HANDLING MISSING DATA IN HIGH-DIMENSIONAL SUBSPACE MODELING

By

Laura Kathryn Balzano

A DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

(ELECTRICAL AND COMPUTER ENGINEERING)

at the

UNIVERSITY OF WISCONSIN – MADISON

2012

Date of final oral examination: 5/10/2012

The dissertation is approved by the following members of the Final Oral Committee:
Robert Nowak, Professor, Electrical and Computer Engineering
Benjamin Recht, Assistant Professor, Computer Science
Barry Van Veen, Professor, Electrical and Computer Engineering
Jordan Ellenberg, Professor, Mathematics
Stark Draper, Assistant Professor, Electrical and Computer Engineering

To Mira and Miles Balzano, born weeks before the start of my graduate work,

to Emma Balzano, born half-way through,

and to James Lawrence Hall, born weeks before the completion of this dissertation.

I will continue to watch with wonder and joy as you grow.

Abstract

Big data are making a big splash, with everyone from bookstores to stock brokers, hospitals to libraries, ecologists to military generals looking to capitalize on data collection opportunities. As we digitize the information in every part of our lives, and spread automatic data collection capabilities throughout our homes, workplaces, networks, and roads, there is no doubt that the data deluge is here to stay. Along with massive data comes a new problem for statistical signal processing: updating our modeling tools in order to address issues with big data.

Big datasets are by definition massive, requiring computationally efficient techniques. Even more consequential is that the data quality is impossible to control. It is truly inevitable that there will be missing data, corrupted measurements, and large gaps in collection. Most classical statistical techniques implicitly assume that these issues have been "cleaned" away before modeling. This thesis takes a first step toward modeling the data as-is without heuristic preprocessing steps that may or may not retain the important information in a collection of measurements.

This thesis focuses in particular on data which are missing uniformly at random. It also focuses on a particular model, the subspace model. All of the fundamental principles of subspace estimation, such as residual vectors and the orthogonality principle, have counterparts for when data are missing. We present these, along with algorithms to estimate and track subspace models when data are missing. The algorithms operate in an online fashion and are computationally very efficient. We also present algorithms to estimate unions of subspaces, a generalization of the subspace model that can approximate modeling with non-linear manifolds.

Acknowledgements

In the course of pursuing my Ph.D., I have been so blessed to have many people giving me endless support and encouragement.

First and foremost I thank my parents, Ed and Linda Balzano, and my brother Geoff, for their support from day one. My parents always understood that I loved pursuing education and knowledge, and they gave me every resource to help me do that, including their full hearts. And I can still remember the look on my brother's face in the moment when I clarified that I would be pursuing not a Master's degree but a Ph.D. "Really?" he asked. "Why?" And yet from the next moment he supported me every step of the way. His question served to always remind me to think about why.

I want to thank Professors Don Johnson and Rich Baraniuk at Rice University for showing me the true beauty of signal processing and for seeing potential in me. I want to thank Ed Knightly and Violeta Gambiroza for showing me for the first time how fun and rewarding research could be. Thank you to John Treichler, the CTO of my employer Applied Signal Technology, for working with me on interesting questions and for giving me a practical perspective on telecommunications.

My friends have been a constant source of support. My friends Josie Row, Kate Soper, Angi Chau, and Jenny Tsui deserve special thanks for all the conversations about life, graduate school, and career choices. And thanks especially to Sheila Hall, my best friend of over a decade, for being the most amazing listener, cheerleader, and friend.

My Ph.D. wouldn't be what it is without my exposure to the vast open problem of messy datasets during my Master's degree at UCLA. My Master's advisor, Mani Srivastava, so freely shared his knowledge, criticisms, and vision with me; I certainly wouldn't have the same perspective without him. Thank you to Deborah Estrin as well, the head of the Center for Embedded Network Sensing, for being such

an excellent mentor and leader. And with warmth I also want to thank Nabil Hajj Chehade, Sadaf Zahedi, Sahar Sedighpour, Murat Ozguc, and Bhama Vemuru, my peers without whom I would not have learned (nor laughed) nearly as much in my classes at UCLA. Thank you also to Nabil for rooting me on at the Ph.D. finish line with many reminders to keep writing and practice talks via Skype.

At Wisconsin I have been fortunate to interact with so many sharp people who ask great challenging questions and are also friendly and fun. I want to thank my math teachers, Andreas Seeger and Gloria Mari-Beffa, for their brilliance and encouragement. Thank you to my committee members for thoughtful and insightful discussions: Barry Van Veen, Stark Draper, Jordan Ellenberg. Thank you to Steve Wright, Vincent Tan, Jun He, and Arthur Szlam for being excellent collaborators. Among my collaborators I send thanks especially to Matthew Roughan, whom I met at a critical point in my Ph.D. He asked me some simple questions with a deep curiosity, reminding me what I love about mathematics and signal processing.

Now for the biggest thank yous. Annie Dillard said, "How you spend your days is how you spend your life." These people have been around me daily for the past four years. The first thank you goes to Ben Recht, my CS advisor whose vast technical knowledge has been an amazing source of education for me. Ben is a caring and thoughtful advisor, and I thank him for his advice and friendship. The next goes to my academic siblings who provided daily support at UW: my senior siblings Aarti Singh, Rui Castro, Jarvis Haupt, Brian Eriksson, and Waheed Bajwa, as well as my junior siblings Matt Malloy, Nikhil Rao, Gautam Dasarathy, Shirzad Malekpour, and Kevin Jameison. The intellectual climate that has been fostered in the Nowak group is truly unprecedented; I can remember enlightening conversations with each person that allowed me to make breakthroughs big and small.

