provides a good balance between achieving good factorization error and good clustering performance (reported in Sec. 6.5.2).

In the subsequent experiments, we compute the relative approximation error as we vary the size of the factorization. Let $D$ be the reference set or dictionary/basis obtained via SEED and PCA/SPCA respectively. The relative approximation error is defined as:

$$\text{err}(X, D) = \frac{\|X - DD^+X\|_F^2}{\|X\|_F^2}.$$ 

5.4.2 Datasets

We evaluate the performance of SEED on five different datasets—four real world datasets and one synthetic example. Details are provided for each of these datasets below.

(YF) Yale Face Image Database: Face recognition remains a challenging problem due to the fact that images of a person’s face exhibit a high degree of variability depending upon the pose and illumination that the image is captured under. While images of faces captured under different illumination conditions exhibit photometric variability (the pixel values are very different), there is a lot of
low-dimensional structure present in multiple images of the same face. In fact, images of Lambertian objects (no specular reflections) captured under different illumination conditions have been shown to be well-approximated by a ten dimensional subspace [23]. Thus, we can pose the problem of clustering images of many subject’s faces under variable illumination as a subspace clustering problem, where the goal is to sort through a collection of images consisting of multiple people’s faces and cluster the images according to their corresponding the identity of the subject.

(LF) **Light Field Data from Multi-Camera Arrays:** Multi-camera arrays generate a large number of images of the same object/scene from varying angles and positions. Acquiring images from different angles and positions provides 3D information about the scene of interest and can be used to estimate the depth of objects in the scene, refocus the image at different depths, and for computing 3D models of objects. To solve a number of problems associated with refocusing and image analysis from multi-camera arrays, the images must be tiled into small patches and then the same patches from different views are stacked into a long vector called a *light field atom*. Thus, being able to quickly decompose and compress large collections of light field patches is essential for solving a number of super-resolution and refocusing problems.

To evaluate the performance of SEED for compressing light field data, we utilize two datasets sampled from the Stanford Computer Graphics Laboratory Archive [62]. In both settings, we generate a large collection of light field atoms from the set of Chess images (chess board) in the database. Our first dataset, LF-i, consists of $N = 3000$ small $4 \times 4$ patches from a $4 \times 4$ subset of the multi-camera array, resulting in a dataset of size $M = 256$ and $N = 3000$. Our second dataset, LF-ii, consists of $8 \times 8$ patches from a $17 \times 17$ camera array, resulting in a dataset
Figure 5.2: Hyperspectral signatures from three different material classes.

of size $M = 18k$ and $N = 100k$.

(HS) **Hyperspectral Imagery:** In contrast to standard imaging devices that acquire three channels of spectral information (RGB) at each spatial position of the image, hyper spectral imaging (HSI) systems acquire information from hundreds of spectral bands. By collecting hundreds of channels of spectral information, the resulting “hyperseptical signatures” can be used to detect small differences in the chemical composition of crops or vegetation which is essential for tracking and monitoring agriculture production, yield, crop health, and potentially detect early signs of infestation or other trends and interactions between different insect populations in crops. Hyperspectral images can be modeled as approxi-
mately piecewise constant regions with similar spectral signatures. See Fig. 5.2 for an example of hyperspectral signatures from the AVARIS sensor [63] that is overlaid on the ground truth map of the different types of vegetation in the underlying scene (e.g., corn, grass, alfalfa).

To evaluate the performance of SEED for clustering HSI remote sensing data, we generate two datasets by subsampling HS images from an agricultural terrain (Salinas) collected via the AVARIS sensor. Both datasets, HS-i and HS-ii, consist of hyperspectral data from five different materials: HS-i contains $N = 4500$ samples with 900 samples from each of the five classes (uniformly distributed data) and HS-ii contains $N = 11k$ samples, where we sample $N_i = 2^i \times 100$ pixels from each of the five classes with $i = \{1, \ldots, 5\}$ to produce non-uniformly distributed data.

(MS) **Motion Segmentation from Point Correspondences:** Motion segmentation is an important yet challenging problem in computer vision where the goal is to segment different motion trajectories (from different objects) from video sequences. Each trajectory corresponds to the motion of a different object or even the motion introduced from the camera, all of which lie on a low-dimensional subspace. When multiple objects are moving within the field of view, the problem of motion segmentation boils down to learning subspaces from point correspondences and then segmenting the data in accordance with these learned subspaces.

(US) **Synthetic Data from Unions of Overlapping Subspaces:** To study the performance of SEED for subspace clustering and outlier detection, we generate collections of synthetic data living on unions of low-dimensional subspaces in accordance with the model provided in Sec. 5.4.3. We generate data from a union of two subspaces of dimension $k = 20$ with a 3-dimensional overlap and
then add a collection of outliers created by generating random Gaussian vectors, resulting in a dataset of size \( N = 450 \) with \( N_1 = 300 \) points in the first subspace, \( N_2 = 100 \) points in the second subspace, and \( N_o = 50 \) outlier points.

### 5.4.3 Generative Model for Synthetic Data

In order to study EFS for unions of subspaces with varied cross-spectra, we will generate synthetic data from unions of overlapping \textit{block sparse signals}.

#### Constructing Sub-dictionaries

We construct a pair of sub-dictionaries as follows: Take two subsets \( \Omega_1 \) and \( \Omega_2 \) of \( k \) signals (atoms) from a dictionary \( A \) containing \( D \) atoms \( \{a_m\}_{m=1}^D \) in its columns, where \( a_m \in \mathbb{R}^M \) and \( |\Omega_1| = |\Omega_2| = k \). Let \( \Psi \in \mathbb{R}^{M \times k} \) denote the subset of atoms indexed by \( \Omega_1 \), and let \( \Phi \in \mathbb{R}^{M \times k} \) denote the subset of atoms indexed by \( \Omega_2 \). Our goal is to select \( \Psi \) and \( \Phi \) such that \( G = \Psi^T \Phi \) is diagonal, i.e., \( \langle \psi_i, \phi_j \rangle = 0 \), if \( i \neq j \), where \( \psi_i \) is the \( i \)th element in \( \Psi \) and \( \phi_j \) is the \( j \)th element of \( \Phi \). In this case, the cross-spectra is defined as \( \sigma = \text{diag}(G) \), where \( \sigma \in [0,1]^k \). For each union, we fix the “overlap” \( q \) or the rank of \( G = \Psi^T \Phi \) to a constant between zero (orthogonal subspaces) and \( k \) (maximal overlap).

To generate a pair of \( k \)-dimensional subspaces with a \( q \)-dimensional overlap, we can pair the elements from \( \Psi \) and \( \Phi \) such that the \( i \)th entry of the cross-spectra equals

\[
\sigma(i) = \begin{cases} 
|\langle \psi_i, \phi_i \rangle| & \text{if } 1 \leq i \leq q, \\
0 & \text{if } i = q + 1 \leq i \leq k.
\end{cases}
\]

We can leverage the banded structure of shift-invariant dictionaries, e.g., dictionary matrices with localized Toeplitz structure, to generate subspaces with structured
cross-spectra as follows. First, we fix a set of \( k \) incoherent (orthogonal) atoms from our shift-invariant dictionary, which we place in the columns of \( \Psi \). Now, holding \( \Psi \) fixed, we set the \( i^{th} \) atom \( \phi_i \) of the second sub-dictionary \( \Phi \) to be a shifted version of the \( i^{th} \) atom \( \psi_i \) of the dictionary \( \Psi \). To be precise, if we set \( \psi_i = d_m \), where \( d_m \) is the \( m^{th} \) atom in our shift-invariant dictionary, then we will set \( \phi_i = d_{m+\Delta} \) for a particular shift \( \Delta \). By varying the shift \( \Delta \), we can control the coherence between \( \psi_i \) and \( \phi_i \). In Fig. 5.3, we show an example of one such construction for \( k = q = 5 \). Since \( \sigma \in (0, 1]^k \), the worst-case pair of subspaces with overlap equal to \( q \) is obtained when we pair \( q \) identical atoms with \( k - q \) orthogonal atoms. In this case, the cross-spectra attains its maximum over its entire support and equals zero otherwise. For such unions, the overlap \( q \) equals the dimension of the intersection between the subspaces. We will refer to this class of block-sparse signals as **orthoblock sparse signals**.

### Coefficient Synthesis

To synthesize a point that lives in the span of the sub-dictionary \( \Psi \in \mathbb{R}^{n \times k} \), we combine the elements \( \{\psi_1, \ldots, \psi_k\} \) and subspace coefficients \( \{\alpha(1), \ldots, \alpha(k)\} \) linearly to form

\[
x_i = \sum_{j=1}^{k} \psi_j \alpha(j),
\]

While shift-invariant dictionaries appear in a wide range of applications of sparse recovery [64, 65], we introduce the idea of using shift-invariant dictionaries to create structured unions of subspaces in [15].
where $\alpha(j)$ is the subspace coefficient associated with the $j^{th}$ column in $\Psi$. Without loss of generality, we will assume that the elements in $\Psi$ are sorted such that the values of the cross-spectra are monotonically decreasing. Let $x_i^c = \sum_{j=1}^{q} \psi_j \alpha_i(j)$ be the “common component” of $x_i$ that lies in the space spanned by the principal directions between the pair of subspaces that correspond to non-orthogonal principal angles between $(\Phi, \Psi)$ and let $x_i^d = \sum_{j=q+1}^{k} \psi_j \alpha_j(j)$ denote the “disjoint component” of $x_i$ that lies in the space orthogonal to the space spanned by the first $q$ principal directions.

For our experiments, we consider points drawn from one of the two following coefficient distributions, which we will refer to as (M1) and (M2) respectively.

- **(M1) Uniformly Distributed on the Sphere**: Generate subspace coefficients according to a standard normal distribution and map the point to the unit sphere

  $$x_i = \frac{\sum_j \psi_j \alpha(j)}{\left\| \sum_j \psi_j \alpha(j) \right\|_2}, \quad \text{where } \alpha(j) \sim \mathcal{N}(0, 1).$$

- **(M2) Bounded Energy Model**: Generate subspace coefficients according to (M1) and rescale each coefficient in order to bound the energy in the common component

  $$x_i = \frac{\tau x_i^c}{\|x_i^c\|_2} + \frac{(1 - \tau) x_i^d}{\|x_i^d\|_2}.$$

By simply restricting the total energy that each point has in its common component, the bounded energy model (M2) can be used to produce ensembles with small bounding constant to test the predictions in Thm. 5.
5.4.4 Results

Results for YF Dataset

To evaluate SEED for clustering face images, we utilize two different datasets sampled from the extended Yale B Image Database [66]. We subsample each image so that they are $48 \times 84$ pixels and then reorder the pixels into a vector of dimension $M = 4038$. Our first dataset YF-i, consists of 6 subject’s faces and our second dataset, YF-ii, consists of 20 subject’s faces under 60 different illumination conditions, resulting in a dataset of size $N = 360$ and $N = 1200$ respectively. In Fig. 6.1, we show an example of two subject’s faces from this dataset and the affinity matrices generated via NN methods (left) and OMP (right).

In Fig. 5.4.4, we show the left factors (principal components) learned via SPCA and also the samples selected via oASIS, SES, leverage score-based, and random sampling. This figure highlights the advantages of oASIS in finding incoherent samples from the span of the data; in this case, we observe that oASIS selects many diverse illumination conditions from which to build a factorization. In contrast, uniform random sampling selects a number of redundant front facing illumination conditions and does not sample a sufficient number of extreme illumination conditions. SES does a better job of selecting both front facing and extreme illuminations, however, SES still picks up some redundant samples. For this reason, the approximation error obtained via uniform sampling is significantly worse than SES and oASIS for small values of $L$. In contrast to sampling-based factorizations, the factors obtained via SPCA (for both sparse right factors (e) and sparse left factors (d)), the principal components obtained via SPCA mix different faces together (components do not contain information from a single face) and thus result in inaccurate clusterings.

In Fig. 5.5, we display the relative factorization error for YF-i (a) and YF-ii (b) as a function the relative factorization size $L/N$. In both of these examples, after
the relative number of samples \( L/N \) exceeds 30%, oASIS beats SES and starts to approach the performance of PCA and SPCA. In contrast, both leverage score and random sampling perform considerably worse in terms of their factorization error.

**Results for LF Datasets**

In Fig. 5.6, we display the relative factorization error for the LF-i dataset. On the left we show the relative error as a function of \( L \) for sampling-based decompositions and SPCA/PCA and in the right column, we zoom into these figures for small values of \( L \). In both of these examples, we observe congruence between the error for oASIS, SES, and SPCA, and PCA; however, leverage and random sampling lag behind.

**Results for HS Datasets**

In Fig. 5.7, we display the relative factorization error for HS-i (uniform sampling \( N = 4.5k \)) and HS-ii (non-uniform sampling \( N = 11k \)) datasets. In both cases, oASIS and SES achieve a similar error decay as SPCA and PCA. In contrast, random sampling requires slightly more samples before achieving perfect factorization error.

**Results for MS Dataset**

To evaluate the performance of SEED for clustering motion trajectories, we utilize a video sequence from the JHU-155 database [67] consisting of two different objects moving (the first object is rotating and translating, the second object is rotating) while the camera is both rotating and translating. The video sequence consists of \( f = 24 \) frames with \( p = 430 \) key points, which generates a dataset of size \( M = 72 \) (3D coordinates of point correspondences tracked over 24 frames) and \( N = 430 \).

In Fig. 5.8, we display the relative factorization error for the MS dataset with three motions. In this example, we observe that oASIS achieves exact matrix re-
covery (error is less than $10^{-20}$) with $L = 50$, which is equivalent to the size of the factorization obtained via PCA and SPCA to achieve exact recovery. In contrast, SES, leverage, and random sampling require $L = \{55, 60, 65\}$ samples respectively, before achieving exact matrix recovery. This is one example where oASIS achieves exact matrix recovery with the number of samples equal to the rank of the data, which in this case, the rank equals $r = 49$.

**Results for US Dataset**

In this Section, we evaluate the performance of SEED for synthetic data generated from a union of overlapping subspaces generated according to the model described in Sec. 5.4.3. In Fig. 5.9, we display the factorization error as a function of $L/N$ for oASIS, SES, leverage scores, random sampling, PCA, and SPCA. We observe that oASIS achieves exact matrix recovery when $L = \text{rank}(X)$, while SES requires slightly more samples before achieving exact matrix recovery. In contrast, random sampling and leverage score sampling do not achieve exact matrix recovery for $L/N = 0.35$ because they do not sample all of the outlier points and thus exhibit a very slow decay in the factorization error in comparison with the other methods.
Figure 5.4: Factors selected from collections of face images. First 18 columns of left factor matrix for (a) oASIS, (b) SES, (c) Random, (d) SPCA (dense factors), (e) SPCA (sparse factors).
Figure 5.5: Factorization error for face image database. Relative factorization error vs. size of factorization for collections of (a) six subject’s faces under sixty different illumination conditions ($M = 4032, N = 360$) and (b) twenty subject’s faces under sixty different illumination conditions ($M = 4032, N = 1200$). In both cases, the data is full rank, i.e., $r = 360$ and $r = 1200$ for (a) and (b) respectively.
Figure 5.6: Factorization error for light field data. The factorization error for LF-i dataset is displayed as function of $L/N$ in (a) and in (b) we show the same results zoomed in.

Figure 5.7: Factorization error for hyperspectral image data. We display the factorization error as a function of $L/N$ for (a) HSI-i and (b) HSI-ii. On the right, we zoom into both (a) and (b) for small values of $L$ to more clearly display the initial decay in the error.
Figure 5.8: *Factorization error for motion segmentation data.* In (a), we display the factorization error for a 3 object motion sequence as a function of $L/N$ and in (b) we show the same results zoomed in.
Figure 5.9: Factorization error for synthetic union of subspace data corrupted by outliers. Relative error for a union of two 20-dimensional subspaces with outliers ($N_1 = 300$, $N_2 = 100$, $N_o = 50$), Overlapping dimensions $q = 3$. 
In this Chapter, we introduce a host of new methods for segmenting data into multiple subspaces with sparse self-expressive decompositions like those described in the previous Chapter, and provide extensive numerical simulations to compare a number of different subspace learning approaches on real world datasets.

### 6.1 Subspace Clustering

Unions of subspaces provide a powerful generalization of single subspace models for collections of high-dimensional data; however, learning multiple subspaces from data is challenging due to the fact that segmentation—the identification of points that live in the same subspace—and subspace estimation must be performed simultaneously. Recently, sparse recovery methods were shown to provide a provable and robust strategy for exact feature selection (EFS)—recovering subsets of points from the ensemble that live in the same subspace.

A common approach for identifying sets of points that live in the same subspace,
or perform *subspace clustering*, is to determine the ‘multi-way affinity’ between points in the set from locally linear approximations to the data [68]. To be precise, these methods compute the affinity between two test points by fitting a linear approximation to the points within an euclidean neighborhood of each test point and computing the similarity between these subspace estimates. After determining the affinity between points in the set, spectral clustering is performed on the resulting affinity matrix. Methods that use nearest neighbor sets to form locally linear approximations to data include: local subspace affinity (LSA) [69], spectral clustering based on locally linear approximations [68], spectral curvature clustering [70], and local best-fit flats [71, 72].

In contrast to clustering approaches which view the graph as encoding the subspace connectivity between points in the set, in consensus approaches, the goal is to utilize the *geometric features* contained in the edges of the graph. To be precise, for each vertex we determine the set of vertices for which an edge exists and map this sample set onto the Grassmanian manifold (set of all \( k \)-dimensional subspaces in \( \mathbb{R}^M \)). By looking at the span of these points, we obtain an estimate of a subspace structure that may be present in the ensemble. The idea is that by looking at a number of such mappings for different vertices in the graph, we can quickly converge to a correct estimate of the subspaces in the ensemble by finding the mode in the mappings. We point the reader to Vidal’s review in [13] on subspace clustering for a thorough description of the subspace clustering problem as well as methods for obtaining solutions to this problem.

When the subspaces present in the ensemble are independent and/or are linearly separable, linear approximations obtained from neighboring points typically provide reliable and stable estimates of the affinity between points in the ensemble. However, neighborhood-based approaches quickly begin to fail as the overlap between the two
structures increases and as the subspace dimension increases. This is due to the fact that as the overlap between two subspaces increases, the set of points that live in neighborhoods of one another are less likely to be contained within the same subspace. This suggests that if we can find another feature selection strategy that improves our probability of selecting a feature set that contains points from the same subspace, then we can use this alternate set of points to form a local subspace estimate\(^1\) instead of forming linear approximations from sets of near neighbors.

### 6.2 Sparse Subspace Clustering

Unions of subspaces provide a natural extension to single subspace (low rank) models; however, modeling collections of data with unions of subspaces is difficult because providing an extension of single subspace estimation techniques such as PCA to estimate multiple subspaces is extremely challenging. This is due to the fact that subspace clustering—or clustering points in accordance with their subspace membership—and subspace estimation must be performed jointly. Nevertheless, if one can accurately determine which points lie within the same subspace, then linear dimensionality reduction techniques can be applied to find a low-dimensional model for each subspace cluster.

Recently, sparse recovery methods have been shown to provide a provable strategy for identifying signals that belong to the same subspace. The motivation underlying this approach is that the sparse representation of a signal under consideration will consist of other signals from the same subspace. The sparse subspace clustering (SSC) algorithm starts by first forming a sparse self-expressive decomposition of the data as

\(^1\)We refer to subspace estimates as being local if they are formed from a subset of points in the ensemble. In contrast, we refer to standard low-rank approximation over the entire set of points as a global subspace estimate.
follows:

\[
\min_{V \in \mathbb{R}^{N \times N}} \|V\|_0 \quad \text{subject to} \quad \|X - XV\|_F \leq \epsilon, \ \text{diag}(V) = 0,
\]

(6.1)

where the constraint on the diagonal of \( V \) requires that each column of \( X \) is represented in terms of other columns in the matrix (but not itself) and \( \|V\|_0 \) counts the number of nonzeros in its matrix argument.

After computing a sparse decomposition of the data, the SSC algorithm [61] proceeds by forming a symmetric subspace affinity matrix as \( W = |V| + |V^T| \). The subspace affinity matrix \( W \) can be interpreted as a graph, where the \((i,j)\) entry of the matrix represents the edge between the \(i^{th}\) and \(j^{th}\) point in the ensemble; the strength of each edge represents the likelihood that two points live in the same subspace. After forming a symmetric affinity matrix, spectral clustering is then performed on the graph Laplacian [73] of \( W \) to obtain labels (indicating the subspace membership) for all the points in the ensemble. In Fig. 6.1 we demonstrate the power of sparse self-expressive decompositions: the affinity matrix on the left is the nearest neighbor graph for a collection of data (an edge is present when signals are nearest neighbors) and on the right, we show the affinity matrix obtained via SSC-OMP [15].

6.3 Greedy Feature Selection for Subspace Clustering

In this Section, we introduce two approaches for subspace clustering that employ greedy sparse recovery (OMP) to produce subspace affinity matrices for an ensemble of data. We demonstrate that greedy feature selection provides a number of distinct advantages over convex optimization-based approaches.
Figure 6.1: *Comparison of nearest neighbor affinity matrix and $\ell_0$-graph for unions of illumination subspaces.* The highlighted point corresponds to a hub where two images of different subjects are identified as near neighbors. On the right, we show that OMP removes spurious edges linking signals from different classes when we form sparse self-expressive representations of the data.

6.3.1 Greedy Spectral Clustering Approach (SSC-OMP)

Instead of solving the sparse recovery problem in (5.1) via $\ell_1$-minimization, we propose the use of a low-complexity method for sparse recovery known as orthogonal matching pursuit (OMP) in Alg. 1. For each point $\mathbf{x}_i$, we solve Alg. 1 to obtain a sparse representation of the signal with respect to the remaining points in $\mathbf{X}$. The output of the OMP algorithm is a feature set, $\Lambda^{(i)}$, which indexes the columns in $\mathbf{X}$ selected to form a sparse representation of $\mathbf{x}_i$. After computing feature sets for each point in the dataset via OMP, either a spectral clustering method or a consensus-based method [74] may then be employed to cluster the data. In Alg. 5, we outline a procedure for performing an OMP-based variant of the SSC algorithm that we will refer to as SSC-OMP. We detail the SSC-OMP algorithm in Alg. 5.
### Algorithm 5: Sparse Subspace Clustering with OMP (SSC-OMP)

**Input:** A data matrix $X \in \mathbb{R}^{M \times N}$ containing $N$ points $\{y_i\}_{i=1}^d$ in its columns, a stopping criterion for OMP, and the number of clusters $p$.

**Output:** A set of $N$ labels $\mathcal{L} = \{\ell(1), \ell(2), \ldots, \ell(d)\}$, where $\ell(i) \in \{1, 2, \ldots, p\}$ is the label associated with $x_i$.