Finally, and sincerely, I want to thank my advisor, Rob Nowak. Ever since the first days when he taught me digital signal processing at Rice University, Rob has been an outstanding mentor, role-model, and friend. His enthusiasm for research is inexhaustible and infectious. None of us know how he does it, but with his students and collaborators he manages to foster the most wonderful environment for

intellectual discovery. I think it has something to do with his willingness and capacity to support his students' growth and independence– a capacity which is truly unrivaled. If I can have half as much fun in my job as Rob has, I will consider myself successful. I thank Rob for everything he's done for me, and I look forward to many years of collaborations in the future.

Contents

Chapter 1

Introduction

Modern signal processing applications present two main challenges, precipitated by the explosion of data collection in science, engineering, and business as well as in society at large. The first challenge is that of *massive data*: data are collected on every aspect of a system and are sometimes collected nearly continuously. The second challenge, and the focus of this thesis, is that of *missing data*: nearly every data collection effort has issues with missing data and the impossibility of collecting every measurement of interest.

Measurement systems are almost never deployed to simply collect measurements but instead deployed to perform inference, estimation, and prediction, or to aid in analysis and decision making. Thus we focus not on data imputation but on *modeling*: we wish to estimate model parameters despite the fact that data are missing. In particular, we focus on estimating or somehow leveraging low-dimensional subspace models. While addressing this task, we insist on mindfulness to computational issues that come with massive data; at the same time we intend to use the massive data to our benefit by leveraging redundancy that is nearly inevitable in heavily measured systems.

In this thesis, the datasets we consider are real valued and take the form of vectors or matrices. For example, if n sensors are deployed in a sensor network to take measurements every minute, then at each time snapshot we can consider a length-n vector of the sensors' measurements, and if we accumulate T time snapshots we can consider an $n \times T$ data matrix of the data from the sensors.

We focus on estimating subspace models with missing data; a little more precisely, we mean the following. A data value in the dataset is missing if we do not have the measurement, but we do know

its location in the data matrix– e.g., the measurement from sensor i at time t is unknown. Then when considering the collection of underlying complete data vectors, we are interested in finding a low-dimensional subspace $\mathcal{S} \subset \mathbb{R}^n$ which models these vectors well. In other words, we are interested in finding a collection of d vectors, $d < n$, such that the vectors in our data matrix are well-approximated by a linear combination of these d vectors.

We first motivate the issues of missing (Section 1.1) data. We then motivate subspace models (Section 1.2), the modeling focus of this thesis.

1.1 Missing Data

Dealing with missing data is a central issue for modern problems in statistical signal processing and machine learning. At the University of California, Irvine (UCI) database for machine learning data sets [41], users upload data sets and documentation for others to test their machine learning algorithms. One piece of meta-data about each data sets is whether or not there are missing values. Approximately 25% of the data sets have a "Yes" in this category[1]. Not including the data sets which are marked "N/A", approximately 40% of the data sets have missing values. In our own research, we have dealt with data from the UW Computer network, the Center for Embedded Networked Sensing (CENS) Cold Air Drainage transect[2], data capturing signal strength of the public wifi in Madison, Wisconsin, and data collected on the climate and ecosystem of the lakes in the state of Wisconsin. In all of these data sets, data were missing.

Taking another perspective on this issue, sometimes it is not that data go missing, but instead that all the data cannot possibly be collected. In fact the problem of interest may be to infer a subset of the data points that we don't have. For example in recommendation systems like Amazon or Netflix, the company is interested in predicting what products or movies you would like most, given your

[1]These percentages were calculated out of all available 199 data sets on April 27, 2011. To the question "Missing values?", 44 said yes, 69 said no and 75 said N/A.

[2]The Center for Embedded Networked Sensing at UCLA deployed this set of sensors at James Reserve in 2004. (Check that)

preferences so far. Helping you wade through the vast unknown possibilities is exactly their business model, and to do that these companies try to model your preferences despite very incomplete user profiles.

Classical signal processing has handled missing data with the EM algorithm [35]. This approach is appropriate both when actual measurements are missing and when estimating hidden, unmeasured variables makes inference much easier. The EM approach is very general and widely applied. However, this approach hinges on the idea that imputing the missing data is required in order to make appropriate inferences. The work in this thesis instead considers model estimation without a focus on imputing missing data (though this is often possible as a side-effect). More importantly, in general the EM algorithm attempts to maximize a likelihood function, and it can only be guaranteed to converge to a local maximum. The work in this thesis focuses on algorithms with provable approximation of the global optimum point.

1.1.1 A Note on How Data go Missing

Data may go missing in a variety of ways. In this thesis, we focus on data which are missing uniformly at random. This means that:

1. Which data are missing or observed is in no way dependent on the actual values themselves;
2. Which data are missing or observed is in no way dependent on which other values are missing or observed;
3. The probability that any one data point is missing is equal to that of any other data point being missing.

Though this is not explored in this thesis, relaxing all three of these assumptions is of great interest. We briefly discuss other models in the future work Section 8.2.

1.2 Subspace Models

The subspace model is used throughout engineering and science as a simple but powerful predictor. The assumption underlying this model is that each data vector is a linear combination of a small number of principal component vectors or singular vectors.

Intuitively, this is a reasonable model whenever the data vectors are thought to be generated as a combination of a small number of factors– for example some number of flows through a network determine the traffic flow across a particular link, or some small number of characteristics factor in to one's assessment of a particular product. This model has also proved powerful simply when coordinates of a data vector are expected to be correlated, like temperature and humidity values across a valley at a nature preserve [4].

Best approximating subspaces can be estimated in the presence of additive noise or small corruptions. With a batch of complete data, the best d-dimensional subspace approximation to this batch, for any d, can be found with the Singular Value Decomposition [44]. Part of our contribution focuses on algorithms that can provably approximate this best d-dimensional subspace when data are missing or corrupted.

In applications where measurements are periodically taken at regular time intervals, the dataset grows with every new measurement. The underlying best d-dimensional subspace will change unless the data are coming from a single d-dimensional subspace for all time. Consider the example in Figures 1. This plot was generated using two weeks of data collected on the University of Wisconsin (UW) computer network. Byte counts were collected every 5 minutes on 67 routers in the network, both incoming and outgoing, for 134 measurements. In the figure, we have windowed a portion of the data, for different window sizes, and taken the SVD. Then we found the best-fit subspace using the top singular vectors. The plot shows the number of singular vectors needed to retain 90% of the signal energy over time. The fact that that number increases as the window grows implies that the subspace is changing over time; the fact that each line is (somewhat) flat implies that the subspace dimension

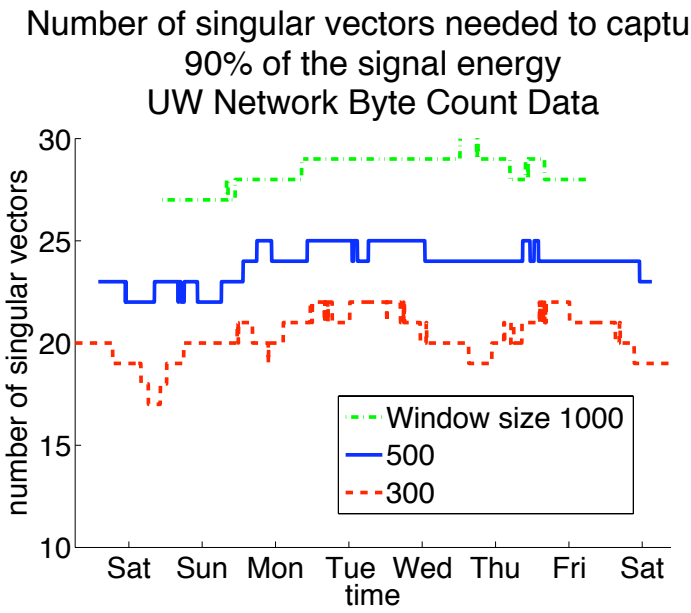


Figure 1: Window size refers to the size of a batch of vectors for which the SVD is taken.

remains the same over time.

The same type of plot is shown in Figure 2 for our CENS cold air drainage data, ozone data from the UCI machine learning repository, data on the number of births in each country around the world, and stock prices of the S&P 500. In all but the ozone data, the subspace varies: over time, or over the size of the window of data to which we attempt to fit our model, or both.

1.3 Summary and Outline

In this thesis we take many perspectives on estimating the low-dimensional subspace model when data values are missing. In Chapter 3, we show that the fundamental building blocks for subspace estimation– the projection operator, the orthogonality principle– have counterparts when data values are missing. We explore this theory in detail and provide simulations for further intuition. We also consider the problem of subspace detection with missing data. In Chapter 4, we propose the algorithm GROUSE, Grassmannian Rank-One Update Subspace Estimation, to perform subspace estimation when data are

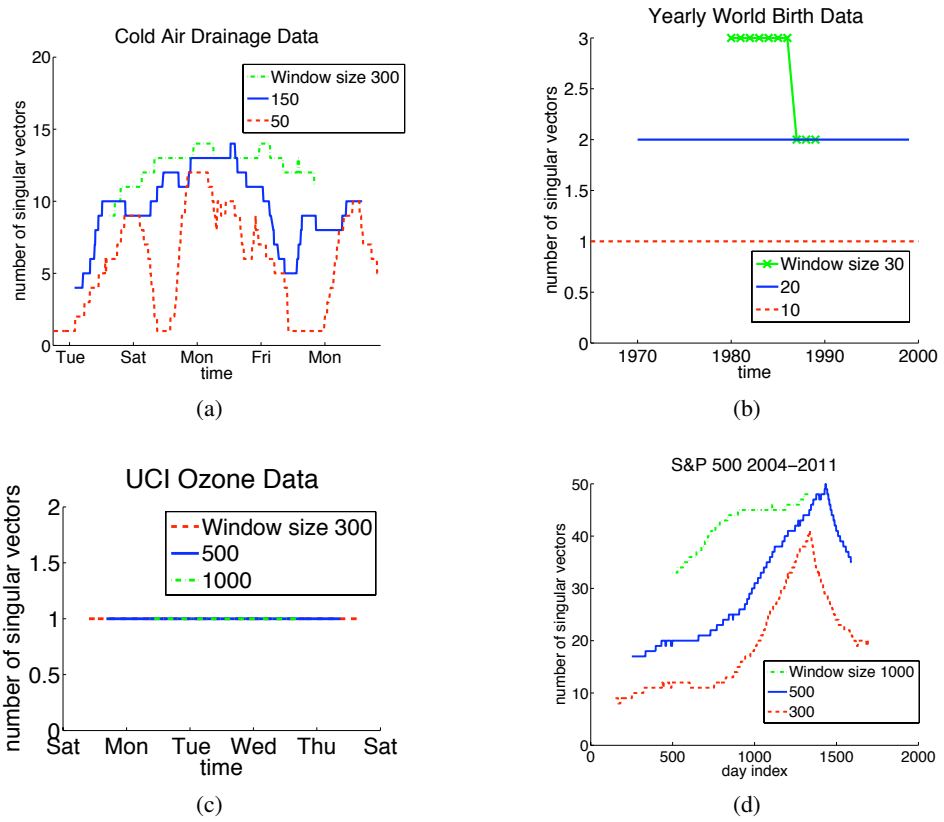


Figure 2: Note the variability in the number of singular vectors needed across the different window sizes. The variability among a single window size is also important, but we will not address that issue in this work.

missing. We examine GROUSE's performance in theory and in simulation. In Chapter 5 we propose two algorithms for estimating unions of subspaces with missing data. With the first we have a proof of success; with the second we have a vast speed improvement over the first which we show with simulation results. In Chapter 6 we discuss the use of GROUSE for tracking subspaces. We also propose algorithms for tracking subspaces robustly, in the presence of corrupted data, and evaluate their performance in simulation. Finally in Chapter 7, we derive an optimization problem for selecting columns from a matrix with missing data, and propose an algorithm to solve it.