**Step 1. Compute Subspace Affinities via OMP**
1. Solve Alg.1 for the $i^{th}$ signal $x_i$ to obtain a feature set $\Lambda^{(i)}$ and coefficient vector $c$.
2. For all $j \in \Lambda^{(i)}$, let $C_{ij} = c(j)$. Otherwise, set $C_{ij} = 0$.
3. Repeat steps (1)–(2) for all $i = 1, \ldots, N$.

**Step 2. Perform Spectral Clustering**
1. Symmetrize the subspace affinity matrix $C$ to obtain $W = |C| + |C^T|$.
2. Perform $p$-way spectral clustering on $W$ to obtain a set of $N$ labels $\mathcal{L}$.

### 6.3.2 Greedy Consensus Approach

When the probability of EFS decreases and more sample sets get corrupted, spectral clustering methods like those described above quickly degrade in their performance. Thus, we propose a consensus method for finding subspaces. The main idea behind our approach is to replace step (1) in existing consensus-based methods that use sets of near neighbors [72] with the feature sets selected via OMP. In contrast to clustering-based approaches where no guarantees can be made, when all of the points in the ensemble admit support sets with exact features, our consensus-based approach is guaranteed to recover the true subspaces underlying the data. This method is very similar in spirit to the iterative subspace identification approach proposed by Gowreesunker et al. in [75]. We first introduced this approach in [76] and detail this method in Alg. 6.

### 6.4 SEED-Clust Method

Rather than computing a $N \times N$ affinity matrix as in SSC, our goal is to cluster $X$ based upon the sparse representations of the data based upon the sparse decom-
Algorithm 6 : Sparse Subspace Consensus Method

**Input:** A collection of $N$ data points $X \in \mathbb{R}^{n \times d}$, subspace dimension $k$, number of points required for consensus $s$, threshold $\lambda$.

**Output:** A collection of ONBs $\{Q_i\}^p_{i=1}$, and the number of points that agree upon each of the $p$ subspace estimates $N = \{n_i\}^p_{i=1}$, where $n_i \geq s$ for all $i$.

Solve the support recovery problem in (5.1) for $X$ to obtain a collection of support sets $\{\Lambda(i)\}^d_{i=1}$.

for $i = 1, \ldots, N$ do

1. Compute an orthonormal basis $Q_i$ for which $\text{range}(X_{\Lambda(i)}) = \text{range}(Q_i)$.
2. Compute the energy of points in the sub-dictionary $X_{\Lambda(j)}$ when projected onto the subspace spanned by $Q_i$

$$d(i, j) = \sum_{n \in \Lambda(j)} (c(n)(I - Q_iQ_i^T)x_n)^2,$$

where $c_j(n)$ is the contribution of the $n^{th}$ point in $\Lambda(j)$ to the representation of $y_j$.
3. Count the number of points that agree upon the $i^{th}$ subspace estimate,

$$n_i = \sum_{j=1}^d T_\lambda(d(i, j)),$$

where $T_\lambda(\cdot) = 1$ when its argument is less than $\lambda$ and 0 otherwise.

end for

4. Place all unique projectors $Q_iQ_i^T$ for which $n_i \geq s$ into the set $\Gamma$.

---

position provided by SEED. We do this by simply performing co-clustering on the sparse representation of the unsampled signals, $V_{-S} \in \mathbb{R}^{L \times (N-L)}$, to simultaneously determine labels for both the sampled (rows) and unsampled signals (columns). To compute the factorization of $X_{-S}$ we use batch OMP to solve (L0-Sparse) by providing the maximum sparsity level $k_{\text{max}}$ to the algorithm. Fixing the sparsity level tends to provide better clustering results because the corresponding bi-partite graph (determined by the rows and columns of $V$) tend to be more balanced. We will now make the idea of interpreting $V_{-S}$ as a bi-partite graph precise.

### 6.4.1 Bipartite Graph Partitioning
Algorithm 7: SEED-Clust

**Input:** A dataset $X \in \mathbb{R}^{M \times N}$ containing $N$ points $\{x_i\}_{i=1}^N$ in its columns, number of columns to select ($L$), maximum sparsity ($K$), and the number of clusters ($P$).

**Output:** A set of $N$ labels $L$ containing the cluster ID for all $N$ signals in $X$.

**Step I. Sparse Self-Expressive Decomposition (SEED)**
1. Select $L$ columns via oASIS (Alg.3) and normalize the selected columns to form $D \in \mathbb{R}^{M \times L}$.
2. Solve OMP for each column of $X$ and stack it into the corresponding column of $V \in \mathbb{R}^{L \times N}$.

**Step II. Bipartite Graph Partitioning**
1. Compute the diagonal matrices $D_1$ and $D_2$ which contains the sum of the rows and columns of $V$ along their diagonals, respectively.
2. Form the matrix $X_n = D_1^{-1/2} V D_2^{-1/2}$.
3. Compute the second through $D + 1$ left and right singular vectors, $U$ and $V$ respectively, of $X_n$, where $U_k \in \mathbb{R}^{M \times D}$ and $V \in \mathbb{R}^{N \times D}$, where $D = \lceil \log_2(P) \rceil$.
4. Form the matrix $Z = [D_1^{-1/2} U_k, D_2^{-1/2} V_k]$ and apply k-means clustering to $Z$ to obtain labels for both the rows and columns of $V$.

The sparse $L \times (N - L)$ matrix $V_{-S}$ formed by computing the sparse representations of the unsampled columns $X_{-S} \in \mathbb{R}^{M \times (N-L)}$ can be interpreted as a bipartite graph, where the $(i, j)$ entry of the matrix represents the edge between the $i^{th}$ sampled point and the $j^{th}$ point in the remaining dataset; the strength of each edge represents the likelihood that two points live in the same subspace. Our goal is to cluster this bipartite graph so that we obtain a clustering of both the rows and columns of $V$ simultaneously. This gives both the partitioning/clustering of the sampled (rows) and unsampled (columns) data. This approach naturally reveals the partition of the entire dataset based upon the rectangular matrix $V$ rather than lifting this problem into a graph over the set of all $N$ points (requiring a $N \times N$ affinity matrix).

The entries of the matrix $V_{-S}$ can be considered as the edges of a bipartite graph, which is represented by the triple $G = (U, V, E)$, where $V$ is a set of $N$ vertices, and $E$ contains the edges between vertices in the sets $U$ and $V$. The edge between the $i^{th}$ vertex in $U$ and $j^{th}$ vertex in $V$ is given by $E(i, j) = V_{ij}$. We denote a “cut” or
partition of the bipartite graph with the sets \((U, V)\) containing the set of vertices in \(U\) and \(V\) respectively over which we partition the graph. The problem of co-clustering a bi-partite graph can be posed as finding a minimum cut (or normalized variant) through the graph that satisfies the following min-cut objective:

\[
\min_{(U,V)} \text{cut}(U,V) = \sum_{i \in U, j \in V} E(i, j),
\]

where this objective measures the sum of the weights across any cut (partitioning) across the rows and columns of \(V - S\).

The spectral co-clustering algorithm introduced in [77] provides an elegant relaxation of the spectral clustering problem for bi-partite graphs. We employ this method to partition \(V - S\) and provide an outline of the proposed algorithm in Alg. 7, which we refer to as SEED-Clust.

In Fig. 6.2, we provide a visualization of the feature matrix \(Z\) used to cluster the data; we show the first three coordinates of the \(Z\) matrix and plot the projection of the unsampled signals \(Z_2 = D_2^{-1/2}V_k\) as (red) dots and the projection of sampled signals \(Z_1 = D_1^{-1/2}U_k\) as (blue) stars. This synthetic example is created by generating signals from a union of five overlapping subspaces. We provide additional details regarding this experiment in Sec. 6.5.2.

### 6.4.2 Comparison of SEED and SSC-OMP

The first and most apparent advantage of SEED over other self-expressive methods such as SSC and LRR for subspace clustering is the fact that we compute a sparse representation of signals with respect to a smaller dictionary (\(L\) columns rather than \(N - 1\)). For this reason, SEED exhibits significantly less complexity than SSC and LRR. In addition to utilizing a smaller dictionary, SEED also uses the same dictionary for all signals—in contrast, SSC uses a different dictionary each time it com-
Figure 6.2: Visualization of coordinates of spectral co-clustering matrix for a union of five overlapping subspaces (US-i). On the left, we display the coordinates for random sampling and on the right, we display the coordinates for oASIS sampling. In each row, we show the results for $L = 100$ (top), $L = 200$ (middle), and $L = 300$ (bottom).

Computes a sparse representation. For this reason, we employ a batch OMP algorithm to accelerate the computation of sparse approximation for large $N$. The proposed clustering algorithm, SEED-CLUST, also solves a bipartite graph partitioning problem on a $L \times N$ matrix instead of a graph partitioning problem on a $N \times N$ matrix (as in SSC); to solve the corresponding co-clustering problem, we must compute the second largest singular vectors of a $L \times N$ matrix rather than the second smallest eigenvectors of a $N \times N$ matrix. Thus, SEED-Clust can be used on massive datasets by iteratively solving for the singular vectors more efficiently (can use an iterative power method to find leading singular vectors).
In addition to computational advantages of using SEED for subspace clustering, an interesting consequence of subsampling the data with SEED is that, in some cases, this approach actually provides better cut ratios than SSC and nearest neighbor (NN) graphs. The cut ratio provides a measure about the sum of the edges outside of a correct cluster relative to the total edges within and across clusters; thus, by subsampling the data, we remove subsets of points from the dictionary that otherwise introduce spurious edges into the bipartite graph. Another advantage of using SEED instead of SSC is that by using an incoherent reference set, the energy of sparse feature vectors in the columns of $V$ are more localized. The main intuition behind this observation is that when we have multiple signals that can be used almost interchangeably, then the sparse representation of multiple signals that use either of these examples, will be spread across both examples, rather than concentrated. As a result, the corresponding affinity matrix is often too sparse to reliably detect subspace clusters with spectral clustering approaches. In contrast, the affinities obtained by maximizing incoherence between samples, are far more concentrated and produce well-connected components within the affinity matrix.

6.5 Numerical Experiments

In this Section, we evaluate the performance of both SSC-OMP and SEED for subspace clustering. To do this, we will examine the rates of EFS, the normalized cut ratios, and in some cases we also report the final clustering performance of these methods.

In Sec. 6.5.1, we compare the performance of greedy feature selection with SSC-OMP to SSC with BP and also to NN graphs. The aim of these experiments is to fully characterize the advantages of greedy methods over other sparse approximation methods and thus, we do not utilize any subsampling in our evaluations. In Sec. 6.5.2,
we compare the performance of different sampling strategies for SEED and when appropriate, compare our results with SPCA, PCA, SSC-OMP, and NN methods.

6.5.1 Clustering Performance of SSC-OMP

In our theoretical analysis of EFS in Sec. 7.1, we reveal an intimate connection between the covering radius of subspaces and the principal angles between pairs of subspaces in the ensemble. In this section, we will conduct an empirical study to explore these connections. In particular, we will study the probability of EFS as we vary the covering radius as well as the dimension of the intersection and/or overlap between subspaces.

Phase Transitions for SSC-OMP

The goal of our first experiment is to study the probability of EFS—the probability that a signal in the dataset admits exact features—as we vary both the number and distribution of points in each subspace as well as the dimension of the intersection between subspaces. For this set of experiments, we generate a union of orthoblock sparse signals, where the overlap equals the dimension of the intersection.

Along the top row of Fig. 6.3, we display the probability of EFS for orthoblock sparse signals generated according to the coefficient model (M1) in Sec. 5.4.3: the probability of EFS is computed as we vary the overlap ratio \( \delta = q/k \in [0, 1] \) in conjunction with the oversampling ratio \( \rho = k/d \in [0, 1] \), where \( q \) equals the dimension of the intersection between the subspaces, and \( d \) is the number of points per subspace. Along the bottom row of Fig. 6.3, we display the probability of EFS for orthoblock sparse signals generated according to the coefficient model (M2) in Sec. 5.4.3: the probability of EFS is computed as we vary the overlap ratio \( \delta \) and the amount of energy \( \tau \in [0, 1] \) each point has within its common component. For these experiments,
Figure 6.3: *Probability of EFS for different coefficient distributions.* The probability of EFS for a union of two subspaces of dimension $k = 20$ (left column) and $k = 50$ (right column). The probability of EFS is displayed as a function of the overlap ratio $\delta \in [0, 1)$ and the logarithm of the oversampling ratio $\log(\rho)$ (top row) and the mutual energy $\tau = \|y_c\|_2$ (bottom row).

As our theory predicts, the oversampling ratio has a strong impact on the degree of overlap between subspaces that can be tolerated before EFS no longer occurs. In particular, as the number of points in each subspace increases (covering radius decreases), the probability of EFS obeys a second-order phase transition, i.e., there is a graceful degradation in the probability of EFS as the dimension of the intersection increases. When the pair of subspaces are densely sampled, the phase boundary is shifted all the way to $\delta = 0.7$, where 70% of the dimensions of each subspace...
intersect. This is due to the fact that as each subspace is sampled more densely, the covering radius becomes sufficiently small to ensure that even when the overlap between planes is high, EFS still occurs with high probability. In contrast, when the subspaces are critically sampled, i.e., the number of points per subspace \( d \approx k \), only a small amount of overlap can be tolerated, where \( \delta < 0.1 \). In addition to shifting the phase boundary, as the oversampling ratio increases, the width of the transition region (where the probability of EFS goes from zero to one) also increases.

Along the bottom row of Fig. 6.3, we study the impact of the bounding constant on EFS, as discussed in Section 7.1.6. In this experiment, we fix the oversampling ratio to \( \rho = 0.1 \) and vary the common energy \( \tau \) in conjunction with the overlap ratio \( \delta \). By reducing the bounding constant of the union, the phase boundary for the uniformly distributed data from model \((M1)\) is shifted from \( \delta = 0.45 \) to \( \delta = 0.7 \) for both \( k = 20 \) and \( k = 50 \). This result confirms our predictions in the discussion of Thm. ?? that by reducing the amount of energy that points have in their subspace intersections EFS will occur for higher degrees of overlap. Another interesting finding of this experiment is that, once \( \tau \) reaches a threshold, the phase boundary remains constant and further reducing the bounding constant has no impact on the phase transitions for EFS.

**Comparison of OMP and NN**

In this section, we compare the probability of EFS for feature selection with OMP and nearest neighbors (NN). First, we compare the performance of both feature selection methods for unions with different cross-spectra. Second, we compare the phase transitions for unions of orthoblock sparse signals as we vary the overlap and oversampling ratio.

For our experiments, we generate pairs of subspaces with structured cross-spectra
Figure 6.4: Probability of EFS for unions with structured cross-spectra. Along the top row, we show the cross-spectra for different unions of block-sparse signals. Along the bottom row, we show the probability of EFS as we vary the overlap ratio $\delta \in [0, 1]$ for OMP (solid) and NN (dash).

as described in Section 5.4.3. The cross-spectra arising from three different unions of block-sparse signals are displayed along the top row of Fig. 6.4. On the left, we show the cross-spectra for a union of orthoblock sparse signals with overlap ratio $\delta = 0.75$, where $q = 15$ and $k = 20$. The cross-spectra obtained by pairing shifted Lorentzian and exponential atoms are displayed in the middle and right columns, respectively. Along the bottom row of Fig. 6.4, we show the probability of EFS for OMP and NN for each of these three subspace unions as we vary the overlap $q$. To do this, we generate subspaces by setting their cross-spectra equal to the first $q$ entries equal to the cross-spectra in Fig. 6.4 and setting the remaining $k-q$ entries of the cross-spectra equal to zero. Each subspace cluster is generated by sampling $d = 100$ points from each subspace according to the coefficient model (M1) in Sec. 5.4.3.

This study provides a number of interesting insights into the role that higher-order principal angles between subspaces play in feature selection for both sparse recovery.
methods and NN. First, we observe that the gap between the probability of EFS for OMP and NN is markedly different for each of the three unions. In the first union of orthoblock sparse signals, the probability of EFS for OMP lies strictly above that obtained for the NN method, but the gap between the performance of both methods is relatively small. In the second union, both methods maintain a high probability of EFS, with OMP admitting nearly perfect feature sets even when the overlap ratio is maximal. In the third union, we observe that the gap between EFS for OMP and NN is most pronounced. In this case, the probability of EFS for NN sets decreases to 0.1, while OMP admits a very high probability of EFS, even when the overlap ratio is maximal. In summary, we observe that when data is distributed uniformly with respect to all of the principal directions between a pair of subspaces and the cross-spectra is sub-linear, then EFS may be guaranteed with high probability for all points in the set provided the sampling density is sufficiently high. This is in agreement with the discussion of EFS bounded unions in Section 7.1.6. Moreover, these results further support our claims that in order to truly understand and predict the behavior of endogenous sparse recovery from unions of subspaces, we require a description that relies on the entire cross-spectra.

In Fig. 6.5, we display the probability of EFS for OMP (left) and sets of NN (right) as we vary the overlap and the oversampling ratio. For this experiment, we consider unions of orthoblock sparse signals living on subspaces of dimension $k = 50$ and vary $\rho \in [0.2, 0.96]$ and $\delta \in [1/k, 1]$. An interesting result of this study is that there are regimes where the probability of EFS equals zero for NN but occurs for OMP with a non-trivial probability. In particular, we observe that when the sampling of each subspace is sparse (the oversampling ratio is low), the gap between OMP and NN increases and OMP significantly outperforms NN in terms of their probability of EFS. Our study of EFS for structured cross-spectra suggests that the gap between NN and
Figure 6.5: *Phase transitions for OMP and NN.* The probability of EFS for orthoblock sparse signals for OMP (a) and NN (b) feature sets as a function of the oversampling ratio $\rho = k/d$ and the overlap ratio $\delta = q/k$, where $k = 20$.

OMP should be even more pronounced for cross-spectra with superlinear decay.

**Results for YF Dataset**

In this section, we compare the performance of sparse recovery methods, i.e., BP and OMP, with NN for clustering unions of *illumination subspaces* arising from a collection of images of faces under different lighting conditions. By fixing the camera center and position of the person’s face and capturing multiple images under different lighting conditions, the resulting images can be well-approximated by a 5-dimensional subspace [23].

In Fig. 6.6, we show three examples of the subspace affinity matrices obtained with NN, BP, and OMP for two different faces under 64 different illumination conditions from the Yale Database B [66], where each image has been subsampled to $48 \times 42$ pixels, with $n = 2016$. In all of the examples, the data is sorted such that the images for each face are placed in a contiguous block.

To generate the NN affinity matrices in the left column of Fig. 6.6, we compute the absolute normalized inner products between all points in the dataset and then
Figure 6.6: Comparison of subspace affinity matrices for illumination subspaces. In each row, we display the affinity matrices obtained for a different pair of illumination subspaces, for NN (left), BP (middle), and OMP (right). To the left of the affinity matrices, we display an exemplar image from each illumination subspace.

threshold each row to select the $k = 5$ nearest neighbors to each point. To generate the OMP affinity matrices in the right column, we employ Step 1 of Alg. 5 with the maximum sparsity set to $k = 5$. To generate the BP affinity matrices in the middle column, we solved the BP denoising (BPDN) problem in (3.9) via a homotopy algorithm where we vary the noise parameter $\kappa$ and choose the smallest value of $\kappa$ that
produces $k \leq 5$ coefficients. The resulting coefficient vectors are then stacked into the rows of a matrix $C$ and the final subspace affinity $W$ is computed by symmetrizing the coefficient matrix, $W = |C| + |C^T|$.

After computing the subspace affinity matrix for each of these three feature selection methods, we employ a spectral clustering approach which partitions the data based upon the eigenvector corresponding to the smallest nonzero eigenvalue of the graph Laplacian of the affinity matrix [73, 78]. For all three feature selection methods, we obtain the best clustering performance when we cluster the data based upon the graph Laplacian instead of the normalized graph Laplacian [73]. In Table 6.1, we display the percentage of points that resulted in EFS and the classification error for all pairs of $\binom{38}{2}$ subspaces in the Yale B database. Along the top row, we display the mean and median percentage of points that resulted in EFS for the full dataset (all 64 illumination conditions), half of the dataset (32 illumination conditions selected at random in each trial), and a quarter of the dataset (16 illumination conditions selected at random in each trial). Along the bottom row, we display the clustering error (percentage of points that were incorrectly classified) for SSC-OMP, SSC, and NN-based clustering (spectral clustering of the NN affinity matrix).

While both sparse recovery methods (BPDN and OMP) admit EFS rates that

<table>
<thead>
<tr>
<th></th>
<th>Full-data</th>
<th></th>
<th>Half-data</th>
<th></th>
<th>Quarter-data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OMP $\ell_1$</td>
<td>NN</td>
<td>OMP $\ell_1$</td>
<td>NN</td>
<td>OMP $\ell_1$</td>
</tr>
<tr>
<td>EFS (%)</td>
<td>Mean</td>
<td>Median</td>
<td>Mean</td>
<td>Median</td>
<td>Mean</td>
</tr>
<tr>
<td></td>
<td>55.48</td>
<td>73.48</td>
<td>60.71</td>
<td>39.74</td>
<td>52.50</td>
</tr>
<tr>
<td></td>
<td>55.91</td>
<td>75.00</td>
<td>62.5</td>
<td>39.06</td>
<td>53.13</td>
</tr>
<tr>
<td>Clustering error (%)</td>
<td>Mean</td>
<td>1.43</td>
<td>3.69</td>
<td>22.03</td>
<td>4.91</td>
</tr>
<tr>
<td></td>
<td>Median</td>
<td>0.78</td>
<td>0.78</td>
<td>15.63</td>
<td>3.13</td>
</tr>
</tbody>
</table>

Table 6.1: Classification and EFS rates for illumination subspaces. Shown are the aggregate results obtained over $\binom{38}{2}$ pairs of subspaces.
Figure 6.7: Cross-spectra for illumination subspaces. We display the cross-spectra for reduced data which has been projected onto a 10D subspace (left) and the cross-spectra obtained from the raw data (right). In each subplot, we overlay the cross-spectra between subspaces \((W_1, W_2)\) (solid) where \(\|\sigma\|_2 = 2.10\), \((W_2, W_3)\) (dash) where \(\|\sigma\|_2 = 2.12\), and between \((W_1, W_3)\) where \(\|\sigma\|_2 = 2.30\) (star).

are comparable to NN on the full dataset, we find that sparse recovery methods provide higher rates of EFS than NN when the sampling of each subspace is sparse, i.e., the half and quarter datasets. These results are also in agreement with our experiments on synthetic data. A surprising result is that OMP provides better clustering performance than BP on this particular dataset, even though BP provides higher rates of EFS.

Results for MS Dataset

In Fig. 6.8 and Fig. 6.9, we compare the results obtained on the Hopkins155 database [67] with \(\ell_0\)-consensus (labeled SSC-Grass) to those obtained with other existing methods, including SSC. We note that both SSC and our method obtain state-of-the-art performance in comparison with other existing methods. In Figure 6.8, we show the classification performance obtained from segmenting video sequences with only two rigid body motions and in Figure 6.9 we show the results from segmenting three motions.
Figure 6.8: Classification performance for segmenting two motions. We show the classification rates for the Hopkins155 Database for segmenting two rigid body motions from point correspondences.