Chapter 2

Survey of Related Work

Since there are a great many motivating applications for subspace modeling, there is a great deal of related work. Subspace models have applications in medical [2] and hyperspectral [60] imaging, communications [73], radar [84], and anomaly detection [97], source localization and target tracking in radar and sonar [59], and multistatic radar [30]. They are useful in computer vision to do background subtraction [106] and object tracking [32, 101] and to represent a single scene under varying illuminations [10, 78]. Certain patterns of computer network traffic including origin-destination flows can be well represented by a subspace model [61]. Environmental monitoring of soil and crop conditions [51], water contamination [80], and seismological activity [105] have all been demonstrated to be efficiently summarized by very low-dimensional subspace representations. Subspace representations of signals from sensor networks have been shown to be useful for sensor calibration [4]. Subspace tracking algorithms could enable rapid detection of traffic spikes or intrusions in computer networks or could provide efficiency gains in managing energy consumption in a large office building [42].

Table 1 shows an organization of our contributions in relation to research in the area. Here we give a survey of related work in Subspace Detection and Tracking, Subspace Estimation with missing data or Low-Rank Matrix Completion, and Robust PCA.

2.1 Subspace Detection

Testing whether a signal lies within a given subspace is a problem arising in a wide range of applications including medical [2] and hyperspectral [60] imaging, communications [73], radar [84], and anomaly

Problem	**no missing data**	**missing data**
Subspace Detection	[92, 93]	Section 3.4
Subspace Assignment (Multiple Hypothesis Testing)	[43]	Section 3.3
Subspace Estimation batch	SVD [44], PCA [55], Gradient methods [38, 96]	Low-Rank Matrix Completion [22, 85]
Subspace Estimation incremental	Incremental SVD [90, 19], Gradient methods [31]	Chapter 4
Estimating Unions of Subspaces	[56, 102, 63, 29, 112]	Chapter 5, [47, 104]
Subspace Tracking	Gradient methods [31, 110, 109]	Section 6.1
Robust PCA batch	[65, 36, 108]	[27, 24]
Robust PCA incremental	[71, 64, 83]	Section 6.2
Column Subset Selection	[15, 49]	Chapter 7

Table 1: An organization of the contributions of this dissertation and representative related work.

detection [97]. The classical formulation of this problem is a binary hypothesis test of the following form. Let $v \in \mathbb{R}^n$ denote a signal and let $x = v + w$, where w is a noise of known distribution. We are given a subspace $S \subset \mathbb{R}^n$ and we wish to decide if $v \in S$ or not, based on x. Tests are usually based on some measure of the energy of x in the subspace S, and these 'matched subspace detectors' enjoy optimal properties [92, 93].

Compressed matched subspace detection [81] provides a way to perform matched subspace detection with fewer than full measurements; this work differs from the current thesis in that it uses compressed measurements, or random linear combinations of all vector entries.

2.2 Subspace Tracking

Comon and Golub [31] give an early survey of adaptive methods for tracking subspaces, both coming from the matrix computation literature, including Lanczos-based recursion algorithms, and gradient-based methods from the signal processing literature.

2.2.1 Linear Algebraic Methods

Many papers have adapted QR and SVD factorizations for the adaptive context. Bischof and Shroff [14] uses ideas from the rank-revealing QR decomposition. The authors in [75] provide an $O(n^2)$ algorithm for updating the SVD for subspace tracking problems. This algorithm extends a QR update to an SVD update with diagonalization and orthogonalization. Another linear algebraic method in [98] is also an $O(n^2)$ algorithm and finds a representation "in between" QR and SVD, which they call the URV decomposition and has some benefits from both decompositions. The work in [14] uses the rank-revealing QR factorization and incremental condition estimation [13] in order to take a subspace estimate of data $[v_1, v_2, \ldots, v_n]$ and transform it to an estimate for the next window, $[v_2, \ldots, v_n, v_{n+1}]$, again $O(n^2)$. Finally in [19], an algorithm is presented to make modifications, one column at a time, to the thin SVD of a strictly rank-d $n \times n$ matrix in $O(n^2 d)$ time.

2.2.2 Descent-based methods

Initial work in signal processing for subspace tracking was aimed at estimating from data the largest eigensubspace for a signal covariance matrix. This is useful, for example, in direction-of-arrival (DOA) estimation: the well-known work in [89] introduces ESPRIT, a parameter estimation algorithm that estimates the DOA of plane waves emanating from a target and being received by a sensor array. It was a follow up to the MUSIC algorithm [94], and it gains computational efficiency for a slight tradeoff in generality of sensor array design. Both MUSIC and ESPRIT find vectors which are at the intersection of the signal subspace and the "array manifold" or the set which includes the steering vectors.

Yang and Kaveh [110] introduced an approach that is not based on batch estimation, thus making it more suitable for adaptive estimation of the signal subspace and covariance matrix. Their approach, like ours, is based on instantaneous gradient. This work was followed by [72] who presented a similar algorithm based on Newton's method. Both of these algorithms find the best-fit eigensubspace using the Rayleigh quotient as their cost function, as opposed to the Frobenius norm cost function that we

have addressed with our work.

Smith [38, 96, 37] thoroughly pursued conjugate gradient descent methods on the Grassmannian for solving the subspace tracking problem again using the Rayleigh quotient. In [38] the authors give a very careful definition of the problem, giving a nice survey comparing the applicability of various approaches. In [37] is an extensive list of subspace tracking references.

Projection Approximation Subspace Tracking, introduced in [109], is an RLS-type subspace tracking algorithm developed from the LMS algorithm we have discussed so far. The PAST algorithm adds exponential weighting to the cost function, so that data far in the past do not have as much an impact on the current subspace estimate as recent data.

Dimension estimation is an important problem for subspace tracking. In [82] the authors develop a rank estimation approach with asymptotic minimax optimality. Future work will be to investigate such a technique for GROUSE as well as cross-validation techniques that have been applied to the similar problem of sparsity estimation in compressed sensing [107].

2.3 Low-Rank Matrix Completion

Here we survey the current state-of-the-art low-rank matrix completion algorithms. In this section only, we use the notation that M is the true underlying matrix, perhaps with some noise added, and Ω is the index set of entries observed; so M_Ω are the observed values. Low-rank matrix completion then attempts to solve the problem

$$\begin{aligned} \min_{Y} \quad & \text{rank}(Y) \\ \text{subject to} \quad & \|M_\Omega - Y_\Omega\|_F^2 \leq \epsilon \end{aligned} \tag{2.1}$$

where $\|\cdot\|_F$ refers to the Frobenius norm. The value of ϵ can be zero in the noise-free case, implying equality on the observed entries. It has been shown [22, 85] that under incoherence assumptions on the

singular vectors of the true matrix M, the following nuclear-norm minimization problem is equivalent as long as enough observations are observed:

$$\begin{aligned} \min_{Y} \quad & \sum_{i=1}^{n} \sigma_i(Y) =: \|Y\|_* \\ \text{subject to} \quad & \|M_\Omega - Y_\Omega\|_F^2 \leq \epsilon \end{aligned} \tag{2.2}$$

where $\sigma_i(Y)$ is the i^{th} singular value of Y; i.e. the nuclear norm is the sum of the singular values. The problem (2.2) can be solved with a semi-definite program [22].

FPCA [68], or Fixed Point Continuation with Approximate SVD, is an algorithm for solving the nuclear-norm regularized problem $\|M - Y\|_F^2 + \lambda\|Y\|_*$; there exists a regularization parameter λ such that this problem is equivalent to (2.2). FPCA uses continuation to find a good λ; starting with a large value of λ gives a good starting point for solving the next SDP with a smaller λ.

An accelerated proximal gradient algorithm for solving the nuclear-norm regularized linear least squares problem is given in [99]. This algorithm is referred to as NNLS (Nuclear-Norm Least Squares) or APGL (Accelerated Proximal Gradient with Linesearch). Experiments in [99] show that APGL outperforms FPCA in both number of algorithm iterations as well as matrix completion error, which is our experience as well.

The authors of [21] seek to approximately yet efficiently solve the problem (2.2). An iteration of their Singular Value Thresholding (SVT) algorithm sets entries of Y_Ω closer to those of M_Ω, and then performs an SVD and a non-linear shrinkage operation on the singular values. This algorithm requires less memory and computational resources than FPCA and APGL.

ADMiRA [62] is a matrix completion algorithm built on the connections between compressed sensing and low-rank matrix completion; specifically, ADMiRA extends CoSaMP [76] to the problem of matrix rank minimization.

Jellyfish [87] is a parallel incremental algorithm that breaks the low-rank matrix completion problem (and a more general class of problems) into partitions so that each can be solved separately. Many

parallel programs which write to the same memory require that piece of memory be locked by one parallel process at a time, meaning the other processes will stall while waiting for that memory to become available. A major benefit of Jellyfish is that it does not require memory locks to be made on the gradient update, which is being updated simultaneously by each parallel process.

Another parallel algorithm, DFC [69] or Divide-Factor-Combine, also divides a low-rank matrix completion problem into several smaller problems. This work leverages some of what is known about random column and row sampling for low-rank matrix approximation in order to recombine the results of the smaller problems.

Since algorithms for low-rank matrix completion often search for the column and row space for M, it is natural to look at algorithms constrained to the Grassman manifold, or the space of all d-dimensional subspaces. Both the OptSpace [58] and SET [33] algorithms do exactly that. OptSpace [58] searches for both column and row space by performing gradient descent on the Grasmannian. They prove their algorithm converges, if it starts from a suitable starting point which is derived in a straightforward manner from the observed entries M_Ω. The SET, or Subspace Evolution and Transfer, algorithm [33] searches for either the column or the row space using gradient descent, while also detecting problematic points on the manifold and using a "transfer" procedure to find a new point from which to conduct the search.

By approaching matrix completion as a column space identification problem, our algorithm GROUSE (Chapter 4) can be applied to solve the low-rank matrix completion problem. GROUSE finds the subspace to minimize the error $\|M - Y\|_F^2$, and naturally results in a low-dimensional subspace because the search is constrained to the Grasmannian. GROUSE outperforms all of the above algorithms in computational efficiency, often by an order of magnitude, while performing competitively in terms of estimate error.

One reason why GROUSE is so computationally efficient is because it avoids the computation of the SVD altogether. Each iteration of GROUSE for completion of an $m \times n$ matrix of rank d requires

only $O(md + |\Omega|d^2)$ flops, where $|\Omega|$ is the number of observations per column. If the number of iterations is a constant multiple of the number of columns of the matrix as we conjecture, then the overall computational requirement is $O(nmd + n|\Omega|d^2)$.