Figure 6.9: Classification performance for segmenting three motions. We show the classification rates for the Hopkins155 Database for segmenting three rigid body motions from point correspondences.

**Comparison of Consensus and Spectral Clustering**

In Fig. 6.10, we compare the performance of our $\ell_0$-consensus approach with the equivalent spectral clustering formulation on the $\ell_0$-graph proposed in [61]. We also compare these methods with a slightly modified version of SSC, where instead of clustering the eigenvector corresponding to the smallest non-zero eigenvalue of the graph
Laplacian, we select the set of $k$ largest and $k$ smallest entries in this vector. These sets correspond to two sets of $k$ points from each cluster that are most separated with respect to their edge weights on the $\ell_0$-graph.

In Fig. 6.10, we observe similar behavior in the subspace recovery from $\ell_0$-graphs our slight modification to SSC and for $\ell_0$-consensus. However, when $q/k \geq 0.8$, we see a quick drop in the probability of recovery for modified SSC and we maintain a non-zero probability of recovery with $\ell_0$-consensus, even when $q/k = 0.9$. In contrast, when we perform standard spectral clustering, we observe a decrease in the probability of recovery when $q/k = 0.7$. These results suggest that $\ell_0$-graphs provide reliable feature sets for both clustering and consensus, even for high degrees of overlap. However, consensus can be used in settings where high degrees of overlap exist to maintain reliable recovery performance even when spectral clustering methods begin to fail.

In Fig. 6.11, we show the gap between modified SSC and $\ell_0$-consensus when we vary $s$ (the number of points that we require to form consensus). We see that the
Figure 6.11: Performance of subspace consensus. In each plot, we overlay the probability of recovery for (⋄) \( \ell_0 \)-consensus, (solid) modified SSC, and (dash) the empirical probability of EFS. The results are averaged across 50 trials with \( k = 10 \) and \( \lambda = 1e - 5 \). On the (left) \( s = 5 \), (middle) \( s = 10 \), and (right) \( s = 20 \). The number of points in each subspace is set to 400 and 200 in the top and bottom rows respectively.

The gap between these methods increases as we require less confidence in the estimate. However, if we require a large degree of confidence for our estimate \( s > 10 \), the performance of \( \ell_0 \)-consensus is very similar to the performance of modified SSC as we vary the overlap between subspaces in the ensemble.

### 6.5.2 Clustering Performance for SEED

In this Section, we evaluate the performance of SEED for dimensionality reduction and clustering for the five datasets described in Sec. 5.4.2.
Evaluation Setup

To evaluate the performance of SEED, we compute a $L$-dimensional sparse factorization of the data $\hat{X} = DV$, where $D \in \mathbb{R}^{M \times L}$ with: (i) oASIS sampling, (ii) sequential error selection (SES), (iii) leverage scores, and (iv) uniform random sampling. We compare these different sampling-based approaches with a generalized power method for SPCA [41]. When computing the sparse principal components, we set the regularization parameter applied to the sparse penalty in SPCA to $\gamma = 0.1$. See Sec. 5.4.1 for more details regarding the generalized method employed for SPCA and our choice of regularization parameter. In all of the subsequent experiments, to ensure that the sparsity of the final factorizations obtained via SPCA and SEED are equivalent, when computing a sparse factorization $\hat{X} = DV$ with SPCA, we compute $D$ by solving Eq. 4.2 and then use the same sparse recovery procedure (including degree of sparsity) to compute the sparse matrix $V$ for all of the methods.

In addition to studying the clustering performance of the aforementioned factorization approaches, we also compare the performance of SEED with the SSC-OMP algorithm [15] and the nearest neighbor graph (NN) for the dataset. To compute the NN graph for a dataset, we (i) compute the absolute normalized inner products between all points in the dataset and (ii) threshold each row to select the $k$ nearest neighbors to each point. To compute the sparse principal components, we set $\gamma = 0.1$, where $\gamma$ is the regularization parameter used to enforce a sparse penalty on the coefficients obtained by factorizing the data as $\hat{X} = DV$, where $D$ are the principal components learned via SPCA. In all of our experiments, we set the stopping criterion $k_{\text{max}}$ to be the same for all of the methods (SSC-OMP, NN, SPCA, and SEED).

In the subsequent experiments, we compute the cost of a normalized cut as we vary the size of the factorization $L$. We define the cost of a normalized cut as follows. Let $R_k$ index the rows of $V$ corresponding to points in the $k^{\text{th}}$ class, $C_k$ index the set
of columns corresponding to the points in the $k^{th}$ class, and $\Omega$ index all $N$ points in the dataset. The cost of the normalized cut for the $k^{th}$ class is defined as:

$$\text{ncut}(V, R_k, C_k) = \frac{\sum_{i \in R_k, j \in C_k} |V_{ij}|}{\sum_{i \in R_k, j \in \Omega} |V_{ij}|} + \frac{\sum_{i \in R_k, j \in C_k} |V_{ij}|}{\sum_{i \in \Omega, j \in C_k} |V_{ij}|}.$$ 

**Results for YF Datasets**

To evaluate SEED for clustering face images, we utilize two different datasets sampled from the extended Yale B Image Database [66]. We subsample each image so that they are $48 \times 84$ pixels and then reorder the pixels into a vector of dimension $M = 4038$. Our first dataset YF-i, consists of 6 subject’s faces and our second dataset, YF-ii, consists of 20 subject’s faces under 60 different illumination conditions, resulting in a dataset of size $N = 360$ and $N = 1200$ respectively. In Fig. 6.1, we show an example of two subject’s faces from this dataset and the affinity matrices generated via NN methods (left) and OMP (right).

In Fig.6.12, we display the normalized cut ratios for YF-i and YF-ii as a function of the relative factorization size $L/N$. We observe that the cut ratios obtained via SEED (for all sampling methods) outperform SPCA by a considerable margin. Moreover, after the relative sampling $L/N$ exceeds 30%, both SES and oASIS obtain better cut ratios than the NN graph and SSC; however, even after sampling 50% of the samples, leverage score and random sampling do not achieve the same cut ratios as NN and SSC.

**Results for HS Datasets**

In Table 6.2, we display the minimum (across all classes) and average cut ratios for $L = 10$ and $L = 60$ for HS-i and HS-ii. In both experiments, we set $k_{\text{max}} = 10$ and $\epsilon = 0.2$. Although SSC achieves the best cut ratios, when $L = 10$ we obtain EFS.
Figure 6.12: Normalized cut ratios for face image database. (top) Normalized cut ratios and (bottom) Relative factorization error vs. size of factorization for collections of twenty subject’s faces under sixty different illumination conditions ($M = 4032, N = 1200$) The data is full rank ($r = 1200$).

with oASIS for one of the classes for both HS-i and HS-ii; for $L = 60$, oASIS achieves comparable minimum and mean cut ratios to SSC and NN at a fraction of the cost. For $L = 60$ samples, oASIS obtains the best minimum and mean cut ratios over all decomposition methods (including other sampling methods and SPCA). In all cases,
SPCA achieves the worse case cut ratios in comparison with all other methods.

To provide a visualization of the differences across various sampling strategies, in Fig. 6.13, we display the sparse coefficient matrices obtained via SEED for the HS-i dataset, where we display (left) oASIS, (middle) SES, and (right) random sampling. One of the most striking differences between these coefficient matrices is seen in the result obtained via oASIS. In particular, we observe that oASIS only selects a single sample from the third class, while selecting a significantly greater number of samples from the second class. Upon investigating the spectral signatures of these two classes, we observe that the third class is highly correlated and much lower-dimensional than the second class; thus, oASIS only requires a single sample from the second class to accurately represent the remaining signatures in the class. We see a similar trend in the SES samples (less samples for the third class) but the inter-class distribution appears more uniform. As expected, the random sampling strategy selects roughly equal numbers of samples from each class.

In Fig. 6.14, we show the symmetrized affinities obtained by computing $\mathbf{W} = \mathbf{V}^T \mathbf{V}$ for the coefficient matrices in Fig. 6.13 obtained via oASIS and random sampling as well as SPCA. In addition, we show the affinities obtained via SSC-OMP. For both oASIS and random sampling, we observe an extremely dense clustering of points in the block diagonal components of the subspace affinity matrix and some false edges in the off-diagonal components. In contrast, SPCA confuses signals from the third and fourth class, making it extremely hard to segment these two classes. The affinities obtained via SSC-OMP are much sparser than the other factorization-based methods but also admit EFS and thus have no false edges linking signals in different classes.
Figure 6.13:  
Sparse coefficient matrices obtained for hyperspectral image data. Sparse coefficient matrices obtained via (a) oASIS, (b) SES, and (c) Random sampling for HS-i data.

Figure 6.14:  
Affinity matrices for hyperspectral image data. The symmetrized affinity matrices obtained for HS-i via oASIS, random sampling, SSC-OMP, and SPCA are displayed ($L = 60$).
Table 6.2: Comparison of normalized graph cut metrics (min, mean) for hyperspectral image data. The minimum and average cut ratios are displayed for HSI-i (uniform) and HSI-ii (non-uniform) for values of $L = \{10, 60\}$.

<table>
<thead>
<tr>
<th>Method</th>
<th>HSI-i (L = 10)</th>
<th>HSI-i (L = 60)</th>
<th>HSI-ii (L = 10)</th>
<th>HSI-ii (L = 60)</th>
</tr>
</thead>
<tbody>
<tr>
<td>oASIS</td>
<td>(0.51e−2)</td>
<td>(0.80e−3)</td>
<td>(0.76e−2)</td>
<td>(0.29e−3)</td>
</tr>
<tr>
<td>SES</td>
<td>(5.6e−3, 3.2e−2)</td>
<td>(4.5e−3, 8.5e−3)</td>
<td>(9.8e−3, 1.7e−1)</td>
<td>(0.147e−2)</td>
</tr>
<tr>
<td>Rand</td>
<td>(5.6e−3, 3.63e−2)</td>
<td>(0.93e−3)</td>
<td>(8.9e−3, 1.6e−1)</td>
<td>(0.47e−2)</td>
</tr>
<tr>
<td>SPCA</td>
<td>(4.9e−1, 7.8e−1)</td>
<td>(1.8e−2, 3.4e−1)</td>
<td>(1.91e−1, 8.9e−1)</td>
<td>(6.1e−3, 5.3e−1)</td>
</tr>
<tr>
<td>SSC</td>
<td>(0.44e−4)</td>
<td>(0.44e−4)</td>
<td>(0.27e−3)</td>
<td>(0.27e−3)</td>
</tr>
<tr>
<td>NN</td>
<td>(6.0e−4, 3.6e−3)</td>
<td>(6.0e−4, 3.6e−3)</td>
<td>(0.28e−3)</td>
<td>(0.28e−3)</td>
</tr>
</tbody>
</table>

Results for MS Dataset

To evaluate the performance of SEED for clustering motion trajectories, we utilize a video sequence from the JHU-155 database [67] consisting of two different objects moving (the first object is rotating and translating, the second object is rotating) while the camera is both rotating and translating. The video sequence consists of $f = 24$ frames with $N = 430$ key points, which generates a dataset of size $M = 72$ (3D coordinates of point correspondences tracked over 24 frames) and $N = 430$.

To examine the cut ratios obtained via SEED and other methods, we set $k_{\text{max}} = 5$ and the maximum error to $\epsilon = 0.05$. In this case, SSC achieves EFS for all classes.

Table 6.3: Comparison of normalized graph cut metrics for motion segmentation data.

<table>
<thead>
<tr>
<th>Method</th>
<th>$L = 20$</th>
<th>$L = 40$</th>
<th>$L = 60$</th>
</tr>
</thead>
<tbody>
<tr>
<td>oASIS</td>
<td>7.88e−2</td>
<td>8.7e−3</td>
<td>1.4e−3</td>
</tr>
<tr>
<td>SES</td>
<td>4.27e−2</td>
<td>1.0e−2</td>
<td>5.9e−3</td>
</tr>
<tr>
<td>Lev</td>
<td>7.5e−2</td>
<td>4.35e−2</td>
<td>2.72e−2</td>
</tr>
<tr>
<td>Rand</td>
<td>1.01e−1</td>
<td>3.17e−2</td>
<td>2.23e−2</td>
</tr>
<tr>
<td>SPCA</td>
<td>2.33e−1</td>
<td>7.6e−3</td>
<td>5e−3</td>
</tr>
<tr>
<td>NN</td>
<td>7.9e−3</td>
<td>7.9e−3</td>
<td>7.9e−3</td>
</tr>
</tbody>
</table>
Table 6.4: Normalized cut metrics for synthetic data living on a union of five overlapping subspaces (US-i).

<table>
<thead>
<tr>
<th>Method</th>
<th>US (L=100)</th>
<th>US (L=300)</th>
</tr>
</thead>
<tbody>
<tr>
<td>oASIS</td>
<td>4.09e-2</td>
<td>2.0e-4</td>
</tr>
<tr>
<td>SES</td>
<td>4.98e-2</td>
<td>4.0e-4</td>
</tr>
<tr>
<td>Rand</td>
<td>1.31e-1</td>
<td>1.05e-2</td>
</tr>
<tr>
<td>Gram</td>
<td>4.60e-1</td>
<td>4.60e-1</td>
</tr>
<tr>
<td>SSC</td>
<td>1.5e-3</td>
<td>1.5e-3</td>
</tr>
<tr>
<td>NN</td>
<td>6.8e-3</td>
<td>6.8e-3</td>
</tr>
</tbody>
</table>

Results for US Data

In this Section, we evaluate the performance of SEED and SSC-OMP on synthetic datasets consisting of points from a union of overlapping subspaces generated according to the model described in Sec. 5.4.3.

In our first experiment, we synthesize a collection of data from a union of five 20-dimensional subspaces with a uniform distribution of points in each subspace \( N_i = 100 \). We synthesize the data such that two pairs of subspaces have a 10-dimensional overlap (each pair of 20-dimensional subspaces share a 10-dimensional intersection) and the fifth subspace is independent from the rest. The ambient dimension is set to \( M = 200 \), the total number of points equals \( N = 500 \), and the rank of the dataset equals \( r = 80 \). In Table 6.4, we report to minimum cut ratios for this dataset when we select \( L = 100 \) and \( L = 300 \) samples. An interesting result of this simulation is that when we select \( L = 300 \), oASIS produces better cut ratios than using all \( N = 500 \) points with SSC-OMP. In Fig. 6.2, we show the first three coordinates of the co-clustering matrix \( Z \) obtained via SEED-Clust for oASIS and random sampling. This figure demonstrates that we obtain good separation of all five subspaces with oASIS with far fewer samples than random sampling. In this example, oASIS separates all five clusters with only \( L = 100 \) samples but with the same number of random samples, we can only discern four clusters.
To evaluate the performance of SEED and SSC-OMP in the presence of outliers, we synthesize a collection of signals from a union of two overlapping 20-dimensional subspaces, where the dimension of the intersection between the two subspaces is $q = 3$, and the ambient dimension $M = 200$. The number of points per subspace is non-uniformly distributed: the number of points in the first subspace $N_1 = 300$, the number of points in the second subspace $N_2 = 100$, and the number of outlier points equals $N_o = 50$. The outliers are synthesized by generating Gaussian random vectors. In Fig. 5.1, we display the sparse coefficient matrices obtained via oASIS (left), random sampling (middle), and SPCA (right). There are a number of interesting points to make regarding Fig. 5.1. First, if we compare the coefficient matrices created by sampling $L = 160$ columns via oASIS (left) and random sampling (middle), we observe that: (i) random sampling selects too many points from the first subspace and does not select a sufficient number of outlier points, (ii) oASIS samples all of the outlier points and preserves a more equal distribution of points from both low-dimensional subspaces. If we examine the distribution of points selected when
we sample \( L = \text{rank}(X) = 87 \) columns, oASIS returns exactly 20 points from the first subspace, 17 points from the second subspace, and all 50 outlier points. Second, we observe that the principal components learned by SPCA \((L = 60)\) mixes the first and second subspace clusters (adding additional PCs degrades the clustering performance).

Finally, we generate an equal number of points from each subspace \( N_1 = N_2 = 200 \) and corrupt the data with \( N_o = 20 \) outliers (otherwise using the same parameters as in the last experiment). In Fig. 6.15, we display the first two coordinates of the \( Z \) matrix obtained via the SEED-Clust algorithm for oASIS and SPCA. where the coordinates for the two subspaces are displayed as red and blue points and the outliers are plotted as green stars. In the case of oASIS, the points corresponding to signals in the two overlapping subspaces exhibit some overlap due to the fact the subspaces overlap, however, the points corresponding to outliers are very well separated from the subspace signals. In contrast, SPCA does a good job of separating outliers but the signals from the two overlapping subspaces are more difficult to separate from one another.
In this Chapter, we provide a theoretical analysis of self-expressive sparse recovery and SEED.

7.1 Analysis of Exact Feature Selection

7.1.1 Exact Feature Selection

In our subsequent analysis, we will develop sufficient conditions that describe when the sparse representation returned by OMP consists of points that belong to the same subspace. This can be used to guarantee that we can utilize sparse recovery to perform subspace clustering and other sparse-representation based classification tasks. We will refer to this event as exact feature selection (EFS). A formal definition is provided below.

**Definition 5 (Exact Feature Selection)** Assume that $x \in \text{span}(D_T)$ and let $v$ be the sparse representation of $x$ with respect to $D$. When EFS occurs for the signal $x$ with respect to the subset $T$, when its sparse representation $v$ is supported over some
set of atoms indexed by $T$, or $\text{supp}(v) \subseteq T$.

Note that the exact recovery condition introduced in [79] is equivalent to EFS when $\text{supp}(v) = T$ and $T$ indexes a set of linearly independent atoms.

**Implications of EFS for Subspace Clustering**

Exact feature selection can be used to characterize the performance of algorithms for unsupervised subspace learning problems. When EFS occurs for a point in the ensemble, this will produce a subspace estimate that coincides with one of the true subspaces contained within the data. In the case of spectral clustering-based approaches, EFS guarantees that no edges exist in the graph between any signal $x \in S$ and a signal in a different subspace. Thus, EFS provides a natural metric for studying the performance of both subspace consensus and subspace clustering methods that are based upon supports drawn from the data. EFS is also important in supervised learning problems that rely on sparse representations of the data to determine the class membership of a new point as in [54]. In supervised learning problems, EFS is not required to guarantee accurate classification but is sufficient to ensure that accurate classification occurs; however, by studying the fundamental properties of the ensemble that govern EFS, we may also understand supervised classification methods that employ self-expressive sparse recovery more deeply as well.

We now provide a few remarks regarding the implications of EFS for different subspace clustering algorithms. As we discussed previously, EFS is intimately linked to the probability that we exactly recover the subspaces present in our ensemble. In particular, consensus methods are guaranteed to recover the subspaces present in the ensemble, as long there are a sufficient number of points that admit exact features across the dataset. Thus, the probability of EFS provides an explicit lower bound for the probability of recovering a sufficient number of correct local subspace estimates;
this will in turn lead to accurate recovery of the subspaces in our union.

In contrast, even when all of the points in the set admit exact features, this is not sufficient to guarantee that spectral clustering based methods like sparse subspace clustering (SSC) \cite{61} will recover the correct set of subspaces from the data. This is due to the fact that even for graphs with no links across subspaces, spectral clustering or graph cuts may still be unwieldy due to scaling and normalization issues. In practice, we find that in a number of settings, spectral clustering over the $\ell_0$-graph will often recover small clusters containing less than $k$ points from the same subspace. Even after removing these points, the same issues arise in subsequent iterations. This results in clusterings consisting of a large number of small clusters, from which the true union of subspaces underlying the data can not be ascertained. This issue is even more pronounced in real-world datasets.

In contrast to clustering-based approaches, consensus methods are designed in a way such that they to remain robust to this degradation in the probability of EFS. Thus, consensus methods lend themselves well to settings where the probability of selecting sets of points with exact features is bounded above zero but not equal to one. For this reason, consensus approaches provide a natural means by which we can obtain efficient solutions to subspace learning problems when high-degrees of overlap exist or in settings where each point in the set can not be guaranteed to admit feature vectors that reveal the subspace membership of points in the ensemble.

### 7.1.2 Complete Reference Set

In order to guarantee EFS, we must ensure that we obtain a sufficient sampling of each subspace when forming a sparse decomposition of the dataset. In particular, the reference set $D$ must contain at least $k$ linearly independent vectors from each $k$-dimensional subspace in $U$—in this case, we say that $D$ provides a complete reference
set for the union $\mathcal{U}$. To make this precise, let $X$ denote the data matrix and $D \in \mathbb{R}^{M \times L}$ be a reference set containing columns of $X$ after being normalized. We say that a matrix $D$ provides a complete reference set for a subspace $\mathcal{S}$ if the following property holds.

**Definition 6 (Complete Reference Set)** Let $T$ index the set of columns in $D$ that lie in the span of a $k$-dimensional subspace $\mathcal{S} \subset \mathbb{R}^d$. The reference set $D$ provides a complete reference set for $\mathcal{S}$ if the rank($D_T$) = $k$.

**Remark.** When the data lives on a single subspace of dimension $k$ and our reference set $D$ is of the same rank, then the requirement that $D$ be a complete reference set is equivalent to the requirement needed to guarantee exact matrix factorization. In Thm. 8, we provide a guarantee that oASIS returns a reference set that satisfies this condition for low rank matrices.

**Remark.** When the data lie on multiple subspaces, to guarantee a complete reference set for the union, we must sample at least $\sum_{i=1}^{p} k_i$ points from $X$, where $k_i$ is the dimension of the $i^{th}$ subspace. In contrast, to guarantee exact matrix recovery, we only require rank($X$) linearly independent points, where

$$\text{rank}(X) = \text{dim}(\mathcal{U}) \leq \sum_{i=1}^{p} k_i,$$

and equality is attained when the subspaces are all independent. This implies that covering each subspace as in Def. 6 requires a potentially larger sampling of the data than simply covering the entire space spanned by $X$. Although it might appear disadvantageous to select more points than are needed to achieve exact recovery, in practice we observe that by selecting more columns than are needed for exact recovery, this leads to even sparser self-expressive representations of the data. In turn, we
obtain better cut ratios and clustering of the data when we slightly oversample the dataset.

### 7.1.3 Sufficient Conditions for EFS

In this Section, we develop a geometric condition to guarantee EFS for SEED. The results in this Section are based upon the analysis of SSC with OMP provided in [15].

**Theorem 3 (Exact Feature Selection)** Let $D$ be a complete reference set for subspace $S$. Let $\Lambda$ index the columns of $X$ that lie in the span of $S$ and $r(D, S)$ denote the inradius of the columns of $D$ along $S$. If the following condition holds

$$\max_{j \notin \Lambda} |\langle d_j, \pi_S(d_j) \rangle| < r(D, S),$$

(7.1)

then EFS will occur for all columns in $X_\Lambda$.

We make a few remarks below and provide a proof of this Theorem in Sec. 7.1.7.

**Remark.** Thm. 3 states that as long as the columns of $X$ that lie outside of $S$ are sufficiently incoherent with $S$ then EFS is guaranteed. The distance that points outside of $S$ must lie to ensure EFS occurs is governed by the inradius of $D$ along $S$, i.e., as long as there is not a large gap in the covering of $S$ with points in $D$, then EFS will occur. A geometric interpretation of Thm. 3 is that as long as the orthogonal projection of a signal outside of $S$ onto the subspace lies within the convex hull of the points within $S$, then EFS is guaranteed. See the analysis of SSC-OMP [15] for a discussion of EFS with OMP and further insight into the geometry underlying EFS from data living on unions of subspaces.
Remark. The sequential incoherence selection criterion used to select columns with oASIS provides a natural strategy for reducing the size of the reference set $D$ while also maintaining a good covering radius (minimizing gaps in each subspace in the union). This is due to the fact that the maximum inradius is achieved when points are maximally incoherent with one another, i.e., the set forms a tight frame for the subspace. The advantage of using oASIS is that we naturally prune away highly coherent columns of $X$, while maintaining a good covering of subspaces in the dataset.

Remark. If we consider all of the columns of $D$ that are not in $S$ to be disjoint one-dimensional subspaces (separated from $S$ by some angle), then the term on the left in Thm. 3 is equivalent to the cosine of the minimum principal angle between $S$ and all of the one-dimensional subspaces formed by the columns of $D_{T^c}$. The minimum principal angle between subspaces $S_i$ and $S_j$, is the smallest angle between a pair of unit vectors $(u_1, v_1)$ drawn from $S_i \times S_j$. A discussion of the minimum and higher-order principal angles between subspaces is provided in [15].

We can also develop a sufficient condition for EFS based upon the mutual coherence between points living in different subspaces.

**Theorem 4** Assume that $D$ is a complete reference set for subspace $S$. Let $\epsilon$ denote the covering diameter of the columns of $D$ along $S$. If the mutual coherence

$$\mu(D_{T^c}, D_{T^c}) < \sqrt{1 - \frac{\epsilon^2}{4}} - \frac{\epsilon}{\sqrt{12}}$$

then EFS will occur for all columns of $X$ that lie in $S$.

Since the columns of $D$ are selected from the dataset, this also implies that as long as the mutual coherence $\mu(X_\Lambda, X_{\Lambda^c})$ satisfies this condition, EFS is guaranteed for all the signals in $S$ in the submatrix $X_\Lambda$. 
Figure 7.1: *Geometry underlying EFS.* A union of two disjoint subspaces of different dimension: the convex hull of a set of points (red circles) living on a 2D subspace is shaded (green). In (a), we show an example where EFS is guaranteed—the projection of points along the 1D subspace lie inside the shaded region. In (b), we show an example where EFS is not guaranteed—the projection of points along the 1D subspace lie outside the shaded region.

This result implies that EFS can be guaranteed for intersecting subspaces as long as the points in distinct subspace clusters are bounded away from intersections between subspaces. When the covering radius shrinks to zero, Thm. 3 requires that $\mu_c < 1$, or that points from different subspaces do not lie exactly in the subspace intersection, i.e., are identifiable from one another.

**EFS for Disjoint Subspaces**

When the subspaces in the ensemble are *disjoint*, i.e., $\cos(\theta_{ij}^*) < 1$, Thm. 3 can be simplified further by using the bound for the mutual coherence in (2.3). This simplification results in the following corollary.

**Corollary 2** Let $\theta_{ij}^*$ denote the first principal angle between a pair of disjoint subspaces $S_i$ and $S_j$, and let $\epsilon$ denote the maximal covering diameter of the points in $X_i$. A sufficient condition for OMP to return exact feature sets for all points in $X_i$ is that
\[
\max_{j \neq i} \cos(\theta_{ij}^*) < \frac{\sqrt{1 - \epsilon^2/4}}{1 + \epsilon/\sqrt{12}}.
\]

7.1.4 Geometry Underlying EFS

The main idea underlying the proof of Thm. 3 is that, at each iteration of Alg. 1, we require that the residual used to select a point to be included in the feature set is closer to a point in the **correct subspace cluster** \(\mathcal{X}_i\) than a point in an **incorrect subspace cluster** \(\mathcal{X}_{-i}\). To be precise, we require that the normalized inner product of the residual signal \(s\) and all points outside of the correct subspace cluster

\[
\max_{y \in \mathcal{X}_{-i}} \frac{|\langle s, y \rangle|}{\|s\|_2} < r(\mathcal{X}_i),
\]

at each iteration of Alg. 1. To provide the result in Thm. 3, we require that (7.8) holds for all \(s \in S_i\), or all possible residual vectors.

A geometric interpretation of the EFS condition in Thm. 3 is that the orthogonal projection of all points outside of a subspace must lie within the antipodal convex hull of the set of normalized points that span the subspace. To see this, consider the projection of the points in \(\mathcal{X}_{-i}\) onto \(S_i\). Let \(z_j^*\) denote the point on subspace \(S_i\) that is closest to the signal \(y_j \in \mathcal{X}_{-i}\),

\[
z_j^* = \arg \min_{z \in \mathcal{S}_i} \|z - y_j\|_2.
\]

We can also write this projection in terms of a orthogonal projection operator \(P_i = D_iD_i^T\), where \(D_i\) is an ONB that spans \(S_i\) and \(z_j^* = P_i y_j\).

By definition, the normalized inner product of the residual with points in incorrect
subspace clusters is upper bounded as

$$\max_{y_j \in \mathcal{X}_-} \frac{|\langle s, y_j \rangle|}{\|s\|_2} \leq \max_{y_j \in \mathcal{X}_-} \frac{|\langle z_j^*, y_j \rangle|}{\|z_j^*\|_2} = \max_{y_j \in \mathcal{X}_-} \cos \angle \{z_j^*, y_j\}$$

Thus to guarantee EFS, we require that the cosine of the angle between all signals in $\mathcal{X}_-$ and their projection onto $\mathcal{S}_i$ is less than the inradius of $\mathcal{X}_i$. Said another way, the EFS condition requires that the length of all projected points be less than the inradius of $\mathcal{X}_i$.

In Fig. 7.1, we provide a geometric visualization of the EFS condition for a union of disjoint subspaces (union of a 1D subspace with a 2D subspace). In (a), we show an example where EFS is guaranteed because the projection of the points outside of the 2D subspace lie well within the antipodal convex hull of the points along the normalized 2D subspace (ring). In (b), we show an example where EFS is not guaranteed because the projection of the points outside of the 2D subspace lie outside of the antipodal convex hull of the points along the normalized 2D subspace (ring).

### 7.1.5 Connections to Previous Work

In this Section, we will connect our results for OMP with previous analyses of EFS with BP for disjoint [55, 14] and intersecting [56] subspaces. Following this, we will contrast the geometry underlying EFS with exact recovery conditions used to guarantee support recovery for both OMP and BP [38, 80].
Subspace Clustering with BP

[55] develops the following sufficient condition for EFS to occur for BP from a union of disjoint subspaces,

\[
\max_{j \neq i} \cos(\theta_{ij}^*) < \max_{\tilde{X}_i \in \mathcal{W}_i} \frac{\sigma_{\min}(\tilde{X}_i)}{\sqrt{k_i}}, \tag{7.4}
\]

where \( \mathcal{W}_i \) is the set of all full rank sub-matrices \( \tilde{Y}_i \in \mathbb{R}^{n \times k_i} \) of the data matrix \( \tilde{X}_i \in \mathbb{R}^{n \times k_i} \) and \( \sigma_{\min}(\tilde{X}_i) \) is the minimum singular value of the sub-matrix \( \tilde{X}_i \). Since we assume that all of the data points have been normalized, \( \sigma_{\min}(\tilde{X}_i) \leq 1 \); thus, the best case result that can be obtained is that the minimum principal angle, \( \cos(\theta_{ij}^*) < 1/\sqrt{k_i} \). This suggests that the minimum principal angle of the union must go to zero, i.e., the union must consist of orthogonal subspaces, as the subspace dimension increases.

In contrast to the condition in (7.4), the conditions we provide in Thm. 3 and Cor. 2 do not depend on the subspace dimension. Rather, we require that there are enough points in each subspace to achieve a sufficiently small covering; in which case, EFS can be guaranteed for subspaces of any dimension.

[56] develops the following sufficient condition for EFS to occur for BP from a union of intersecting subspaces,

\[
\mu_v(\mathcal{X}_i) = \max_{y \in \mathcal{X}_{\sim i}} \| V_{(i)} y \|_\infty < r(\mathcal{X}_i), \tag{7.5}
\]

where the matrix \( V_{(i)} \in \mathbb{R}^{d_i \times n} \) contains the dual directions (the dual vectors for each point in \( \mathcal{X}_i \) embedded in \( \mathbb{R}^n \)) in its columns,\(^1\) and \( r(\mathcal{X}_i) \) is the inradius as defined in (2.4). In words, (7.5) requires that the maximum coherence between any point in \( \mathcal{X}_{\sim i} \)

\(^1\)See Def. 2.2 for a formal definition of the dual directions and insight into the geometry underlying their guarantees for EFS via BP [56].
and the dual directions contained in \( V_{(i)} \) be less than the inradius of the points in \( X_i \).

To link the result in (7.5) to our guarantee for OMP in Thm. 3, we observe that while (7.5) requires that \( \mu_v(X_i) \) (coherence between a point in a subspace cluster and the dual directions of points in a different subspace cluster) be less than the inradius, Thm. 3 requires that the mutual coherence \( \mu_c(X_i) \) (coherence between two points in different subspace clusters) be less than the inradius minus an additional term that depends on the covering radius. For an arbitrary set of points that live on a union of subspaces, the precise relationship between the two coherence parameters \( \mu_c(X_i) \) and \( \mu_v(X_i) \) is not straightforward; however, when the points in each subspace cluster are distributed uniformly and at random along each subspace, the dual directions will also be distributed uniformly along each subspace.\(^1\) In this case, \( \mu_v(X_i) \) will be roughly equivalent to the mutual coherence \( \mu_c(X_i) \).

This simplification reveals the connection between the result in (7.5) for BP and the condition in Thm. 3 for OMP. In particular, when \( \mu_v(X_i) \approx \mu_c(X_i) \), our result for OMP requires that the mutual coherence is smaller than the inradius minus an additional term that is linear in the covering diameter \( \epsilon \). For this reason, our result in Thm. 3 is more restrictive than the result provided in (7.5). The gap between the two bounds shrinks to zero only when the minimum principal angle \( \theta_{ij}^* \to \pi/2 \) (orthogonal subspaces) or when the covering diameter \( \epsilon \to 0 \).

In our empirical studies, we find that when BPDN is tuned to an appropriate value of the noise parameter \( \kappa \), BPDN tends to produce higher rates of EFS than OMP. This suggests that the theoretical gap between the two approaches might not be an artifact of our current analysis; rather, there might exist an intrinsic gap between the performance of each method with respect to EFS. Nonetheless, an interesting finding from our empirical study in Sec. 6.5.1, is that despite the fact that BPDN provides

\(^1\)This approximation is based upon personal correspondence with M. Soltankotabi, an author of the work in [56].
better rates of EFS than OMP, OMP typically provides better clustering results than BPDN. For these reasons, we maintain that OMP offers a powerful low-complexity alternative to $\ell_1$-minimization approaches for feature selection.

**Exact Recovery Conditions for Sparse Recovery**

To provide further intuition about EFS in endogenous sparse recovery, we will compare the geometry underlying the EFS condition with the geometry of the exact recovery condition (ERC) for sparse signal recovery methods [38, 80] discussed in Sec. 3.6.

Recall, that a geometric interpretation of the ERC is that it provides a measure of how far a projected atom $\varphi_i$ outside of the set $\Lambda$ lies from the antipodal convex hull of the atoms in $\Lambda$. When a projected atom lies outside of the antipodal convex hull formed by the set of points in the sub-dictionary $D_\Lambda$, then the ERC condition is violated and support recovery is not guaranteed. For this reason, the ERC requires that the maximum coherence between the atoms in $D$ is sufficiently low or that $D$ is incoherent.

While the ERC condition requires a *global incoherence* property on all of the columns of $D$, we can interpret EFS as requiring a *local incoherence* property. In particular, the EFS condition requires that the projection of atoms in an incorrect subspace cluster $X_{-i}$ onto $S_i$ must be incoherent with any deep holes in $X_i$ along $S_i$. In contrast, we require that the points within a subspace cluster exhibit local coherence in order to produce a small covering radius.

### 7.1.6 EFS for Bounded Unions of Subspaces

In this Section, we study the connection between EFS and the higher-order principal angles (beyond the minimum angle) between pairs of intersecting subspaces.
The sufficient conditions for EFS in Thm. 3 and Cor. 2 reveal an interesting relationship between the covering radius, mutual coherence, and the minimum principal angle between pairs of subspaces in the ensemble. However, we have yet to reveal any dependence between EFS and higher-order principal angles. To make this connection more apparent, we will make additional assumptions about the distribution of points in the ensemble, namely that the dataset produces a *bounded union of subspaces* relative to the principal vectors supporting pairs of subspaces in the ensemble.

Let \( \mathbf{X} = [\mathbf{X}_i \mathbf{X}_j] \) denote a collection of unit-norm data points, where \( \mathbf{X}_i \) and \( \mathbf{X}_j \) contain the points in subspaces \( \mathcal{S}_i \) and \( \mathcal{S}_j \), respectively. Let \( \mathbf{G} = \mathbf{\Phi}_i^T \mathbf{\Phi}_j = \mathbf{U} \Sigma \mathbf{V}^T \) denote the SVD of \( \mathbf{G} \), where \( \mathbf{\Phi}_i \) is an ONB that spans \( \mathcal{S}_i \), and \( \text{rank}(\mathbf{G}) = q \). Let \( \tilde{\mathbf{U}} = \mathbf{\Phi}_i \mathbf{U}_q \) denote the set of left principal vectors of \( \mathbf{G} \) that are associated with the \( q \) nonzero singular values in \( \Sigma \). Similarly, let \( \tilde{\mathbf{V}} = \mathbf{\Phi}_j \mathbf{V}_q \) denote the set of right principal vectors of \( \mathbf{G} \) that are associated with the nonzero singular values in \( \Sigma \). When the points in each subspace are incoherent with the principal vectors in the columns of \( \tilde{\mathbf{U}} \) and \( \tilde{\mathbf{V}} \), we say that the ensemble \( \mathbf{X} \) is a *bounded union of subspaces*. Formally, we require the following incoherence property holds:

\[
\left( \| \mathbf{X}_i^T \tilde{\mathbf{U}} \|_{\infty}, \| \mathbf{X}_j^T \tilde{\mathbf{V}} \|_{\infty} \right) \leq \gamma, \tag{7.6}
\]

where \( \| \cdot \|_{\infty} \) is the entry-wise maximum and \( \gamma \in (0, 1] \). This property requires that the inner products between the points in a subspace and the set of principal vectors that span non-orthogonal directions between a pair of subspaces is bounded by a fixed constant.

When the points in each subspace are distributed such that (7.6) holds, we can rewrite the mutual coherence between any two points from different subspaces to reveal its dependence on higher-order principal angles. In particular, we show (in Sec. 7.1.7) that the coherence between the residual \( \mathbf{r} \) used in Alg. 1 to select the next
point to be included in the representation of a signal in \( S_i \) and a point in \( S_j \) is upper bounded by

\[
\max_{x \in S_j} \frac{|\langle r, x \rangle|}{\|r\|_2} \leq \gamma \|\sigma_{ij}\|_1, \tag{7.7}
\]

where \( \gamma \) is the bounding constant of the data \( Y \) and \( \|\sigma_{ij}\|_1 \) is the \( \ell_1 \)-norm of the cross-spectra or equivalently, the trace norm of \( G \). Using the bound in (7.7), we arrive at the following sufficient condition for EFS from bounded unions of subspaces. We provide the proof in Sec. 7.1.7.

**Theorem 5** Let \( Y \) live on a bounded union of subspaces, where \( q = \text{rank}(G) \) and \( \gamma < \sqrt{1/q} \). Let \( \sigma_{ij} \) denote the cross-spectra of the subspaces \( S_i \) and \( S_j \) and let \( \epsilon \) denote the covering diameter of the signals in \( X \) living in \( S_i \). A sufficient condition for Alg. 1 to return exact feature sets for all points in \( S_i \) is that the covering diameter

\[
\epsilon < \min_{j \neq i} \sqrt{1 - \gamma^2 \|\sigma_{ij}\|_2^2}.
\]

This condition requires that both the covering diameter of each subspace and the bounding constant of the union be sufficiently small in order to guarantee EFS. One way to guarantee that the ensemble has a small bounding constant is to constrain the total amount of energy that points in \( X_j \) have in the \( q \)-dimensional subspace spanned by the principal vectors in \( \tilde{V} \).

Our analysis for bounded unions assumes that the nonzero entries of the cross-spectra are equal, and thus each pair of supporting principal vectors in \( \tilde{V} \) are equally important in determining whether points in \( X_i \) will admit EFS. However, this assumption is not true in general. When the union is supported by principal vectors with non-uniform principal angles, our analysis suggests that a weaker form of incoherence is required. Instead of requiring incoherence with all principal vectors, the data must be sufficiently incoherent with the principal vectors that correspond to small princi-
pal angles (or large values of the cross-spectra). This means that as long as points are not concentrated along the principal directions with small principal angles (i.e., intersections), then EFS can be guaranteed, even when subspaces exhibit non-trivial intersections. To test this prediction, we will study EFS for a bounded energy model in Sec. 6.5.1. We show that when the dataset is sparsely sampled (larger covering radius), reducing the amount of energy that points contain in subspace intersections, does in fact increase the probability that points admit EFS.

Finally, our analysis of bounded unions suggests that the decay of the cross-spectra is likely to play an important role in determining whether points will admit EFS or not. To test this hypothesis, we will study the role that the structure of the cross-spectra plays in EFS in Sec. 6.5.1.

### 7.1.7 Proofs

**Proof of Thm. 3**

The main idea underlying the proof of Thm. 3 is that at each iteration of OMP (Alg. 1), the residual that is used to select a column of $D$ to be included in the representation, is closer to a signal in the *correct subspace cluster* (indexed by the set $T$) than a signal in an *incorrect subspace cluster* (outside of the set $T$). To be precise, we require that the normalized inner product of the residual signal $s$ and all signals outside of the correct subspace cluster

$$\max_{i \notin T} \frac{|\langle s, d_i \rangle|}{\|s\|_2} < \max_{i \in T} \frac{|\langle s, d_i \rangle|}{\|s\|_2},$$

at each iteration of Alg. 1, where $d_i$ is the $i^{th}$ column of the reference set $D$. As long as we can guarantee that (7.8) holds at each iteration of OMP, then this is sufficient to guarantee that EFS occurs for every point in $X$ in $S$. We will prove this by induction.
Consider the greedy selection step in OMP (see Alg. 1) for a signal $x$ which lies in $S$. Recall that at the $m^{th}$ step of OMP, the column of $D$ that is maximally correlated with the signal residual will be selected to be included in the representation. Let $\Lambda$ index the set of columns from $D$ already selected. The residual at the $m^{th}$ step is computed as

$$s^m = (I - P_\Lambda)x,$$

where $P_\Lambda = D_\Lambda D_\Lambda^T$ is a projector onto the subspace spanned by the points in the current feature set $\Lambda$, where $|\Lambda| = m - 1$. Thus, as long as $\Lambda \subset T$, this guarantees that the residual $s \in S$.

To prove that the selection criterion in (7.8) holds at each step of OMP, our aim is to develop a lower bound on the LHS (the maximum inner product between the residual and a point outside of $S$) and an upper bound on the RHS (the minimum inner product between the residual and a column of $D_T$).

Since we require that (7.8) holds for all possible $s \in S$, or all possible residual vectors that lie in the span of $S$, the RHS is bounded by the inradius as follows,

$$\max_{i \in T} |\langle s, d_i \rangle| \|s\|_2 \geq r(D_T, S),$$

where the inradius provides a measure of the worst-case coherence between any residual that lies in $S$ and a point in $D_T$.

Now, our goal is to develop an upper bound the term on the left. To do this, consider the orthogonal projection of all columns of $D$ not in $S$ (indexed by $T^c$) onto $S$ and find the signal with maximum inner product (closest to $S$). Thus, we have the following upper bound in effect

$$\max_{i \notin T} |\langle s, d_i \rangle| \|s\|_2 \leq \max_{i \notin T} |\langle \pi_S(d_i), d_i \rangle|.$$
Putting our upper and lower bound together, we arrive at our final sufficient condition
\[
\max_{i \in T} |\langle \pi_S(d_i), d_i \rangle| < r(D_T, \mathcal{S}).
\]
Since we have shown that this condition is sufficient to guarantee EFS at each step of Alg. 1 provided the residual stays in the correct subspace, Thm. 3 follows by induction. \hfill \square

**Proof of Thm. 4**

We now simplify Thm. 3 to provide a bound for EFS based upon the mutual coherence between signals living on a subspace \(\mathcal{S}\) and all other signals in the dataset. Using the analysis in [15], we can develop the following bound on the LHS (maximum inner product between the residual and a point outside of \(\mathcal{S}\)).

Just as before, we assume that the signal residual \(s\) remains in \(\mathcal{S}\); we can write the normalized residual \(s^m = z + e\), where \(z\) is the atom in \(D_T\) closest to \(s^m\) and \(e\) is the remaining portion of the residual which also lies in \(\mathcal{S}\). Thus, we can rewrite and bound the LHS of the selection criterion in (7.8) as

\[
\max_{i \in T} |\langle s^m, d_i \rangle| = \max_{i \in T} |\langle z + e, d_i \rangle|
\]
\[
\leq \max_{i \in T} |\langle z, d_i \rangle| + |\langle e, d_i \rangle|
\]
\[
\leq \mu(D_T, D_T^C) + \max_{i \in T} |\langle e, d_i \rangle|
\]
\[
\leq \mu(D_T, D_T^C) + \|e\|_2 \|d_i\|_2,
\]

where \(\theta_0\) is the minimum principal angle between \(\mathcal{S}\) and all other subspaces in the ensemble.