In contrast, with full data the computation of a full SVD of an $m \times n$ requires $O(mn^2 + n^3)$ flops [44]. More comparable is the the thin-SVD of [19]. Again with full data, if the rank d of the matrix is restricted to $d < \sqrt{\min(m, n)}$ then the algorithm in [19] can be used to compute the thin-SVD in $O(nmd + nd^3)$ flops. For GROUSE with full data, we conjecture that we need only a constant multiple of the dimension in order to converge, resulting in computation time of $O(md^2 + nd^3)$ for the left singular vectors only.

2.4 Robust Principal Component Analysis

Principal Components Analysis [55] is a critical tool for data analysis in many fields. Given a parameter d for the number of components desired, PCA seeks to find the best-fit (in an l^2 norm sense) d-dimensional subspace to data; in other words, it finds the best d vectors, the principal components, such that the data can be approximated by a linear combination of those d vectors.

The residuals of an l^2-norm error function will be Gaussian distributed. Therefore, even with one outlier data point, the principal components can be arbitrarily far from those without the outlier data point [54]. Modern data applications– such as those in sensor networks, collaborative filtering, video surveillance or the network monitoring example just given– will all experience data failures that result in outliers. Sometimes the outliers are even the signal of interest, as in the case of network anomaly detection or identifying moving objects in the foreground of a surveillance camera.

A good deal of research is therefore focused on Robust PCA, including [27, 24]. Recent work focuses on a problem definition which seeks a low-rank and sparse matrix whose sum is the observed data. The majority of algorithms use SVD (singular value decomposition) computations to perform Robust PCA. The SVD is too slow for many real-time applications, and consequently many online

SVD and subspace identification algorithms have been developed, as we discussed in Section 2.2. We are therefore motivated to bridge the gap between online algorithms and robust algorithms with the work in Chapter 6, Section 6.2.

2.4.1 Robust Subspace Tracking

The work of [71] addresses the problem of robust online subspace tracking. They focus on the problem where outliers are found in a fraction of vectors (that is, some vectors have no outliers), though they do remark that this can be extended to handle the case where outliers are sparse in every vector. They have a very nice proposition relating l^0-(pseudo)norm minimization to the least trimmed squares estimator.

We note here that GRASTA, the algorithm in Section 6.2, differs from [71] in that it directly focuses on the case where every vector may have outliers, it operates on the Grassmannian for greater efficiency, and it can handle missing data. A comparison to [71] is a subject of future investigation.

More online algorithms for robust PCA can be found in [64, 83].

Chapter 3

Subspace Projection with Missing Data

Let $\mathcal{S} \subset \mathbb{R}^n$ be a d-dimensional subspace spanned by the columns of the orthonormal matrix $U \in \mathbb{R}^{n \times d}$; the projection operator onto $\mathcal{S}$ is thus $P_{\mathcal{S}} = UU^T$. Let $v \in \mathbb{R}^n$ be an arbitrary vector in $\mathbb{R}^n$. Then v can be written as the direct sum of its *in-subspace* component $P_{\mathcal{S}}v$ and its *out-of-subspace*, *residual*, or *orthogonal* component $(I - P_{\mathcal{S}})v$. By the orthogonality principle ([50], p 534), the inner product $\langle P_{\mathcal{S}}v, (I - P_{\mathcal{S}})v \rangle = 0$.

We start by looking at what happens to these fundamental facts, the building blocks for subspace modeling, when the data vectors are incomplete. The contributions of this chapter, and in fact the whole thesis, revolve around what we call the *incomplete data projection*: projecting an incomplete vector only onto the coordinates of the subspace corresponding to the observed coordinates of the vector.

In this chapter, we build up the foundation for subspace projections with missing data in Section 3.1. We look at the multiple hypothesis testing problem for subspaces in Section 3.3 and subspace detection in Section 3.4.

3.1 Problem Formulation

Again let $\mathcal{S} \subset \mathbb{R}^n$ be a d-dimensional subspace spanned by the columns of the orthonormal matrix $U \in \mathbb{R}^{n \times d}$ and let $v \in \mathbb{R}^n$ be an arbitrary vector in $\mathbb{R}^n$. Let $\Omega \subset \{1, \ldots, n\}$ be the indices of v that are observed. In most of the theory of this dissertation, we assume that some number of observations are chosen uniformly *with replacement* to constitute Ω, so therefore Ω is actually a *multiset* and not

a subset of $\{1, \ldots, n\}$. Lemma 5.1 in Chapter 5, Section 2.3 in [25], and Proposition 3.1 in [85] all address translation between probability results various types of random sampling.

Let v_Ω be the vector of dimension $|\Omega| \times 1$ comprised of the elements v_i, $i \in \Omega$, ordered lexigraphically, possibly with repetitions; here $|\Omega|$ denotes the cardinality of Ω.

Definition 1. *The* incomplete data projection residual *is defined as*

$$v_\Omega - P_{\mathcal{S}_\Omega} v_\Omega \tag{3.1}$$

where Ω as a subscript indicates restriction to the rows indicated by Ω, and

$$P_{\mathcal{S}_\Omega} = U_\Omega \left(U_\Omega^T U_\Omega\right)^\dagger U_\Omega^T = U_\Omega U_\Omega^\dagger \tag{3.2}$$

is the projection operator onto the observed coordinates, where † denotes the pseudoinverse.

In this section we will show that this incomplete data residual preserves both the *magnitude* (Theorem 3.1) and *angle* (Corollary 3.2.3) of the complete data residual $v - UU^T v$, as long as there are enough observations, where "enough" depends on the dimension of the subspace and the coherence of the subspace and the given vector. As a result of Theorem 3.1, we show the orthogonality principle is maintained (Corollary 3.2.2) and we can bound the norm of the projection itself (Corollary 3.2.2).