Using the fact that the covering radius of the set of columns \(D_T\) along \(\mathcal{S}\) equals
\( \epsilon/2 \), we can bound the \( \ell_2 \)-norm of the vector \( e \) as

\[
\|e\|_2 = \|s - z\|_2 \\
= \sqrt{\|s\|_2^2 + \|z\|_2^2 - 2|\langle s, z \rangle|} \\
\leq \sqrt{2 - 2\sqrt{1 - (\epsilon/2)^2}} \\
= \sqrt{2 - \sqrt{4 - \epsilon^2}}.
\]

Plugging this quantity into our expression for the LHS, we arrive at the following upper bound

\[
\max_{i \notin T} |\langle s^m, d_i \rangle| \leq \mu_c(D_T, D_{T^c}) + \sqrt{2 - \sqrt{4 - \epsilon^2}} < \mu_c(D_T, D_{T^c}) + \frac{\epsilon}{\sqrt{12}},
\]

where the final simplification comes from invoking the following Lemma.

**Lemma 2** For \( 0 \leq x \leq 1 \),

\[
\sqrt{2 - \sqrt{4 - x^2}} \leq \frac{x}{\sqrt{12}}.
\]

**Proof of Lemma 1**: We wish to develop an upper bound on the function

\[
f(x) = 2 - \sqrt{4 - x^2}, \quad \text{for } 0 \leq x \leq 1.
\]

Thus our goal is to identify a function \( g(x) \), where \( f'(x) \leq g'(x) \) for \( 0 \leq x \leq 1 \), and \( g(0) = f(0) \). The derivative of \( f(x) \) can be upper bounded easily as follows

\[
f'(x) = \frac{x}{\sqrt{4 - x^2}} \leq \frac{x}{\sqrt{3}}, \quad \text{for } 0 \leq x \leq 1.
\]

Thus, \( g'(x) = x/\sqrt{3} \), and \( g(x) = x^2/\sqrt{12} \); this ensures that \( f'(x) \leq g'(x) \) for \( 0 \leq x \leq 1 \).
1, and \( g(0) = f(0) \). By the Fundamental Theorem of Integral Calculus, \( g(x) \) provides an upper bound for \( f(x) \) over the domain of interest where, \( 0 \leq x \leq 1 \). To obtain the final result, take the square root of both sides, \( \sqrt{2 - \sqrt{4 - x^2}} \leq \sqrt{x^2/\sqrt{12}} = x/\sqrt{12} \).

\[ \square \]

Now, finally we use this bound to simplify Thm. 3 and relate the mutual coherence between signals in \( \mathbf{D} \) that lie in different subspaces and the relationship between the inradius and covering of the reference set \( \mathbf{D} \) along \( \mathcal{S} \).

\[
\mu_c(\mathbf{D}_T, \mathbf{D}_{T^c}) < \sqrt{1 - \epsilon^2/4} - \frac{\epsilon}{\sqrt{12}}.
\]

This completes our proof. \( \square \)

**Proof of Thm. 5**

To prove Thm. 5, we will assume that the union of subspaces is bounded in accordance with (7.6). This assumption enables us to develop a tighter upper bound on the mutual coherence between any residual signal \( \mathbf{s} \in \mathcal{S}_i \) and the points in \( \mathbf{X} \) that lie in \( \mathcal{S}_j \) which we denote by the set \( \mathcal{X}_j \). Since \( \mathbf{s} \in \mathcal{S}_i \), the residual can be expressed as \( \mathbf{s} = \Phi_i \alpha \), where \( \Phi_i \in \mathbb{R}^{n \times k_i} \) is an ONB that spans \( \mathcal{S}_i \) and \( \alpha = \Phi_i^T \mathbf{s} \). Similarly, we can write each signal in \( \mathcal{X}_j \) as \( \mathbf{x} = \Phi_j \beta \), where \( \Phi_j \in \mathbb{R}^{n \times k_j} \) is an ONB that spans \( \mathcal{S}_j \) and \( \beta = \Phi_j^T \mathbf{y} \). Let \( \mathcal{B}_j = \{ \Phi_j^T \mathbf{x}_i \}_{i=1}^{d_j} \) denote the set of all subspace coefficients for all \( \mathbf{x}_i \in \mathcal{X}_j \).

The coherence between the residual and a point in a different subspace can be
expanded as follows:

\[
\max_{x \in X_j} \frac{|\langle s, x \rangle|}{\|s\|_2} = \max_{\beta \in B_j} \frac{|\langle \Phi_j \alpha, \Phi_j \beta \rangle|}{\|\alpha\|_2} \\
= \max_{\beta \in B_j} \frac{|\langle \alpha, \Phi_j^T \Phi_j \beta \rangle|}{\|\alpha\|_2} \\
= \max_{\beta \in B_j} \frac{|\langle \alpha, U \Sigma V^T \beta \rangle|}{\|\alpha\|_2} \\
= \max_{\beta \in B_j} \frac{|\langle U^T \alpha, \Sigma V^T \beta \rangle|}{\|\alpha\|_2} \\
\leq \max_{\beta \in B_j} \frac{\|U^T \alpha\|_\infty}{\|\alpha\|_2} \|\Sigma V^T \beta\|_1,
\]

(7.9)

where the last step comes from an application of Holder’s inequality, i.e., \(|\langle w, z \rangle| < \|w\|_\infty \|z\|_1\).

Now, we tackle the final term in (7.9), which we can write as

\[
\max_{\beta \in B_j} \|\Sigma V^T \beta\|_1 = \max_{x \in X_j} \|\Sigma V^T \Phi_j^T x\|_1 = \max_{x \in X_j} \|\Sigma (\Phi_j V)^T x\|_1,
\]

where the matrix \(\Phi_j V\) contains the principal vectors in subspace \(S_j\). Thus, this term is simply a sum of weighted inner products between the principal vectors \(\Phi_j V\) and all of the points in \(S_j\), where \(\Sigma\) contains the cross-spectra in its diagonal entries.

Since we have assumed that the union is bounded, this implies that the inner product between the first \(q\) principal vectors and the points in \(X_j\) are bounded by \(\gamma\), where \(q = \|\sigma_{ij}\|_0 = \text{rank}(G)\). Let \(\Phi_j V_q \in \mathbb{R}^{n \times q}\) be the first \(q\) singular vectors of \(G\) corresponding to the nonzero singular values in \(\Sigma\) and let \(\Sigma_q \in \mathbb{R}^{q \times q}\) be a diagonal matrix with the first \(q\) nonzero singular values of \(G\) along its diagonal. It follows that

\[
\|\Sigma (\Phi_j V)^T y\|_\infty = \|\Sigma_q (\Phi_j V_q)^T y\|_\infty \leq \gamma.
\]

Now, suppose that the bounding constant
$\gamma < \sqrt{1/q}$. In this case,

$$\max_{y \in \lambda_j} \| \Sigma (\Phi_j V)^T x \|_1 \leq \gamma \| \sigma_{ij} \|_1.$$ 

Note that for bounded unions of subspaces, the term on the right can be made small by requiring that the bounding constant $\gamma \ll 1$. Plugging this bound into (7.9), we obtain the following expression

$$\max_{x \in \lambda_j} \frac{|\langle s, x \rangle|}{\|s\|_2} \leq \gamma \| \sigma_{ij} \|_1 \frac{\| U^T \alpha \|_\infty}{\| \alpha \|_2} = \gamma \| \sigma_{ij} \|_1 \| U \|_{2,2} = \gamma \| \sigma_{ij} \|_1,$$

where this last simplification comes from the fact that $U$ is unitary and has spectral norm equal to one. Note that this bound on the mutual coherence is informative only when $\gamma \| \sigma_{ij} \|_1 < \sigma_{\text{max}} \leq 1$. This completes the proof. \hfill \Box

### 7.2 Rank Revealing Property of SEED

An interesting property of sparse self-expressive decomposition is that the sparsity of each column of $V$ is a strong indicator of the underlying geometry of the data and reference set. When the dataset lies on a union of subspaces, the sparse representations obtained via self-expression are **rank revealing**. By rank revealing we mean that the number of non zeros (sparsity) in each column of $V$ is bounded by the dimension of the subspace the signal lies on. This enables insight into the subspace dimension of signals based upon their self-expressive representations. In contrast, outlier points that are incoherent from other points in the dataset admit dense representations. Thus, we can utilize the sparsity level of each column of $V$ to develop a guarantee for outlier detection in Thm. 7.
7.2.1 Lower Bound on Sparsity Level

We now introduce a geometric property of the data that provides a lower bound on the $\ell_0$-norm of solutions obtained via OMP. The size of the minimal dependent set (MDS) is the smallest submatrix in $\Omega$ such that adding $x$ to the submatrix will make it rank deficient. We now supply a formal definition.

Definition 7 Let $\Omega = \{D_{T_1}, \ldots, D_{T_Q}\}$ denote the set of all full rank submatrices of $D$. The size of the minimal dependent set (MDS) for the pair $(x, D)$ is defined as

$$s(x, D) = \min_i |T_i| \quad \text{subject to} \quad \text{rank}([D_{T_i} \ x]) = |T_i|, \ D_{T_i} \in \Omega$$

In words, the MDS $s(x, D)$ measures the minimum number of columns of $D$ required to form an exact representation of $x$.

Using the notion of a MDS, the following proposition is in effect.

Proposition 1 Let $v^*$ denote the sparse representation of a signal $x$ with respect to $D$ returned by OMP, where $\epsilon = 0$. The $\ell_0$-norm of $v^*$, $\|v^*\|_0 \geq s(x, D)$.

Proof. Recall that the sparse approximation problem (P0) seeks the sparsest solution that satisfies an exact equality constraint, or equivalently the solution of (L0-Error) with $\epsilon = 0$. The sparsest representation of $x$ with respect to $D$ is that which requires the smallest number of columns of $D$ and thus, the $\ell_0$-norm of this solution coincides with the MDS. Therefore, the solution to (P0), which we denote by $\tilde{v}$, has $\ell_0$-norm equal to $s(x, D)$, i.e., $\|\tilde{v}\|_0 = s(x, D)$. Since OMP finds an approximate solution to (P0), the MDS provides a lower bound on the sparsity of the representation formed by OMP, i.e., $\|v^*\|_0 \geq s(x, D)$. This completes our proof. \qed
7.2.2 Upper Bound on Sparsity Level

When $D$ forms a complete reference set for a $k$-dimensional subspace $S$, this implies that there are at least $k$ linearly independent vectors that lie in the span of $S$. Thus, for all signals in $X$ that lie in $S$, indexed by $\Lambda$, the MDS

$$s(x_i, D) \leq k, \; \forall i \in \Lambda.$$ 

This means that all signals that lie in $S$ admit $k$-sparse representations.

When $D$ contains enough samples from $S$ to satisfy the EFS condition in Theo. 3, this also implies that the residual used to select columns from $D$ in OMP remains in $S$. Thus, a direct consequence of our result for EFS is that when EFS occurs for signals in a $k$-dimensional subspace, this also implies that no more than $k$ columns from $D$ are necessary to represent all of the signals in the subspace. The following theorem makes this idea precise.

**Theorem 6** Let $D$ be a complete reference set for subspace $S$ with $\dim(S) = k$ and assume that the EFS condition in Theo. 3 holds. The sparse representation of all columns of $X$ in $S$ returned by OMP will contain at most $k$ nonzeros.

The proof is contained in Sec. 7.2.4. This result states that when EFS occurs (all signals outside of $S$ are sufficiently incoherent with the set of columns of $D$ that lie in $S$), this also guarantees that we obtain a $k$-sparse representation of all signals in $X$ that lie in $S$. Thus, the sparsity level of these signals also reveals the subspace dimension of $S$.

7.2.3 Results for Outlier Detection

In contrast to the fact that the sparsity of signals in low-dimensional subspaces is upper bounded by their subspace dimension, when columns of $X$ are incoherent with
other points in the dataset, they tend to admit dense representations. When there is a sufficient gap between the sparsity level of inliers (signals on a low-dimensional subspace $\mathcal{U}$) and outliers, we can use the sparsity level to perform outlier detection. In the sequel, we will consider the case where the inliers live on a $k$-dimensional subspace and our goal is to separate incoherent outliers from the points living on this subspace.

While developing a bound for the MDS is challenging, simply forming a sparse representation with OMP provides a strong indicator of the MDS and the underlying geometry of signals in the dataset. For instance, in numerical experiments like those shown in Fig. 5.3.1, we find a significant gap between the sparsity level of signals living on low-dimensional subspaces and those signals that are incoherent from other signals in the dataset. The larger the gap, the easier it is to segment outliers from the rest of the data.

Predicting the precise gap between the MDS is difficult, however, we can guarantee that there is a gap provided that the outlier points contained in $\tilde{X}$ are sufficiently incoherent from one another and from all inliers, contained in the columns of $X_1$. In particular, we make two assumptions, detailed below as (A1) and (A2).

(A1) The outliers in $\tilde{X}$ satisfy the following mutual incoherence condition:

$$
\mu(X_I, \tilde{X}) < m - \frac{\epsilon}{\sqrt{12}},
$$

where $\epsilon$ is covering diameter of $D$ along $S$.

(A2) The outliers in $\tilde{X}$ satisfy the following incoherence condition:

$$
\mu(\tilde{X}) < m.
$$
Proposition 2 Provided that there exists a constant \( m \leq 1/k \) for which (A1) and (A2) are satisfied, we can guarantee that the MDS of the columns of \( \tilde{X} \) is greater than the subspace dimension \( k = \dim(S) \).

The proof of this proposition is contained in Sec. 7.2.4.

This proposition guarantees that there will be a gap between the sparsity level of outlier and inlier points. Thus, we can develop the following result for outlier detection with Alg. 4.

Theorem 7 [Outlier Detection] Assume that the EFS condition in Thm. 3 is satisfied and that (A1) and (A2) are satisfied. The outlier detection method in Alg. 4 is guaranteed to reject all outlier points in \( \tilde{X} \) provided that the sparsity threshold \( \tau \leq s(x_i, D) \). Moreover, Alg. 4 will not reject any inlier points in \( X_1 \) provided that \( \tau > k \).

This result states that when we have a sufficient sampling of signals in subspaces (complete reference set) such that our EFS criterion is satisfied, then the sparsity of signals in the subspace is bounded by the subspace dimension. Thus, as long as the MDS for outlier points exceeds the maximum subspace dimension, then we will observe a gap between the sparseness of inlier and outlier points. In practice, we find that this gap becomes more pronounced as we sample more signals from each subspace (as the covering radius of \( D \) along each subspace shrinks).

7.2.4 Proofs

Proof of Thm. 6

Our goal is to prove that OMP yields a \( k \)-sparse representation of any signal that lies entirely in the span of \( S \), indexed by \( T \), where \( k \) is the dimension of \( S \). To do this, we must ensure that \( D \) provides a complete reference set for \( S \) and that at each
iteration of OMP we select a column from $D$ that lies in $S$. If we assume that the EFS condition holds, this guarantees that the residual used to select columns of $D$ to form a sparse representation of signals in $X_T$ will remain in $S$. A consequence of the EFS condition in Theo. 3 is that at each iteration of OMP, the residual signal used to select a new column of $D$ will remain in $S$. As long as the residual remains in $S$, any $k$ linearly independent points from $D$ that lie in $S$ will provide an exact representation of any signal $x$ that lies in $S$. A property of OMP is that we are guaranteed to select a column of the dictionary that is linearly independent from the atoms selected at previous iteration. This is easy to see based upon the orthogonalization step, where $r \in V^\perp$ and $V = \text{span}(D_A)$ is the subspace spanned by the current support set. Thus, this guarantees that the representation of each column of $X_T$ will be $k$-sparse, i.e., contain at most $k$ nonzeros. This completes our proof \( \Box \).

Proof of Prop. 2

Without loss of generality, we assume that the dataset $X$ is reordered such that all of the inliers $X_1$ and outliers $\tilde{X}$ are stacked together, where $X = [X_1 \ \tilde{X}]$. Let $S$ denote the $k$-dimensional subspace spanned by the inliers $X_1$ and $\epsilon$ denote the covering radius of the reference set $D$ along $S$.

Recall that Prop. 2 requires that there exists a constant $m < 1/k$ for which the following assumptions (A1) and (A2) are satisfied.

(A1) The outliers in $\tilde{X}$ satisfy the following mutual incoherence condition:

$$\mu(X_I, \tilde{X}) < m - \frac{\epsilon}{\sqrt{12}},$$

where $\epsilon$ is covering diameter of $D$ along $S$. 

The outliers in $\tilde{X}$ satisfy the following incoherence condition:

$$\mu(\tilde{X}) < m.$$ 

Our goal is to prove that if $m < 1/k$, then the MDS of the outliers in $\tilde{X}$ exceeds $k$ and thus, there is a gap in the sparsity level of outliers and inliers. Recall that the MDS of an outlier column $\{x_i\}_{i \in I}$ is equivalent to the minimum number of columns of $D$ that must be selected to exactly represent $x_i$. We will prove this in two steps.

First, we will guarantee that no outlier can be written as a linear combination of points from $S$. To do this, we must ensure that the projection

$$\max_{i \notin I} |\langle x_i, \pi_S(x_i) \rangle | < 1.$$ 

We can rewrite this term in terms of the mutual coherence between signals in $X_1$ and outliers in $\tilde{X}$. In particular, we can write the projection of $d_i$ onto $S$, $\pi_S(d_i) = z + e$ in terms of two components, where $z$ is the closest point in $D$ that lies on $S$ (inlier) and $e$ is the residual which also lies on $S$. In this case, we can expand and bound this term as follows.

$$|\langle d_i, \pi_S(d_i) \rangle | \leq |\langle d_i, z \rangle | + |\langle d_i, e \rangle | \leq \mu(\tilde{X}, X_1) + \|d_i\|_2\|e\|_2$$

$$\leq \mu(\tilde{X}, X_1) + \frac{\epsilon}{\sqrt{12}},$$

where the last step comes from the same argument provided in the proof of Thm. 4 in Sec. 7.1.7. Thus, finally to guarantee that no outlier can be perfectly represented in terms of signals from $X_1$, we must ensure that the mutual coherence between outliers
and inliers

$$\mu(\tilde{X}, X_1) < 1 - \frac{\epsilon}{\sqrt{12}}.$$  

Second, we will guarantee that no outlier can be exactly represented with $k$ other outliers, in which case, the MDS would equal $k$. To do this, we simply need to ensure that the spark of $\tilde{X}$ exceeds $k$. Thus, we invoke the coherence-based bound in Prop. 22 to bound the spark. Using this result, in order to guarantee that the spark($\tilde{X}$) > $k$, a sufficient condition is that

$$\mu(\tilde{X}) < 1/k.$$  

Finally, since we ensure that as long as (A1) and (A2) hold, this guarantees that no outlier admits a $k$-sparse representation and thus the MDS of these columns must exceed $k$. This completes our proof.

\[\square\]

### 7.3 Exact Matrix Recovery

In this Section, we develop conditions for exact matrix recovery and provide a complexity analysis for oASIS.

#### 7.3.1 Condition for Exact Matrix Recovery

We now examine situations where we obtain exact matrix recovery for a particular reference set $D$. Recall that exact recovery means $X = DD^+X$. This property holds whenever $D$ contains at least $r$ linearly independent columns from $X$, in which case all columns of $X$ lie in the range of $D$. Fortunately, oASIS provides a selection strategy that is guaranteed to select linearly independent columns before termination.

Based upon the properties of oASIS, we develop the following result for low rank matrix recovery. Before stating our result, we must make sure that the CSS algorithm is initialized with a set of linearly independent columns from $X$. Thus, we must ensure
that the number of columns used for initialization is less than the spark of \( \mathbf{X} \). We are now equipped to state our result for exact matrix recovery.

**Theorem 8** Let \( \mathbf{X} \) be a rank \( r \) matrix with \( \text{spark}(\mathbf{X}) = s \). Assume that Alg. 3 is initialized by randomly selecting fewer than \( s \) columns from \( \mathbf{X} \). If Algorithm 3 returns a set of \( r \) columns indexed by the set \( S \) before terminating with \( \Delta(i) = 0 \), then exact matrix recovery is guaranteed, i.e., \( \mathbf{X} = \pi_S(\mathbf{X}) \).

The proof is contained in Sec. 7.3.2.

Using the cumulative coherence, we can obtain a lower bound on the spark and thus the following corollary is in effect.

**Corollary 3** Let \( k^* \) equal the maximum integer such that \( \mu_{k^*}(\mathbf{G}) < 1 \). Assume that Alg. 3 is initialized by selecting less than \( k^* \) columns from \( \mathbf{X} \). If Alg. 3 returns a set of \( r \) columns indexed by the set \( S \) before terminating \( \Delta(i) = 0 \) then exact matrix recovery is guaranteed, i.e., \( \| \mathbf{X} - \pi_S(\mathbf{X}) \|_F = 0 \).

This corollary gives us an estimate of how many columns to use in the initialization of oASIS. In addition, the coherence of the matrix also provides insight into the gap between the performance of random versus adaptive sampling. When the columns of \( \mathbf{X} \) are incoherent, then random sampling performs similarly with oASIS, however, when certain subsets of columns of \( \mathbf{X} \) are coherent, then adaptive sampling outperforms random sampling.

The result above guarantees that oASIS will return a set of \( r \) linearly independent columns in \( r \) steps as long as the selection criterion \( \Delta(i) \neq 0 \) holds before exact reconstruction occurs. Unfortunately, in the pathological case in which the algorithm fails with \( \Delta(i) = 0 \) before \( r \) columns have been selected, the algorithm may terminate before \( r \) columns have been selected. While it is possible to construct pathological
matrices where this occurs, we have not observed this early termination in practice. In fact, the following theorem shows that for random matrices, the algorithm succeeds with probability 1.

**Theorem 9** Suppose $X$ is drawn from a continuous random distribution (and that this distribution has a density function). Assume that Alg. 3 is initialized by randomly selecting fewer than $r$ columns from $X$. Then oASIS succeeds in generating $r$ linearly independent columns with probability 1.

The proof is contained below in Sec. 7.3.3.

**Remark.** After sampling enough columns to span the range of $X$, $\Delta(i) = 0$ for all remaining columns. We thus terminate the algorithm when $\Delta$ is nearly zero (to machine precision). One advantage of oASIS is that this stopping criterion is extremely easy to compute, thus oASIS provides a very natural strategy for deciding how many columns to sample.

Interestingly, we observe that in many experiments on real-world and synthetic data, oASIS actually achieves the same approximation error decay rate as PCA (which also provides an exact representation of a matrix with a factorization of size $r$). In contrast, SES and other sampling methods are not capable of achieving this error rate.

### 7.3.2 Proof of Thm. 8

Our proof is based upon the following lemma about Alg. 3.

**Lemma 3 (Exact Recovery Condition)** Let $X$ be a rank $r$ matrix and $G_S$ be the set of columns selected from the corresponding Gram matrix via Alg. 3. Exact matrix recovery $X = \pi_S(X)$ occurs when $\text{rank}(G_S) = r$. 
**Proof.** Note that $G_S = X^T X_S$, and so $\text{rank}(G_S) = \text{rank}(X^T X_S) = \text{rank}(X_S)$. This implies that the subset of columns $X_S$ spans the entire range of $X$ and thus $X = \pi_S(X)$, i.e., exact matrix recovery is guaranteed.