3.1.1 Weights of the projection of an incomplete vector

In order to build intuition about the incomplete data residual, we make the following initial observation for the special case of $v \in \mathcal{S}$.

Lemma 3.1. *Assume $x \in \mathcal{S}$ and $U_\Omega^T U_\Omega$ is invertible. Then the weights, or the basis projection coefficients, of the complete-data and incomplete-data projections onto $\mathcal{S}$ are equal:*

$$U^T x = \left(U_\Omega^T U_\Omega\right)^{-1} U_\Omega^T x_\Omega \; .$$

Proof. Call $a_1 = U^T x$. Since $x \in \mathcal{S}$, $Ua_1 = x$, and a_1 is in fact the unique solution to $Ua = x$. Now consider the equation $U_\Omega a = x_\Omega$. The assumption that $U_\Omega^T U_\Omega$ is invertible implies that $a_2 = \left(U_\Omega^T U_\Omega\right)^{-1} U_\Omega^T x_\Omega$ exists and is the unique solution to $U_\Omega a = x_\Omega$. However, $U_\Omega a_1 = x_\Omega$ as well, meaning that $a_1 = a_2$. □

As a consequence of the lemma, we have equality of the residuals, $(x - UU^T x)_\Omega = x_\Omega - U_\Omega U^T x = x_\Omega - P_{\mathcal{S}_\Omega} x_\Omega =$ the all-zeros vector. Thus also, $\|x_\Omega - P_{\mathcal{S}_\Omega} x_\Omega\| = \|x - UU^T x\| = 0$ when $x \in \mathcal{S}$.

3.1.2 Discussion of Incoherence

When a vector is subsampled, there is a risk that important entries of that vector will be missed. As an extreme example, suppose we subsample a vector $v = [1 \;\; 0 \ldots 0]^T$. Unless we sample the first entry, then without prior knowledge about the signal structure we will believe from our samples that the vector is all zero. Consider another situation where the subspace $\mathcal{S}$ is spanned by the first d euclidean basis vectors:

$$U = \begin{bmatrix} 1 & 0 & & 0 \\ 0 & 1 & & \vdots \\ 0 & 0 & & 0 \\ \vdots & \vdots & \cdots & 1 \\ 0 & 0 & & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & & 0 \end{bmatrix} \leftarrow d$$

Thus, the projection of a vector v onto $\mathcal{S}$ simply selects the first d components. If those components of the vector are not observed, then the projection operator of 3.2 will be undefined, as $\left(U_\Omega^T U_\Omega\right)$ will not be invertible, and the problem $w = \arg\min_a \|v_\Omega - U_\Omega a\|_2^2$ will not have a unique solution.

In order to develop general theory that holds in such a scenario, we must have a way to quantify

how much information each sample of v provides.

Definition 2. *We define the* coherence $\mu(\mathcal{S})$ *of a* d*-dimensional subspace* $\mathcal{S} \subset \mathbb{R}^n$ *to be:*

$$\mu(\mathcal{S}) := \frac{n}{d} \max_j \|P_{\mathcal{S}} e_j\|_2^2 .$$

The quantity $\mu(\mathcal{S})$ measures the maximum magnitude attainable by projecting a standard basis element onto $\mathcal{S}$. We use the standard basis because we are considering subsampling vectors entry-wise, so our vectors or the subspace must be incoherent with respect to this sampling operation. Note that $1 \leq \mu(\mathcal{S}) \leq \frac{n}{d}$. The minimum $\mu(\mathcal{S}) = 1$ can be attained by looking at the span of any d columns of the discrete Fourier transform. Any subspace that contains a standard basis element will maximize $\mu(\mathcal{S})$. For a vector z, we let $\mu(z)$ denote the coherence of the subspace spanned by z. By plugging in the definition, we have

$$\mu(z) = \frac{n\|z\|_\infty^2}{\|z\|_2^2} .$$

This parameter $\mu(\cdot)$ allows us to make generic statements about the incomplete data residual. If the coherence parameter of a vector v is near 1, then every sample of v is in some sense equally informative; if the coherence parameter of a subspace $\mathcal{S}$ is near 1, then every coordinate dimension is in some sense equally representative of the entire subspace.

3.2 Theoretical Analysis

3.2.1 Residual Magnitude Estimation from Incomplete Data

To state our first theorem, write $v = x + y$ where $x \in \mathcal{S}$ and $y \in \mathcal{S}^\perp$. Let the entries of v be sampled uniformly with replacement. Again let Ω refer to the set of indices for observations of entries in v, and denote $|\Omega| = m$. Given these conventions, we have the following result, which shows that the magnitude of the incomplete data residual is near that of the complete data residual, when scaled by the fraction of measurements.

Theorem 3.1. *Let $\delta > 0$ and $m \geq \frac{8}{3} d\mu(\mathcal{S}) \log\left(\frac{2d}{\delta}\right)$. Then with probability at least $1 - 4\delta$,*

$$\frac{m(1-\alpha) - d\mu(\mathcal{S})\frac{(1+\beta)^2}{(1-\gamma)}}{n} \|v - P_{\mathcal{S}} v\|_2^2 \leq \|v_\Omega - P_{\mathcal{S}_\Omega} v_\Omega\|_2^2$$

and

$$\|v_\Omega - P_{\mathcal{S}_\Omega} v_\Omega\|_2^2 \leq (1+\alpha)\frac{m}{n}\|v - P_{\mathcal{S}} v\|_2^2$$

where $\alpha = \sqrt{\frac{2\mu(y)^2}{m} \log\left(\frac{1}{\delta}\right)}$, $\beta = \sqrt{2\mu(y) \log\left(\frac{1}{\delta}\right)}$, and $\gamma = \sqrt{\frac{8d\mu(\mathcal{S})}{3m} \log\left(\frac{2d}{\delta}\right)}$.