In order to ensure that $\text{rank}(G_S) = r$, $G_S$ must consist of $r$ linearly independent points. Fortunately, the properties of oASIS guarantee that a linear independent column is chosen on each iteration. This statement is made precise by the following Lemma.

**Lemma 4** At each step of Alg. 3, $\Delta(i) > 0$ only if $x_i$ is linearly independent from the previously selected columns contained in $G_k$.

**Proof.** We prove this Lemma by induction. Consider $W_k$, the square matrix comprising the entries of $G$ at the selected row and column indices after $k$ columns have been selected. Assume for induction that $W_k$ is invertible. Now, consider $W_{k+1}$ given by

$$W_{k+1} = \begin{bmatrix} W_k & b_{k+1} \\ b_{k+1}^T & d_{k+1} \end{bmatrix}. \quad (7.13)$$

This matrix is invertible provided the Shur complement of $W_k$ is non-zero. But the Shur compliment is just $d_{k+1} - b_{k+1}^T W_k^{-1} b_{k+1} = \Delta_k$. Thus, if $\Delta_k$ is nonzero, then $W_{k+1}$ contains linear independent columns, and the corresponding columns of $G$ from which $W_{k+1}$ is drawn must also be linearly independent.

Lemma 4 implies that if we initialize oASIS with linearly independent columns, the algorithm will select the remaining columns to form a set of $r$ linearly independent columns of $G$, provided our termination condition does not equal zero. By Lemma 3, this guarantees that we obtain exact matrix recovery for the original data matrix $X$.

Note that the initialization columns will be independent as long as we choose fewer than $\text{spark}(X)$. Note that this is always satisfied when we select a single column from $X$. However, when the columns of $X$ exhibit an incoherence property, we can select
a larger subset of columns from $X$ and still obtain a set of $r$ linearly independent columns from $X$.

### 7.3.3 Proof of Thm. 9

We now prove that oASIS succeeds to sample $r$ columns from a rank $r$ dataset with probability 1, provided the dataset is drawn from a continuous random variable with density.

We begin by showing that the randomly chosen initialization columns of the Gram matrix $G$ have full rank. The matrix $G$ is a product of random matrices, and thus is itself a continuous random matrix with distribution. Note the set of singular matrices has positive co-dimension and thus measure 0. The probability of choosing initialization columns from this set is zero.

We now show that the probability of oASIS terminating with only $k < r$ columns is zero. Observe that the algorithm fails to choose the $k$th vector only if $\Delta_i = d_i - b_i^T W_{k-1}^{-1} b_i = 0$. Equivalently, the algorithm fails if and only if

$$d_i = b_i^T W_{k-1}^{-1} b_i, \quad \forall i \in \{1, 2, \cdots, N\}. \quad (7.14)$$

Note that, by construction, condition (7.14) holds for $i \in \{1, 2, \cdots, k - 1\}$ (the $k-1$ Nyström approximation used by oASIS on iteration $k$ perfectly represents these columns). However, there is an infinite set of matrices that share these $k-1$ columns, while having different values for $\{d_i\}_{i=k}^N$. In fact, as above, this set has positive co-dimension, and thus forms a set of measure zero. It follows that the probability of terminating on iteration $k < r$ is zero. $\square$. 


7.4 Complexity Analysis of SEED

One of the major advantages of SEED is its complexity and low memory footprint. The complexity of this method enables its use for extremely large datasets. Now, we analyze the complexity and discuss the advantages of this approach over other sparse matrix factorization methods.

7.4.1 Complexity of SEED

The rate-limiting step of Alg. 3 is the computation of $R_{k+1}$ by updating $R_k$. Eq. (5.4) allows this to be performed by sweeping over the entries of $R_k$, which has dimensions $k \times N$. The complexity of a single iteration is thus $O(kN)$. To sample $L$ columns, then $\sum_{k=1}^{L} kN = \frac{1}{2}L(L + 1)N$ entries must be updated. The resulting complexity of oASIS is $O(L^2N)$.

After sampling $L$ columns from $X$, the next step of SEED is to compute sparse representations of the remaining $N - L$ columns in terms of the selected set of columns (after normalization). Since we use the same dictionary to encode all of the remaining columns, we employ the batch OMP algorithm [37] to form the columns of the sparse matrix $V$.

The total complexity of running batch OMP per signal equals $T_{omp} = K^3 + K^2M + 2ML$ with an additional cost of $M^2L$ to compute the Gram matrix of the $M \times L$ dictionary used. Thus, to compute $V \in \mathbb{R}^{L \times N}$, the number of operations is given by

$$T_{bomp} = N(K^3 + K^2M + 2ML) + M^2L.$$ 

In many of the settings of interest where our aim is to use SEED to cluster data, $K$ is extremely small, e.g., $5 \leq K \leq 10$. When using SEED to provide a faithful representation of a collection of data, $K$ is variable and scaled based upon the total
approximation error of the factorization. However, even in this case $k \leq L \ll N$, and thus the batch OMP algorithm is highly efficient for sparse coding of large collections of signals. Moreover, forming each column of $V$ is independent and thus we can easily parallelize these computations over multiple distributed machines/nodes.

The total complexity of SEED equals

$$T_{seed} = T_{samp} + T_{omp} = N(K^3 + K^2M + 2ML + L^2) + M^2L.$$ 

In practice, $K \ll L$ and $L \leq M$, making SEED very efficient for large datasets when $K$ is small. When $K \leq \sqrt{L}$, then the complexity of SEED is $O(MLN)$. When $M$ and $L$ are much smaller than $N$, the proposed method for sparse factorization has roughly linear complexity in $N$ and thus can be utilized in large-scale settings. The storage requirement for SEED is $2NL$.

### 7.4.2 Comparison with Other Methods

In practice, the number of sampled columns $L \ll N$ and thus oASIS is considerably more efficient than adaptive methods such as [50], which require the computation of $M \times N$ residual matrices at each step, resulting in $O(LMN)$ complexity. Leverage score sampling [51] requires the approximate SVD of $X$, which also requires iterations of cost $O(M^2N)$ over dense matrices. The low complexity of oASIS makes it practical for extremely large matrices where other adaptive sampling schemes are intractable.

In addition to its low runtime complexity, oASIS is capable of benefiting from sparse matrix structure. For sparse matrices, the adaptive method [50] requires the computation of $M \times N$ “residuals,” which may be dense even in the case that $X$ is extremely sparse. In contrast, oASIS requires only the storage of much smaller $L \times N$ matrices. This benefit of oASIS is highly relevant for extremely large datasets where
sparse approximations to similarity matrices are formed using $K$-nearest-neighbor algorithms that only store the most significant entries in each matrix column.

We now compare the complexity of SEED with other matrix factorization methods. Sparse matrix factorization methods such as SPCA exhibit $O(M^2N)$ complexity whereas SEED has $O(LMN)$ complexity. Thus, when $L \ll M$, SEED is considerably faster than SPCA. In addition, SEED is a one-shot method and thus doesn’t rely on multiple iterations before convergence. In contrast, SPCA and KSVD both require many iterations (unknown a priori) before convergence which has a big impact on the speed of these methods.

SEED is also considerably faster and requires less memory than state-of-the-art methods for subspace clustering such as SSC-OMP, which requires computing the $N \times N$ sparse matrix with respect to a dictionary of size $M \times (N - 1)$ for $N$ signals. Thus, SSC-OMP requires $O(N^2)$ complexity making it infeasible for large-scale datasets. In addition, SSC-OMP cannot be paired with a batch OMP algorithm because our dictionary changes for each signal and therefore cannot benefit from the savings obtained by a batch algorithm. In addition, SSC-OMP and other incoherence-driven sampling strategies require $N^2$ storage because we compute the entire affinity matrix and/or Gram matrix for the dataset.
In this Chapter, we introduce a novel scalable framework for large-scale distributed computing for dense and structured datasets. Our framework leverages low-dimensional structure present in data to reduce the cost of distributed computation on dense graphs that encapsulate the dependencies between signals in the dataset. We provide methods for sparse matrix decomposition that map large dense graphs into a factorized graph model consisting of a sparse bipartite graph and a small dense graph. We introduce a host of methods to perform graph-parallel computation on the resulting factorized graph model. Our model exploits sparsity to accelerate distributed matrix computations employed in a wide range of optimization methods. We demonstrate that computation on our factorized representation of dense and low-rank data can be done significantly faster while reducing consumed memory, computational power, and network bandwidth congestion. Communication overhead are presented as a function of data structure.
8.1 Motivation

With the rapid growth of content available from the Internet and other data collection sources, the construction and study of systems that can learn and extract useful knowledge from “big data” has become extremely challenging. This is due to the fact that many algorithms used to analyze data in machine learning, signal processing and computer vision, do not scale to extremely large data sizes.

In order to cope with large datasets that cannot fit in a single machine, smaller subsets of the data are distributed across different computing nodes and each node processes its subset of the data in parallel; after performing local intra-node computation, the computing nodes must “communicate” in order to share the results of their local computation. Thus, a central step in distributed computing is partitioning the data and computation into the available computing nodes and memory blocks in order to reduce communication across distributed computing nodes.

The pairwise correlation (Gram) matrix (or distance matrix) of the variables or signals is often used to split the data into different machines based upon their dependencies. For instance, if $\mathbf{X} \in \mathbb{R}^{M \times N}$ denotes a collection of $N$ signals of $m$-dimensions, then the Gram matrix is given by $\mathbf{G} = \mathbf{X}^T \mathbf{X}$. If we view the Gram matrix as a graph that encodes the pairwise dependencies between signals, then graph-cuts and clustering methods can be employed on this graph to split the data in such a way as to minimize the number of edges that exist between signals on different machines. When the structure of this graph is sparse, such a partitioning has been demonstrated to be effective for big data computations [81].

Partitioning of sparse graphs has been done in multitude of ways, including clustering and graph-based methods. However, spectral graph clustering methods, such as those suggested in [82] are not scalable to massive datasets because they require the computation of a truncated SVD for a $N \times N$ matrix. Heuristic multi-level recursive
partitioning methods such as METIS [83] are more scalable, but their performance is limited, especially for the case of graphs with highly nonuniform edge distributions. An alternative set of graph vertex-cut based approaches attempt at placing partitions of a sparse graph on different machines while minimizing the inter-machine communications. Successful applications to large sparse graphs have been demonstrated [84]. Graph-parallel approaches to distributed computing such as those described in [81] have been shown to yield a significant reduction in the amount of energy and time required to perform large-scale machine learning tasks.

While graph-parallelism has been shown to accelerate machine learning and signal processing tasks for sparse graphs, this approach cannot be applied when the data exhibit dense dependencies or dense Gram matrices. This is due in part by the fact that standard graph-based approaches fail to find efficient cuts to partition the data when the graph is too dense. Thus, the development of general-purpose strategies to perform parallel computation on datasets that admit dense dependencies are of utmost importance.

8.1.1 Our Approach

In the sequel, we introduce RankMap, a novel data-aware framework for efficient execution of a broad class of ML algorithms in a distributed manner. Our framework is data-aware and domain-specific in that it attempts to learn and exploit the low dimensional structures within the data as well as the underlying computational fabric for the pertinent application. Effectively, the RankMap framework provides a set of interfaces and transformations that enable data-aware content analysis, as well as coordinated mapping and optimization of the underlying hardware components. Our overarching goal is to reduce the following critical metrics in high performance distributed computing systems: execution speed, memory usage, and communication
overhead.

The RankMap framework and its APIs target a widely-used class of data analysis algorithms that rely on iterative updates of the parameters of interest. Some prominent examples of such algorithms and their applications are penalized regression methods such as the LASSO [32], belief propagation, expectation maximization, and stochastic optimizations [85, 86, 87]. In all of these settings, the objective function of interest can be translated into a projected gradient descent (PGD) method to handle a broad class of non-smooth objective functions. Since PGD is a generalization of gradient descent (for smooth problems), our framework can be applied to a wide range of standard gradient descent methods as well, including least-squares and iterative eigenvalue decompositions such as the power method.

Our aim is to leverage any low-dimensional structure present in the data to reduce the complexity of distributed matrix computations performed on the data. To do this, we develop methods to decompose large dense graphs into a layered graph model consisting of a large sparse bipartite graph and a small dense graph. We demonstrate that computation on dense and low-rank data can be done significantly faster while reducing consumed memory, computational power, and network bandwidth congestion.

When \( X \) exhibits low-dimensional structure, our goal is to squeeze out redundancies in the data to transform a dense graph corresponding to \( G \) into a factorized graph model of much smaller dimension (near the rank or intrinsic dimensionality of \( X \)). In particular, our goal is to decompose \( X = DV \) into two components in order to obtain an approximation of the positive semidefinite (PSD) matrix \( \hat{G} = V^T(D^TD)V \), where \( D \in \mathbb{R}^{M \times L}, V \in \mathbb{R}^{L \times N}, \) and \( L \leq M \ll N \). When \( X \) admits such a factorization, then we can view the mapping \( V : \mathbb{R}^N \rightarrow \mathbb{R}^L \) as a bi-partite graph that reduces a high-dimensional input to a smaller dimension. Similarly, a mapping corresponding to \( D \)
expands a low-dimensional input back into a higher-dimensional space $\mathbb{R}^M$. See Fig. 8.1 for a schematic representation of our proposed decomposition for dense graphs, where we show how a dense graph $\mathcal{G}$ specified by $\mathbf{G}$ can be effectively decomposed into a two-layer graph consisting of sparse bi-partite graph $\mathcal{V}$ specified by $\mathbf{V}$.

In general, the redundancy factor in $\mathbf{V}$ is much greater than $\mathbf{D}$, where $L \ll N$ but in some cases $m \approx L$. The advantages of this mapping are two fold. First, we are able to reduce the computation, memory usage and communication. Second, we can perform graph partitioning and parallel computation more efficiently.

Mapping a dense graph to a factorized graph model reduces the total number of edges in the resulting graph from $N^2$ to at most $2L(M+N)$. This mapping represents a significant reduction in the total communication and storage required to perform computations on the resulting data structure. However, in order to reduce the number of edges in the resulting graph to lessen computation and communication (edges across different machines), we enforce sparsity in $\mathbf{V}$ by computing an approximation to the following objective NP hard objective function:

$$\min_{\mathbf{V} \in \mathbb{R}^{L \times N}} \|\mathbf{V}\|_0 + \lambda L \quad \text{subject to} \quad \|\mathbf{X} - \mathbf{D}\mathbf{V}\|_F \leq \delta,$$  \hspace{1cm} (8.1)

where $\delta$ is the amount of error allowed in the resulting factorization and $\lambda$ is a user set regularization parameter that enables a tradeoff between the sparsity in $\mathbf{V}$ with the size of the factorized model $L$. In other words, when $\lambda$ is large, our aim is to learn a factorized model with small dimension $L$ and possibly denser $\mathbf{V}$. When $\lambda$ is small, then our aim is to find a sparser $\mathbf{V}$ by introducing some redundancy in the columns of $\mathbf{D}$ (larger $L$).
Figure 8.1: Graph decomposition approach behind RankMap. On the top, we depict the decomposition of a dense matrix $A$ into a product of a dense subset of columns of $A$ (given by $D$) and a sparse matrix $V$. Below, we show how the corresponding Gram matrix for the data $G = A^T A$ can be transformed into a two-layer graph specified by $D$ and $V$.

8.2 RankMap Framework

In this Section, we provide an overview of our approach. We explain our objective, target applications, and introduce the main components required to achieve our optimization goals.

8.2.1 Target Applications

RankMap targets advanced knowledge extraction techniques that involve iterative updates on dense matrices that admit low dimensional structures. In particular, our framework targets a broad class of optimization problems that can be solved either via a standard gradient descent method or projected gradient descent (PGD) methods like those described in Sec. 3.4.2.
PGD is a generalization of standard gradient descent methods that generates a sequence of approximate solutions to a wide range of non-smooth objective functions. A large number of machine learning objective functions and penalized regression methods such as the LASSO or BPDN [25], Ridge regression [88], and the Elastic net [89], can be solved via a PGD approach.

To ground RankMap in a real-world problem, we use an image reconstruction application as a running example. In order to make our notations consistent, we write vectors in bold lowercase script and matrices in bold uppercase script. Let $X$ denote a matrix whose columns ($y$’s) are patches of a noisy image. Let $A$ (the domain data) be a dense and massive overcomplete dictionary created from a large collection of sample image patches. The reconstruction problem is to find a solution vector $x$ corresponding to each noisy patch $y$ such that the following objective function (OF) holds:

$$
\arg \min_x \|Ax - y\|_2 + f(x),
$$

where $f(x)$ is a penalty function which regularizes the least-squares problem to avoid overfitting. For example $f(x)$ could be $\lambda_1\|x\|_1$ (Lasso regression or $\ell_1$ regularization), $\lambda_2\|x\|_2$ (Ridge regression or $\ell_2$ regularization), or a combination of both (Elastic net regression); where $\lambda_i$ ($i = 1, 2$) is a regularization coefficient. The objective function can be solved by using a PGD-based iterative method like those described in Sec. 3.4.2. Recall, that the OF can be solved by an iterative update equation of the following form:

$$
x^{iter+1} = x^{iter} - \gamma(Gx^{iter} - A^T y),
$$

followed by a low-complexity operation (such as soft-thresholding) to account for
Figure 8.2: Overview of the RankMap framework.

\( f(x) \) at each iteration. In Eq. 8.3, \( G \) (the dependency or Gramian matrix) is equal to \( A^T A \), \( x^{iter} \) is an estimate of \( x \) at iteration \( iter \), and \( \gamma \) is a step size. For each input patch, the iterative update has to be applied repeatedly until a stopping (convergence) criteria is met. Applications of penalized regression range from image denoising and super-resolution [37] to genomic data analysis and classification [90].

Applying the iterative methods at large scale can be very challenging due to the huge size of the multiplicative matrices specially when they are dense. For example, the main computational complexity of Eq. 8.3 arises from the \( Gx^{iter} \) term. As a case-study in our evaluations, we solve an image reconstruction problem for a light field dictionary of size \( 18k \times 100k \). This dictionary induces more than 3.6 billion floating point multiplications per iteration in order to reconstruct of a single sparse solution.

### 8.2.2 Overview of RankMap

Fig. 8.2 shows an overview of the RankMap framework. RankMap consists of three main components which we summarize below.
1. **Sparse Decomposition of Data:** RankMap employs a scalable sparse decomposition method that reduce the size of the domain content to the limits of its rank,

2. **Design flow of iterative updates:** RankMap proposes two designs to implement the flow of updates. The first design takes a vertex-centric model for applying the iterative computation. The second design takes a matrix-based representation and computation approach. The proposed models exploit the low-dimensional decomposition components and the presented sparsity to carry out the challenging phase of the iterative update approaches, i.e., the expensive matrix multiplications. However, depending on the algorithm, further low-overhead processing may be needed to complete an iteration. The RankMap framework and its interfaces can be readily modified to incorporate those different low-overhead processes to support a large class of iterative problems.

3. **Partitioning and distributed computing:** RankMap proposes a distributed model that efficiently embeds the decomposed data on a cluster of computing nodes. It introduces an efficient partitioning algorithm that balances the storage in each computing node while reducing the inter-node communication. Partial updates on the large but sparse data (corresponding to $V$) are done locally on each machine. The results are then communicated to a central node where computations on $D$ are performed. RankMap's proposed graph partitioning and vertex-centric computing flow is implemented using GraphLab API [91]. Its matrix-based distributed computing model is implemented using standard message passing interface (MPI). Our evaluations show that RankMap yields significant performance improvements in comparison with tailored distributed GraphLab and MPI implementations.
The proposed decomposition is an offline process that must only be completed once. Once the decomposition is done, we can apply various type of iterative update algorithms on the decomposed components for different input data. In the subsequent Sections, we provide a thorough description of each of the three components that comprise the RankMap framework.

8.3 Sparse Matrix Decomposition Approach

In this Section, we present our novel sparsity driven and scalable method for data decomposition (the Decomposition phase in Fig. 8.2).

8.3.1 Sparse Matrix Decomposition via SEED

Our approach decomposes data matrix $A$ by finding an approximate solution to the following NP hard problem:

$$\min_{D \in \mathbb{R}^{M \times L}, V \in \mathbb{R}^{L \times N}} \| V \|_0 + \lambda L \quad \text{subject to} \quad \frac{1}{N} \| A - DV \|_F^2 \leq \delta,$$

where $\| V \|_0$ measures the total number of non-zeros in $V$ and $\| \cdot \|_F$ is the Frobenius norm. Eq. 8.4 naturally expresses RankMap’s goal to find a sparse and low-dimensional decomposition of $A$, as measured by $\| V \|_0$ and $\lambda L$ respectively, that provides a fixed amount of approximation error. This objective is non-convex and thus computing an exact solution is difficult because $D$ and $V$ are coupled and also due to the non-convex $\ell_0$ (sparsity) penalty on $V$.

To compute an approximate solution to the sparse decomposition problem in Eq. 8.4, we employ a variant of SEED in Alg. 2 that uses a sequential error selection (SES)-based approach for subsampling the data. Just as in SEED, we form the reference set $D$ by sub-sampling columns of $A$; however, we employ a SES-based criterion until
the squared least-squares error of each column of the approximation is less than a user set parameter $\delta$. After computing the reference set $D$, Eq. 8.4 is equivalent to a sparse approximation problem, where each column $v_i$ of $V$ is a sparse approximation of the column $a_i$ of $A$ with respect to $D$. This approximation problem can be solved by an efficient greedy routine called orthogonal matching pursuit (OMP) [22]. When the data is sufficiently sampled such that the span of columns of $D$ is “close” to the span of columns of $A$, OMP can successfully find $V$ such that the error tolerance criterion is met. Setting the error tolerance to zero ($\delta = 0$) guarantees that we obtain the same approximation error as existing methods for column subset selection-based matrix factorization [46].

In Alg. 8, we outline RankMap’s adaptive SES-based method to form $D$. The idea behind this adaptive sampling approach is to randomly select a small number of columns from $A$ to form $D$ at the beginning. Then more columns are added to $D$ according to the sampling distribution given by the relative decomposition error of columns of $A$. The column selection terminates once $L$ columns are selected, where $L$ is a user-defined input. After forming $D$, $V$ can be computed by performing an OMP with respect to the $D$ and the error tolerance $\delta$.

Note that Algorithm 1 can be easily modified to eliminate the need to specify $L$. One approach to do so is to change the condition of the while loop to be the $\ell_2$ error:

$$\sum_{i=1}^{N} \|DD^+a_i - a_i\|_2 < \delta^*,$$

for a desired $\delta^*$ error tolerance. Thus, SEED selects $l_{select}$ columns from $A$ until the error criterion is met. Alternatively, one can apply OMP in Step 3 by setting the desired sparsity-level (percentage of non-zeros in columns of $V$) instead of the error. In Sec. 8.6, we provide evaluation results for these alternative approaches.
Algorithm 8: Sequential Error Selection-Based SEED

**Input:** Matrix $A \in \mathbb{R}^{M \times N}$, the number of columns to select at each iteration $L_{\text{select}}$, an error tolerance $\delta$, and the maximum number of columns to select $L$.

**Output:** Factors $V \in \mathbb{R}^{L^* \times N}$ and $D \in \mathbb{R}^{M \times L^*}$, where $L^* \leq L$.

0. Initialize $D$ by adding and normalizing $L_{\text{select}}$ columns from $A$ with uniform random sampling.

**while** $\text{ncols}(D) < L$ **do**

1. Compute the relative error for all $i = \{1, \ldots, N\}$:
   
   $$\text{err}(D, a_i) = \frac{\|DD^+a_i - a_i\|_2}{\|a_i\|_2}.$$  

2. Select $L_{\text{select}}$ columns from $A$ according to the distribution of the relative error, where the probability of selecting the $i^{th}$ column $p(i) \propto \text{err}(D, a_i)$.

3. Normalize the columns of $A$ selected in Step 2 and add to $D$.

4. If the maximum relative error $\max_i \text{err}(D, a_i) < \delta$, then exit while loop.

**end while**

3. Compute $V$ by applying OMP to form an error-constrained representation of each column of $A$ with respect to $D$, where the maximum error equals $\delta$.

8.3.2 Impact of Approximation Error on Iterative Updates

As discussed in the Intro, one of the key applications of the RankMap framework is enabling fast and efficient iterative updates that involve the Gram matrix $G = A^T A$ of the dataset. In particular, the proposed framework can be readily applied to a wide range of regularized least-squares problems that can be written as in Eq. 8.2 including: ridge regression, $\ell_1$-minimization (BPDN and LASSO), and the elastic net.

While the proposed sparse matrix decomposition method (SEED) can be used to form an exact decomposition of the data (by setting $\delta = 0$ and allowing $L \approx M$), in practice, by allowing some amount of error in the factorization, this significantly reduces the sparsity level of $V$ and thus results in considerable acceleration of the iterative updates computed with the approximate Gram matrix.
Bound on Solution Error

We now discuss how the approximation error affects the solution obtained via the approximate Gram matrix ($\hat{G}$) in comparison to the optimal solution for the original problem (with $G$).

In a previous theoretical analysis of the impact of kernel approximations on learning methods [92], the authors study the accuracy of kernel-based learning methods when an approximate kernel matrix is used. The Gram matrix is a particular type of kernel matrix and thus the results obtained for kernel ridge regression can be applied here. Recall that the ridge regression objective function is given by:

$$\|Ax - y\|^2_2 + \lambda\|x\|^2_2$$  \hspace{1cm} (8.5)

and the closed-form solution of (8.5) is given by

$$x_{opt} = (G + \lambda I)^{-1}A^Ty.$$  

Using a similar approach to that taken in [92], we can prove the following result for the impact of our factorization approach for ridge and least-squares regression. To state this result, we must assume that Alg. 8 selects a sufficient number of columns from $A$ such that the total error for each column is bounded as follows, $\|\hat{A} - A\|^2_2 \leq \delta$ and that we use the same error parameter $\delta$ when solving OMP to compute $V$. Under these assumptions, we prove the following result which describes the impact of the factorization error on the error of the final solution vector obtained via RankMap.

\textbf{Theorem 10} Let $x_{opt}$ denote the optimal solution to Eq. 8.5 and let $x^*$ denote the solution obtained by utilizing the approximate Gram matrix $\hat{G} = \hat{A}^T\hat{A}$. Assume that the squared $\ell_2$-error of each column of $A$ is bounded by $\delta$, i.e., $\|a_i - \hat{a}_i\|^2_2 \leq \delta$ and the
input is normalized such that \( \|y\|_2 = 1 \). Then,

\[
\|x_{opt} - x^*\|_2 \leq \frac{\delta_1\|A\|_2}{\lambda^2},
\]

where \( \delta_1 = L(2\delta + \delta^2) + (N - L)(2\sqrt{\delta} + \delta) \).

The proof of this Thm. is provided below. However, before jumping into the proof, we make a few remarks.

**Remark.** The result in Thm. 10 demonstrates the connection between the factorization error and how this error propagates into the solution to a ridge regression problem. Clearly, as \( \lambda \) increases (as we place less importance on the least-squares fit between the input and its reconstruction), the total solution error decreases. In addition, we observe linear dependence on \( \delta_1 \) or the maximum column sum of the error residual \( G - \hat{G} \), which measures the total absolute error between the inner products of a single column and the rest of the columns in \( A \). Thus, as long as we sample a sufficient number of columns to ensure that \( \delta \) is small, this error will have minimal impact on the quality of our final solution. Lastly, we see that the solution error is dependent on the spectral norm \( \|A\|_2 \), which depends on the coherence of the matrix \( A \); for matrices containing incoherent columns, the spectral norm will be close to one.

**Proof of Thm. 10**

To prove Thm. 10, we must first prove the following Lemma which relates the factorization error in SEED (with respect to \( A \)) with the factorization error of the Gram matrix.

**Lemma 5** Assume that the squared least-squares error of each column of the residual error matrix \( E_A = \hat{A} - A \) is bounded by \( \delta \) and that the columns of \( A \) are scaled such
that norm of each column of $A$ is less than or equal to one. Then the maximum column norm of the factorization error

$$\delta_1 = \|\mathbf{G} - \mathbf{G}\|_{1,1} \leq L(2\delta + \delta^2) + (N - L)(2\sqrt{\delta} + \delta).$$

**Proof.** This bound is obtained by computing the absolute error of the residual $E_G = \hat{\mathbf{G}} - \mathbf{G}$, based upon whether the signal has been sampled to form $D$ or not. Let $S$ index the set of columns from $A$ selected to form $D$ and $T$ index the remaining $N - L$ columns. Clearly, the error of the inner products between sampled signals, \[\|A_S^T A_S - \hat{A}_S^T \hat{A}_S\|_F = 0.\] Now, if we examine the $L \times (N - L)$ block containing the inner products between $A_S$ and the remaining unsampled signals $A_T$, we can bound the error of each entry of the residual $E_{ST} = A_S^T A_T - \hat{A}_S^T \hat{A}_T$ as follows. Let $a_i$ denote a sampled column where $i \in S$ and $\hat{a}_j = a_j + \epsilon_j$ denote an approximation to unsampled column $a_j$ where $j \in T$ and by our original assumption, $\|\epsilon_j\|_2^2 \leq \delta$.

Each entry of $E_{ST}$ is bounded as follows.

$$\|a_i^T a_j - \hat{a}_i^T \hat{a}_j\|_2^2 = \|a_i^T (a_j - (a_j + \epsilon_j))\|_2^2$$

$$= \|a_i^T \epsilon_j\|_2^2 \leq \|a_i\|_2 \|\epsilon_j\|_2 \leq \sqrt{\delta},$$

where the simplification in the last step comes from our assumption that the norm of each column of $A$ is bounded by one and that $\|\epsilon_j\|_2^2 \leq \delta$ for all $j \in T$. Since $G$ is symmetric, each entry of $E_T S = A_T^T A_S - \hat{A}_T^T \hat{A}_S$ is also bounded by the same value.

Now, we must compute the absolute error contained within the large $N - L \times N - L$ sub-block containing the inner products between two unsampled columns or the residual $E_T = A_T^T A_T - \hat{A}_T^T \hat{A}_T$. To do this, we follow a similar approach as in our previous calculation. This time both the columns $a_i$ and $a_j$ are unsampled and thus,
Each entry of $E_T$ is bounded as follows.

$$
\|a_i^T a_j - \hat{a}_i^T \hat{a}_j\|_2^2 = \|a_i^T a_j - (a_i + \epsilon_i)^T (a_j + \epsilon_j)\|_2^2 \\
= \|a_i^T a_j - a_i^T a_j - (a_i^T \epsilon_j + a_j^T \epsilon_i + \epsilon_i^T \epsilon_j)\|_2^2 \\
\leq \|a_i\|_2 \|\epsilon_j\|_2 \\
= \|a_i^T \epsilon_j + a_j^T \epsilon_i + \epsilon_i^T \epsilon_j\|_2^2 \\
\leq 2\|a_i^T \epsilon_j\|_2^2 + \|\epsilon_i^T \epsilon_j\|_2^2 \\
\leq 2\|a_i\|_2 \|\epsilon_j\|_2 + \|\epsilon_i\|_2 \|\epsilon_j\|_2 \\
\leq 2\sqrt{\delta} + \delta.
$$

Finally, we can bound the maximum column sum in the entire residual $\|E_G\|_{1,1} \leq L(2\delta + \delta^2) + (N - L)(2\sqrt{\delta} + \delta)$, by considering a column consisting of $L$ entries from $E_{ST}$ and $N - L$ entries from $E_T$. This completes our proof. \hfill \Box

Going back to our proof of Thm. 10, the difference between the true solution $x_{opt}$ and the solution obtained by using $\hat{G}$ can be computed exactly due to the fact that we have a closed-form solution for the ridge regression problem. In this case, we assume that at the first iteration, we compute $b = A^T y$ as an initialization for our method. Writing this out, we have the difference between the optimal solution and the solution obtained by applying iterative updates with $\hat{G}$ is as follows,

$$
x_{opt} - x^* = (G + \lambda I)^{-1} b - (\hat{G} + \lambda I)^{-1} b \\
= [(G + \lambda I)^{-1} (G - \hat{G}) (\hat{G} + \lambda I)^{-1}] b.
$$
Now, we can bound the error between these two solutions.

\[
\|x_{\text{opt}} - x^*\|_2 \leq \|(G + \lambda I)^{-1}\|_2 \|G - \hat{G}\|_2 \|(\hat{G} + \lambda I)^{-1}\|_2 \|b\|_2
\]

\[
\leq \frac{\|G - \hat{G}\|_2 \|b\|_2}{\lambda_{\text{min}}(G + \lambda I) \lambda_{\text{min}}(\hat{G} + \lambda I)},
\]

where \(\lambda_{\text{min}}(G + \lambda I)\) is the minimum eigenvalue of \(G + \lambda I\).

We can simplify this bound further by utilizing the fact that since \(G\) and \(\hat{G}\) are both PSD, both \(\lambda_{\text{min}}(G + \lambda I)\) and \(\lambda_{\text{min}}(\hat{G} + \lambda I)\) must be greater than the regularization parameter \(\lambda\). Furthermore, the spectral norm \(\|A\|_2 \leq \sqrt{\|A\|_1 \|A\|_\infty,\infty}\), where \(\|A\|_1\) is the maximum absolute column sum of \(A\) and \(\|A\|_\infty,\infty = \|A^T\|_{1,1}\). Using both of these facts,

\[
\|x_{\text{opt}} - x^*\|_2 \leq \frac{\|G - \hat{G}\|_2 \|b\|_2}{\lambda^2}
\]

\[
\leq \frac{\|G\|_{1,1} \|b\|}{\lambda^2}
\]

\[
\leq \frac{\|G\|_{1,1} \|A\|_2}{\lambda^2},
\]

where the simplification in the last step follows from the assumption that \(\|y\|_2 = 1\) and thus, \(\|A^Ty\|_2^2 \leq \|A\|_2\).

Finally, by invoking Lemma 5, we can bound the spectral norm \(\|G - \hat{G}\|_{1,1} \leq L(2\delta + \delta^2) + (N - L)(2\sqrt{\delta} + \delta)\). This completes our proof. \(\square\)

### 8.3.3 Sparsity Guarantees for Unions of Subspaces

In both our analysis of the complexity of iterative updates (Sec. 8.5.3) and also in our evaluations, the sparsity of the columns of \(V\) is a key limiting factor in the cost of an update with RankMap. While determining the sparsity of \(V\) is in general, a difficult problem, when the data lives on (or close) to a union of subspaces, then
we can leverage the fact that SEED is rank revealing to bound the sparsity level of columns of $V$ based upon the dimension of the subspaces they live on.

Recall that Thm. 6 states that when a sufficient number of columns from $A$ that lie on a $k$-dimensional subspace $S$ are selected to form the reference set $D$, then the representations of the remaining columns (not selected to be included in $D$) that lie in the same subspace will admit a $k$-sparse representation. This result can be directly applied to bound the complexity of iterative updates within the RankMap framework. Based upon our performance bounds and evaluations provided in Sec. 8.5.3 and Sec. 8.6 respectively, the sparsity level of each column of $V$ plays a significant role in the amount of communication and computation required to carry out an iterative update. Thus, the fact that the sparsity of each column of $V$ is bounded by the subspace dimension, provides a bound on the communication and computation required within our framework. This in turn provides significant reduction in the overhead required to carry out iterative updates.

8.3.4 Complexity Analysis

In Algorithm 1, the complexities of Steps 1 and 2 (the adaptive column selection) are less than $O\left(\frac{L}{L_{\text{select}}} (L^3 + L^2M + LMN)\right)$. The complexity terms correspond to computing $D^+, DD^+$, and $DD^+A$. The latter term’s induced complexity is linear in $N$, but is parallelizable (i.e., each $DD^+a_i$ can be computed independently). In our evaluations $\frac{L}{L_{\text{select}}} = 10$.

The complexity of Step 3, using the batch OMP method [37], is $O(LMN + L^3N)$. Since each column of $V$ is computed independently, Step 3 is also highly parallelizable. Let $n_c$ be the number of parallel processing cores. By storing $D$ (which is small and dense) and a fraction of columns of $A$ in each node (i.e., $\frac{N}{n_c}$ columns), the complexity of Step 3 reduces to $O(\frac{N}{n_c} (LM + L^3))$. Sec. 8.6.2 provides a more detailed numerical
and experimental complexity analysis of the proposed method in comparison with other popular data decomposition methods. Note that the decomposition of a large data is an off-line one-time procedure. For instance in our image reconstruction example, once a large-enough dataset of light fields is decomposed, it can be used for reconstruction of any input light field.

8.4 Execution Flow of Iterative Updates

In this Section, we introduce our approach for applying iterative computation on decomposed data (the execution phase in Fig. 8.2). In order to represent the decomposed data, exploit the sparsity of $V$, and apply the iterative computations, we implement two separate models, one is graph-based and the other one is matrix-based.

8.4.1 Graph-based Model

We form a layered graph-based model to represent and perform subsequent computational operations on data. Representing data in a graph model enables us to efficiently take advantage of sparsity of $V$ and the decomposed format. It not only improves the required memory and computations, but also facilitates efficient distributed computing, which will be discussed more in Sec. 8.5.

RankMap’s graph model is shown in Fig. 8.3. The decomposed data graph denoted by $G_A(S_X, S_P, S_R)$ is a 3-layer graph with vertex sets $S_X = \{X_i\}_{i=1}^N$ in the bottom layer, $S_P = \{P_i\}_{i=1}^L$ in the middle layer, and $S_R = \{R_1\}$ in the top layer. Each non-zero element in $V$, e.g., $V_{ij}$, is represented by an edge which connects $X_i$ to $P_j$. Each column of $D$, e.g., $D_i$, is represented by an edge which connects $P_i$ to $R_1$. Value of vertices in $S_X$ correspond to the elements of vector $x$ in Eq. 8.3.
**Vertex Update Computations:** To ensure that an algorithm can run in a fully distributed fashion, we perform the computations at vertex level: each vertex updates itself by collecting information from its neighbors. After a vertex is fully updated, it then makes its value available to its neighbors. Similarly, other vertices also update themselves by collecting their own neighbors’ values. To implement the vertex computations, we take advantage of GraphLab, an existing vertex-centric API [91]. To implement a vertex update, assume that vertex $P_j$ in $S_P$ has $n_{P_j}$ adjacent edges. During a vertex update, $P_j$ first multiplies $V_{ij}x(i)$ for $(1 \leq i \leq n_{P_j})$ in parallel and then reduces the results to update itself.

**Iterative Updates:** A key step of RankMap is performing iterative computations on the dependency matrix to update $x$ (Eq. 8.3). The vertices in $S_X$ are initialized with random values $^1$. The vertex values are updated repeatedly until the values of vertices in $S_X$ converge to the optimal solution. Thus, a convergence criterion should be set. The following two stopping criteria are common: the mean squared error

---

$^1$In general, initial point should not affect convergence to the optimal solution.
(MSE) \( \|Ax - y\|_2 \), and the difference between the elements of \( x \) in two consecutive iterations. The latter criterion is an approximate approach which is used to avoid the expensive MSE computations. Note that computing \( Ax \) to evaluate the MSE can be expensive for a large and dense \( A \). RankMap efficiently supports both of these stopping criterion.

The computation flow of RankMap on the graph-based model is shown in Fig. 8.3. We break down Eq. 8.3 into the following 4 steps:

1. Computing \( p = Vx \): At the beginning of an iteration, the vertices in \( S_P \) perform vertex updates with respect to edges of \( V \) to compute \( Vx \).

2. Computing \( r = Dp - y \): In this step, vertex \( R_1 \) updates itself with respect to its edges (columns of \( D \)) and the updated values of vertices in \( S_P \) (from Step 1) to compute the vector \( Dp \). Afterwards, the norm of the residual or \( \|Dp - y\|_2 \) can be easily computed.

3. Computing \( p = D^T r \): In this step, vertices in \( S_P \) update themselves again, this time with respect to columns of \( D \) and the updated vector in \( R_1 \) (updated from Step 2) to compute \( D^T(Dp - y) \).

4. Computing \( x = V^T p \): In this last step, vertices in \( S_X \) update themselves with respect to edges in \( V \) and the values of vertices in \( S_P \) (updated from Step 3) to compute \( V^T p \), where \( V^T p = V^T D^T DVx - V^T D^T y \).

After completing Steps 1-4, subsequent processing can then be applied on the elements of \( x \). For example, to implement a \( \ell_1 \) minimization problem, a soft thresholding operation truncates small elements if \( x \) in each iteration.
8.4.2 Matrix-based Model

In addition to graph model, we develop an alternative board upon matrix computation. In this model, we present data in a matrix format. To exploit the sparsity, we use a compressed column storage format to store $V$. The matrix $D$ is stored in a regular matrix format. The design is similar to the graph-based approach in that the same order (Steps 1 to 4 in Sec. 8.4.1) is followed for performing an iterative update on the decomposed data. However, instead of vertex updates, matrix-vector multiplications are only performed on the non-zero elements.

8.5 Data Partitioning and Distributed Computing

In this Section, we present our methods and architectures to efficiently embed the designs introduced in Sec. 8.4 in a distributed system with multiple computing nodes. RankMap’s distribution scheme observes the following parameters: number of available computing nodes, available memory of each node, and inter-node communication cost. In the following, we assume there are $n_c$ available computing nodes.

8.5.1 Distributed Graph Allocation

RankMap pursues two main objectives to distribute $G_A(S_X, S_P, S_R)$: first, to balance the number of components assigned to each node, and second, to minimize the internode communications characterized by the edges. Since the edge distribution of $G_A$ is highly non-uniform ($L \ll N$) and vertex partitioning inevitably results in many edge-cuts across the computing nodes, RankMap applies a vertex-cut method. In this method graph edges are evenly partitioned into the computing nodes such that number of vertices spanning nodes is minimized. As a result of edge partitioning, some vertices may be copied onto two or more nodes. In this case we assign one of
the copies to be the master vertex and the others to be the replica vertices (these definitions are borrowed from GraphLab [81]). The replicas directly cause (expensive) inter-node communication costs. In Algorithm 9, we present the our proposed edge partitioning method.

Fig. 8.4 shows the graph-based distributed design. The proposed edge partitioning algorithm is highly efficient in that it does not induce any replicas for vertices in $S_X$ and $S_R$. However from step 2 of the algorithm, replicas of vertices in $S_P$ may exist.
in computing nodes other than the central node. At the beginning of an iteration, master vertices in $S_P$ and their replicas perform vertex updates with respect to $S_X$. The replicas send the updated values to their own master vertices in the central node. The master vertices in $S_P$ reduce the received values ($p = Vx$). Then master vertex $R_1$ performs a vertex update ($r = Dp - y$). Next master vertices in $S_P$ complete vertex updates with respect to $S_R$ and broadcast the results to their own replicas ($p = D^TR$). Finally master vertices in $S_X$ update themselves ($x = V^Tp$). RankMap implements the proposed partitioning and distributed computation flow on top of the distributed GraphLab API [81].

### 8.5.2 Distributed Matrix Allocation

RankMap partitions columns of $V$ uniformly across the computing nodes. Thus, $\frac{N}{nc}$ number of columns are assigned to each node and the estimated $x$ is divided into chunks of size $\frac{N}{nc} \times 1$. Each chunk is then allocated to the node that hosts the corresponding columns of $V$. Fig. 8.5.1 shows the RankMap distributed matrix computation model.

To compute $Vx$, matrix-vector multiplications are performed locally on the columns of $V$ and elements of $x$. The resulting $L \times 1$ vectors are then reduced to a central node to create $p = Vx$. Next, $D^T(Dp - y)$ is computed locally in the central node. The resulting $L \times 1$ vector is broadcasted back to the computing nodes. Finally each of the computing nodes can locally update the $x$ vector elements by multiplying their local $V^T$ to the broadcasted vector. RankMap implements the distributed matrix computations using MPI.
Algorithm 9: Decomposed Graph Partitioning

**Input:** Decomposed data graph $G_A(S_X, S_P, S_R)$ and number of computing nodes $n_c$.

**Output:** Assignment of edges to the underlying computing nodes.

1. Distribute master of vertices $X_i \in S_X$ uniformly onto the available computing nodes such that vertex chunks of size $\frac{n}{n_c}$ are assigned to each node.
2. Add the edges between vertices $X_i \in S_X$ and $P_j \in S_P$ to the node in which the corresponding master of $X_i$ resides.
3. Add master of vertices $P_i \in S_P$ and $R_1 \in S_R$ to a central node.
4. Add the edges between the vertices $P_i \in S_P$ and $R_1 \in S_R$ to the central node.

### 8.5.3 Performance Bounds

We provide performance analysis of RankMap’s proposed partitionings and distributed designs. Function $nnz()$ returns the number of non-zeros of its input matrix and function $rep()$ returns the number of replicas of its input vertex.

**Performance Bounds for Graph-based Model**

- **Memory usage**
  \[
  \# \text{ edges} \propto \text{nnz}(V) + L.
  \]
  \[
  \# \text{ vertices} \propto N + \sum_{1 \leq i \leq L} \text{rep}(P_i).
  \]

- **Computation** (per iteration)
  \[
  \# \text{ additions} \propto 2(\text{nnz}(V) + ML) + \sum_{1 \leq i \leq L} \text{rep}(P_i).
  \]
  \[
  \# \text{ multiplications} \propto 2(\text{nnz}(V) + ML).
  \]

- **Communication**
  \[
  \# \text{ edge-cuts} \propto 2 \sum_{1 \leq i \leq L} \text{rep}(P_i).
  \]

Each of the computing nodes receive approximately $\frac{1}{n_c}(n + \sum_{1 \leq i \leq L} \text{rep}(P_i))$ vertices and $\frac{1}{n_c}\text{nnz}(V)$ edges. The central node has $L$ additional edges between the
master vertices in $S_P$ and $R_1$. The computation cost is induced by vertex update operations. The communication overhead is incurred by the message passing across master and replica vertices in $S_P$.

**Bounds on $\sum_{1 \leq i \leq L} \text{rep}(P_i)$.** From above analysis, it is clear that reducing number of replicas of $S_P$ reduces the communication overhead. The following are the bounds on the total number of replicas: $l \leq \sum_{1 \leq i \leq L} \text{rep}(P_i) \leq Ln_c$. The inequalities hold since each $P_i$ is replicated at least once and at most $n_c$ times (one replica per computing node). Both $L$ and $n_c$ are much smaller than the size of the graph. Thus, RankMap’s graph-based model readily provides efficient/balanced computation and reduced communication without using complicated and costly graph partitioning algorithms. The minimum communication corresponds to a case where $V$ is block-diagonal.

**Performance Bounds for Matrix-based Model**

- **Memory usage**
  \[ \# \text{non-zeroes} \propto (\text{nnz}(V) + LM) + N + M. \]

- **Computation** (per iteration)
  \[ \# \text{additions} \propto 2(\text{nnz}(V) + LM + Ln_c) \]
  \[ \# \text{multiplications} \propto 2(\text{nnz}(V) + LM) \]

- **Communication**
  \[ \#\text{edges} \propto 2Ln_c. \]

Matrix $V$ is stored in a compressed column storage. Thus only the non-zero values are stored and operated on. Matrix $D$ is stored in a regular dense matrix format. The communication corresponds to sending and receiving the $L \times 1$ vectors from each computing node to the central node.
8.6 Evaluations

8.6.1 Evaluation Setup

In this Section, we describe the distributed tools and datasets that we utilize to evaluate the RankMap framework.

Dataset and Application

Light field dataset. The RankMap framework is applied to a massive light field dictionary collected from Stanford Computer Graphics Laboratory Archive [62]. A light field is a multi-dimensional array of images. Each image is captured from a slightly different viewpoint. Combining the images enables creating new views and representing observer positions not present in the original array. Most currently available devices trade image resolution for the ability to capture different views of a light field. As a result, the final image resolution is reduced by orders of magnitude compared to the sensor resolution. To address this challenge $\ell_1$ regularized least-squares methods (Eq. 8.2) on over-complete light field dictionaries have been proposed [93].

Application of such methods so far has been limited to small arrays and lower resolution images. To the best of our knowledge, the largest light field dictionary that has been used for image reconstruction is developed in a recent work by [93]. In that work the dictionary is constructed from 5000 light field samples collected from a $5 \times 5$ array of $480 \times 270$ pixels. RankMap is tested on two different sets of light field dictionaries that are constructed as follows: (i) 10k randomly selected atoms of a $5 \times 5$ array of light field (collected from Chess Images), each atom consists of 25 $8 \times 8$-patches. The resulting $A$ matrix is of size $1.6k \times 10k$ with 16 million non-zeros. (ii) 100k randomly selected atoms from the collection of all available light fields in
the archive. Each atom consists of 8 × 8-patches from 17 × 17 light field arrays of up to 1536 × 1152 pixels. The resulting \( A \) matrix is of size 18496 × 100k with 1.849 billion non-zeros.

**Synthetic dataset.** To evaluate RankMap for different data structures, we create synthetic decomposed data for \( n = 10M, m = 1k \) with varying \( L \) and sparsity levels in \( V \).

**Computing Platform**

To evaluate the decomposition methods on light field dataset (i) an 8-core CPU (Intel Core\textsuperscript{TM}i7 processor) with 12GB of RAM is used. For computations on light field dataset (ii), a cluster of 16 m3.large nodes (machines) on Amazon EC2 is instanciated. Each node has 16 cores (two Intel Xeon processors) at 7.5GB of RAM per node. The synthetic datasets are evaluated on IBM iDataPlex computing cluster which has 2304 cores in 192 Westmere nodes (12 processor cores per node) at 48GB of RAM per node.

**Distributed Tools**

The RankMap framework’s sparse graph-based design is implemented using GraphLab, a high-level graph-parallel abstraction [81]. GraphLab enables vertex-update-based computations. We implemented RankMap’s customized partitioning using Graphlab’s ingress class. The proposed architectures are mapped efficiently into GraphLab API (Sec. 8.5). Note that the GraphLab framework is only suitable for sparse dataset, thus it is not suited to fit the original dense dataset (\( A \)) unless after the proposed sparse decomposition.

The RankMap framework’s sparse matrix-based computations is implemented using Eigen library to represent data in a compressed column storage (CCS) format [94]. It uses MPI standard system to distribute the data and computation.
Table 8.1: Complexity of different decomposition methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Error</th>
<th>Density of V</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVD</td>
<td>0.077</td>
<td>100%</td>
<td>$M^2N + M^3$</td>
</tr>
<tr>
<td>SPCA</td>
<td>0.14</td>
<td>11%</td>
<td>$LMN + M^2N + M^3$</td>
</tr>
<tr>
<td>K-SVD</td>
<td>0.11</td>
<td>56%</td>
<td>$L^2MN + L^3N$</td>
</tr>
<tr>
<td>SEED-SES</td>
<td>0.15</td>
<td>13%</td>
<td>$LMN + L^3N$</td>
</tr>
</tbody>
</table>

8.6.2 Decomposition Results

In this part we evaluate different properties of the adaptive SEED decomposition method using light field dataset (i).

Comparison of Decomposition Methods

In order to evaluate the performance of RankMap’s decomposition (i.e., adaptive SEED), we compared it with other known decomposition methods including SVD, SPCA, and K-SVD. Decomposition parameters are set as follows (where applicable): $\delta = 0.1$ and $L = 10$. Table 8.1 shows the resulting decomposition error and density of $V$ for each method. As it is expected, (truncated) SVD results in the most accurate decomposition. The sparsity of $V$ from adaptive SEED is comparable to that of SPCA, and is considerably better than K-SVD. The theoretical complexities of each of the algorithms are also shown in the table.

Fig. 8.6 (a) shows the growth of the computation time as we scale dataset dimensions. All dimensions $L$, $M$, and $N$ are scaled with (multiplied by) the same constant factors. All the algorithms are tested on an Intel Core™i7 processor (on a single core). The slope of the curves represent the relative complexity of the algorithms. As can be seen, the complexity of the SEED method is considerably less than the other algorithms which makes it a well-suited approach for decomposing large datasets.
Decomposition Error

In order to evaluate the adaptive column selection method to create $\mathbf{D}$ (Steps 1 and 2 in Algorithm 1), it is compared against a random column selection method which we refer to by random SEED. Fig. 8.6 (b) shows the normalized decomposition error as a function of $L$ for both methods. The OMP maximum sparsity level is set to 0.05. Increasing $L$ reduces the decomposition error. For the same $L$, the adaptive approach results in less decomposition error compared to the random one. Both methods are executed 10 times. The adaptive SEED, as opposed to random method, provides almost invariant decomposition error for different trials. This is because the adaptive SEED method carefully selects the columns such that the decomposition error is minimized.

Sparsity of Decomposition

Fig. 8.6 (c) shows the resulting density of $\mathbf{V}$ as a function of $L$ for random and adaptive methods. The OMP maximum decomposition error $\delta$ is set to 0.05. Increasing $L$ reduces the density of $\mathbf{V}$. Similar to the decomposition error, the adaptive SEED provides more consistent density results. For a larger $L$ ($> 100$), the density level shrinks to only a few elements. This is in part because more structures is present in $\mathbf{D}$ in that case.

8.6.3 Reconstruction Results

In this part we evaluate RankMap’s performance in solving the light field data reconstruction problem. In all the following experiments light field dataset (ii) is used.
Reconstruction Method

Several iterative gradient-based algorithms have been proposed to solve regularized least square equations. To solve Eq. 8.2 for $\ell_1$ penalty, the fast iterative shrinkage-thresholding algorithm (FISTA) is used. FISTA is a high performance solver with provably fast convergence guarantees [34]. The iterative update in FISTA fits into the form of Eq. 8.3 followed by a shrinkage-thresholding function.

Reconstruction Performance

We first apply adaptive SEED method for decomposing the dictionary corresponding to datasets (ii). Three different values of $L$ (240, 480, and 1000) are used in Algorithm 1. The OMP error $\delta$ is set to 0.1. Then, we evaluate FISTA on the decomposed data for both graph-based and matrix-based models. We also implement a tailored distributed MPI-based model to evaluate FISTA on the original datasets ($A$) using regular dense matrix representations. In the following, we refer to this model as baseline. In all the experiments, a batch of 10 noisy input patches is used as the input.

Figures 8.7 (a), 8.7 (b), and 8.7 (c) show runtime (of FISTA) versus normalized reconstruction error for different decomposition sizes. As can be observed, for a smaller $L$ FISTA converges faster. However, a larger $L$ results in a lower convergence error. The runtime to achieve the same reconstruction error (0.1) is orders of magnitude faster compared to the baseline (almost 2 orders of magnitude for $l = 240$ and an order of magnitude for $l = 480$ and $l = 100$). The matrix-based model is considerably faster for $L = 240$, however as $L$ increases the performances of both graph and matrix-based models become similar. This observation can be explained by the fact that for small $L$’s, the decomposition results are less sparse. Thus the overhead of the graph-based model to represent the edges makes it less efficient.
Figures 8.7 (d), 8.7 (e), and 8.7 (f) show the average density of the solution vectors (reconstructed coefficients or $x$). Applying FISTA on the decomposed data successfully results in sparser reconstructed coefficients compared to the baseline. Recall that the reconstruction objective is to concurrently minimize the reconstruction error and density of solution $x$.

### 8.6.4 Graph vs. Matrix-based Models

**Runtime Efficiency**

In this part we compare the performance of RankMap’s proposed vertex and matrix-based models for various synthetic decomposed data. The purpose of these evaluations is to determine the advantage of each of the models with respect to the structure of the data. In all the experiments, the iterative update in Eq. 8.3 is applied on a random input vector $y$. The experiments are done on an IBM iDataPlex computing cluster. Other than Fig. 8.6.4 (e), all the experiments are done on 4 nodes each with 12 cores. In all the figures, the runtime results for the dense matrix-based implementation (i.e., regular deployment of the decomposed matrices without using CCS format) are also provided to demonstrate the efficiency achieved by exploiting sparsity in $V$ through graph-based and sparse matrix-based models.

Figures 8.6.4 (a) and 8.6.4 (b) show the runtime performance for different $V$’s with fixed density-levels (set to 1%) and varying $L$’s. In the first figure, $V$ is block-diagonal and in the second figure it is uniformly filled. As $L$ increases, graph-based model performs better in comparison with the matrix-based model. This is because the communication overhead becomes more dominant for larger $L$’s and that the graph-based model provides more efficient communications. In particular, the graph-based model achieves its optimal communication bound when $V$ is block-diagonal. Note that although in the real-world light field dataset we obtained favorable recon-
struction performance with $L \ll M$, an insightful observation from these figures is that RankMap can be advantageous even with $l \geq m$ when $V$ is sufficiently sparse.

Fig. 8.6.4 (c) shows the performance for different (block-diagonal) $V$’s with fixed number of non-zeros (set to 100M). Thus as $L$ increases, the density-level of $V$ would decrease. The graph-based model’s performance is more consistent as $L$ increases. However, the matrix-based model’s performance degrades for larger $L$’s. This observation can also be explained due to the fact that the communication overhead of the matrix-based model is more affected by larger $L$’s.

Fig. 8.6.4 (d) shows the performance for a fixed $L = 500$ on block-diagonal matrices $V$ for varying densities of $V$. As density increases, the performance decreases in both models. However, the performance degradation in the graph-based model is worse due to the overhead of representing a large number of edges. Lastly, Fig. 8.6.4 (e) shows the scaling performance of the models for various number of processors. When the number of processors are less than 12, the computations are done on a single node. Thus the reverse scaling behavior while increasing the number of processors from 8 to 16 is due to the high overhead of the inter-node communication cost.

For comparison purposes, the scaling performance of the baseline dense $m = 1k$ by $n = 10M$ dataset (before decomposition) is provided. It can be seen that as the number of processors increases, the performance gap between different methods shrink. However, even with a large number of processors ($\geq 100$), the decomposed models perform up to 2 orders of magnitude better than the baseline.

These experiments provide insights on the use-case of each model. Depending on the structure of the decomposed data and the specifications of the platforms an appropriate model should be selected. A more systematic domain-specific approach for model selection is the subject of future work.
Table 8.2: *Comparison of memory usage (MB).*

<table>
<thead>
<tr>
<th>$L$</th>
<th>240</th>
<th>480</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graph-based</td>
<td>82</td>
<td>88</td>
<td>180</td>
</tr>
<tr>
<td>Sparse matrix-based</td>
<td>54</td>
<td>77</td>
<td>160</td>
</tr>
<tr>
<td>Dense matrix-based</td>
<td>226</td>
<td>453</td>
<td>944</td>
</tr>
<tr>
<td>Baseline</td>
<td>14400</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Memory Efficiency**

As discussed in Sec. 8.5, memory footprint is mainly determined by $nnz(V)$ and $ML$. Table 8.2 provides the total memory usage for representing the decomposed components for various $L$’s for light field dataset (ii). The memory usage of the baseline method for storing $A$ is 14.4GB which is up to two orders of magnitude more than that of decomposed data with $L = 240$. The memory usage for (regular) dense matrix-based representation of the decomposed data is also provided. As it is expected, the sparse matrix-based model results in more efficient memory performance compared to the dense model.
Figure 8.6: *Comparison of error and sparsity for adaptive and random sampling methods.* (a) Computational complexity vs. data dimension. (b) Decomposition error vs. Size of Factorization. (c) Relative density of $V$ vs. Size of Factorization.
Figure 8.7: Runtime comparisons for iterative updates for FISTA. Runtime vs. reconstruction error for (a) $L = 240$ (b) $L = 480$ (c) $L = 1000$. Runtime vs. density of solution $x$ for (d) $L = 240$ (e) $L = 480$ (f) $L = 1000$. 


Figure 8.8: Comparison of matrix-based and graph-based architectures. Runtime vs. $L$ for (a) a block diagonal and (b) a uniform $V$. Runtime vs. $L$ for $V$ with fixed sparsity level (c). Runtime vs. density of $V$ (d). Runtime vs. number of processors (e).
As modern datasets continue to grow, it has become evident that using a global model to capture all of the low-dimensional structure in a dataset is no longer possible. Thus, the development of new hybrid models that model different pieces of the data with different low-dimensional models is essential for squeezing out redundancies and providing a more compact description of “big data”. This thesis takes a necessary step towards providing new theory and methods for modeling massive datasets with unions of subspaces. The results of this thesis provide new insights into how to process big datasets that have hybrid structures, all the way from finding representative samples from the data, to dimensionality reduction, clustering, and finally for exploiting low-dimensional structure for data-aware processing.

The key ingredient underlying our approach to model, learn, and exploit union of subspace structure in data is to express the data in terms of a small but intelligently selected subset of the data. We refer to this notion of expressing the data with respect to itself as “self expression”. This thesis develops new theory and algorithms for learning and modeling data that rely on forming sparse self-expressive representations of collections of data. To do this in a scalable and efficient manner, this thesis focuses on the development of low-complexity methods for (i) selecting subsets of signals from
the data that provide good archetypes for the entire dataset and (ii) forming efficient sparse representations of the data in terms of the selected set of signals. The resulting approach for decomposition not only yields a scalable approach for dimensionality reduction that can be used in place of PCA and sparse PCA (SPCA) but can also be used to cluster datasets according to the subspace membership of the underlying signals. In conjunction with the development of new methods for sampling and clustering large datasets, in Chap. 7, we also developed a number of new theoretical results that demonstrate the utility of the proposed methods for dimensionality reduction and clustering.

In tandem with the new methods and theory developed for decomposing and clustering signals in unions of subspaces, In Chap. 8, we also introduced a host of tools for distributed processing of data that admit multi-subspace structure. We show that self-expressive decompositions can be used to quickly decompose and assess the structure in the data; this structure can then be leveraged to accelerate distributed matrix and graph-based computations on the dataset required in a host of regression and learning problems.

In many respects, hybrid models are still in their infancy. This is due to the fact that learning and exploiting hybrid structure is inherently more challenging than learning a global model for the data. This thesis provides new insights into the power of hybrid models and how self expression can be used to reveal hybrid structures in data. With these new results, it is our hope that hybrid modeling will continue to gain more traction in the signal processing and machine learning communities and be utilized in new settings throughout the computational sciences. To conclude, we pose a number of open questions and directions for future lines of research.
9.1 The Power of Self-Expression

This thesis highlights the utility of self expression for revealing and exploiting union of subspace structures in massive datasets. However, another important aspect of self-expressive decompositions is that the factorizations that result contain factors (samples) that lend themselves to interpretation. For this reason, self-expressive decompositions can potentially be utilized in settings where an human or domain expert must interpret the resulting factorization of the data. For instance, in computational biology applications, where each column of the data $X$ represents a gene expression profile from a particular person, the selected reference set $X_S$ contains a set of reference gene expression profiles that correspond to realizable expression levels. In contrast, factorizations provided by sparse PCA might consist of a mixture of different expression profiles and thus result in a set of unrealizable expression profiles that could not occur in nature. Clearly self expression provides a powerful paradigm for factorization that must be explored further. Understanding the settings in which self expression is possible is an interesting and open question.

9.2 "Data Driven" Sparse Approximation

The standard paradigm in signal processing and approximation theory is to compute a representation of a signal in a fixed and pre-specified basis or overcomplete dictionary. In most cases, the dictionaries used to form these representations are designed according to some mathematical desiderata. A more recent approach has been to learn a dictionary from a collection of data, such that the data admit a sparse representation with respect to the learned dictionary [44, 45].

The applicability and utility of self-expressive sparse recovery in subspace learning draws into question whether we can use endogenous sparse recovery for other tasks,
including inference, prediction, approximation and compression. A question that naturally arises is, “do we design a dictionary, learn a dictionary, or use the data as a dictionary?” Understanding the advantages and tradeoffs between each of these approaches in different applications is an interesting and open question.

The results of this thesis suggest that when the data lie on a union of subspaces, self expression provides a natural strategy to reduce the dimensionality of the dataset. However, understanding when sparse self expression can be used to reduce the dimensionality of datasets that admit other types of nonlinear structures is an open question.

9.3 Learning Block-Sparse Signal Models

Block-sparse signals and other structured sparse signals have received a great deal of attention over the past few years, especially in the context of compressive sensing from structured unions of subspaces [4, 5] and in model-based compressive sensing [6]. In all of these settings, the fact that a class or collection of signals admit structured support patterns is leveraged in order to obtain improved recovery of sparse signals in noise and in the presence of undersampling.

To exploit such structure in sparse signals—especially in situations where the structure of signals or blocks of active atoms may be changing across different instances in time, space, etc.—the underlying subspaces that the signals occupy must be learned directly from the data. The methods that we have described for learning union of subspaces from ensembles of data can be utilized in the context of learning block sparse and other structured sparse signal models. The application of subspace clustering methods for this purpose is an interesting direction for future research.
9.4 Beyond Coherence

While the maximum and cumulative coherence provide measures of the uniqueness of sub-dictionaries that are necessary to guarantee exact signal recovery for sparse recovery methods [38], the results of the geometric analysis in this thesis suggests that examining the principal angles formed from pairs of sub-dictionaries could provide an even richer description of the geometric properties of a dictionary. Thus, a study of the principal angles formed by different subsets of atoms from a dictionary might provide new insights into the performance of sparse recovery methods with coherent dictionaries and for compressive sensing from structured matrices. In addition, our empirical results in Section 6.5.1 suggest that there might exist an intrinsic difference between sparse recovery from dictionaries that exhibit sublinear versus superlinear decay in their principal angles or cross-spectra. It would be interesting to explore whether these two “classes” of dictionaries exhibit different phase transitions for sparse recovery.

9.5 Discriminative Dictionary Learning

While dictionary learning was originally proposed for learning dictionaries that admit sparse representations of a collection of signals [44, 45], dictionary learning has recently been employed for classification. To use learned dictionaries for classification, a dictionary is learned for each class of training signals and then a sparse representation of a test signal is formed with respect to each of the learned dictionaries. The idea is that the test signal will admit a more compact representation with respect to the dictionary that was learned from the class of signals that the test signal belongs to.

Instead of learning these dictionaries independently of one another, discriminative dictionary learning [95, 53], aims to learn a collection of dictionaries \( \{\Phi_1, \Phi_2, \ldots, \Phi_p\} \)
that are incoherent from one another. This is accomplished by minimizing either the spectral [95] or Frobenius norm [53] of the matrix product $\Phi_i^T \Phi_j$ between pairs of dictionaries. This same approach may also be utilized to learn sensing matrices for CS that are incoherent with a learned dictionary [96].

There are a number of interesting connections between discriminative dictionary learning and our current study of EFS from collections of unions of subspaces. In particular, our study provides new insights into the role that the principal angles between two dictionaries tell us about our ability to separate classes of data based upon their sparse representations. Our study of EFS from unions with structured cross-spectra suggests that the decay of the cross-spectra between different data classes provides a powerful predictor of the performance of sparse recovery methods from data living on a union of low-dimensional subspaces. This suggests that in the context of discriminative dictionary learning, it might be more advantageous to reduce the $\ell_1$-norm of the cross-spectra rather than simply minimizing the maximum coherence and/or Frobenius norm between points in different subspaces. To do this, each class of data must first be embedded within a subspace, a ONB is formed for each subspace, and then the $\ell_1$-norm of the cross-spectra must be minimized. An interesting question is how one might impose such a constraint in discriminative dictionary learning methods.

### 9.6 Greedy Feature Selection

While EFS provides a natural measure of how well a feature selection algorithm will perform for the task of subspace clustering, our empirical results suggest that EFS does not necessarily predict the performance of spectral clustering methods when applied to the resulting subspace affinity matrices. In particular, we find that while OMP obtains lower rates of EFS than BPDN on real-world data, OMP yields better clustering results on the same dataset. Understanding where this difference in
performance might arise from is an interesting direction for future research.

Another interesting finding of our empirical study is that the gap between the rates of EFS for sparse recovery methods and NN depends on the sampling density of each subspace. In particular, we found that for dense samplings of each subspace, the performance of NN is comparable to sparse recovery methods; however, when each subspace is more sparsely sampled, sparse recovery methods provide significant gains over NN methods. This result suggests that endogenous sparse recovery provides a powerful strategy for clustering when the sampling of subspace clusters is sparse. Analyzing the gap between sparse recovery methods and NN methods for feature selection is an interesting direction for future research.


[37] R. Rubinstein, M. Zibulevsky, and M. Elad, “Efficient implementation of the K-SVD algorithm using batch orthogonal matching pursuit,” *CS Technion*, p. 40, 2008. 3.5, 7.4.1, 8.2.1, 8.3.4