This result is proved in Section 3.6. It uses the following three lemmas whose proofs can also be seen in Section 3.6.2; we state them here, as two of them are used in further corollaries.

Lemma 3.2. *With the same notations as Theorem 3.1,*

$$(1-\alpha)\frac{m}{n}\|y\|_2^2 \leq \|y_\Omega\|_2^2 \leq (1+\alpha)\frac{m}{n}\|y\|_2^2$$

with probability at least $1 - 2\delta$.

Lemma 3.3. *With the same notations as Theorem 3.1,*

$$\|U_\Omega^T y_\Omega\|_2^2 \leq (\beta+1)^2 \frac{m}{n}\frac{d\mu(\mathcal{S})}{n}\|y\|_2^2$$

with probability at least $1 - \delta$.

Lemma 3.4. *With the same notations as Theorem 3.1,*

$$\| \left(U_\Omega^T U_\Omega\right)^{-1} \|_2 \leq \frac{n}{(1-\gamma)m}$$

with probability at least $1 - \delta$, provided that $\gamma < 1$.

Discussion of Theorem 3.1

In this section we wish to give some intuition for the lower bound in Theorem 3.1. If the parameters α, β, γ are very near 0 (e.g., as $n \to \infty$ and m is a constant fraction of n), our lower bound is

approximately equal to

$$\frac{m - d\mu(\mathcal{S})}{n}\|v - P_{\mathcal{S}}v\|_2^2$$

For an incoherent subspace, the parameter $\mu(\mathcal{S}) = 1$. In this case, for $m \leq d$ the bound is ≤ 0, which is consistent with the fact that if $|\Omega| =: m < d$, it is very possible to fit the data such that $\|v_\Omega - P_{\mathcal{S}_\Omega} v_\Omega\|_2 = 0$. For example consider the case where our subspace has dimension $d = 1$ and we have only one measurement from our vector; unless that subspace coordinate corresponding to the measurement location is 0, it is always possible to fit our vector exactly in the subspace on that one coordinate.

Once $m \geq d+1$, linear algebraic reasoning tells us that $\|v_\Omega - P_{\mathcal{S}_\Omega} v_\Omega\|_2$ will be strictly positive with positive probability; Theorem 3.1 goes further to say the norm is strictly positive with high probability once $m \sim \mathcal{O}(d \log d)$.

The parameters α, β, γ all depend on $\sqrt{\log\left(\frac{1}{\delta}\right)}$; these parameters grow as δ gets very small. Increasing the number of observations m will counteract this behavior for α and γ, but this does not hold for β. In fact, even if the vector y is incoherent and $\mu(y) = 1$, its minimum value, then $\beta = 2$ for $\delta \approx .135$. To get β very near zero, δ must be *very* near one, but this is not a useful regime.

We can see, however, that in simulations these large constants are somewhat irrelevant; The large deviations analysis needed for the proof is overly conservative in most cases. We investigate this in more detail in Section 3.5.

3.2.2 Orthogonality Principle and Magnitude of the Projection

As a direct consequence of Lemma 3.4, invertibility of $U_\Omega^T U_\Omega$, we have that the incomplete data projection residual is orthogonal to the projection $P_{\mathcal{S}_\Omega} v_\Omega$ itself.

Corollary.

$$\langle P_{\mathcal{S}_\Omega} v_\Omega, (I - P_{\mathcal{S}_\Omega}) v_\Omega \rangle = 0 \ .$$

Proof.

$$\begin{aligned}\langle P_{\mathcal{S}_\Omega} v_\Omega, (I - P_{\mathcal{S}_\Omega}) v_\Omega \rangle &= v_\Omega^T U_\Omega \left(U_\Omega^T U_\Omega\right)^{-1} \left[I - \left(U_\Omega^T U_\Omega\right)\left(U_\Omega^T U_\Omega\right)^{-1}\right] U_\Omega^T v_\Omega \\ &= 0\end{aligned}$$

since $\left(U_\Omega^T U_\Omega\right)^{-1}$ exists. □

The next consequence is therefore that v_Ω can be written as the direct sum of $P_{\mathcal{S}_\Omega} v_\Omega$ and $(I - P_{\mathcal{S}_\Omega}) v_\Omega$. Thus $\|v_\Omega\|_2^2 = \|v_\Omega - P_{\mathcal{S}_\Omega} v_\Omega\|_2^2 + \|P_{\mathcal{S}_\Omega} v_\Omega\|_2^2$ and we have the following corollary.

Corollary. *Let $\delta > 0$ and $m \geq \frac{8}{3} d\mu(\mathcal{S}) \log\left(\frac{2d}{\delta}\right)$. Then with probability at least $1 - 6\delta$,*

$$(1 - \alpha_v)\frac{m}{n}\|v\|_2^2 - (1 + \alpha)\frac{m}{n}\|v - P_{\mathcal{S}} v\|_2^2 \leq \|P_{\mathcal{S}_\Omega} v_\Omega\|_2^2 \tag{3.3}$$

and

$$\|P_{\mathcal{S}_\Omega} v_\Omega\|_2^2 \leq (1 + \alpha_v)\frac{m}{n}\|v\|_2^2 - \frac{m(1-\alpha) - d\mu(\mathcal{S})\frac{(1+\beta)^2}{(1-\gamma)}}{n}\|v - P_{\mathcal{S}} v\|_2^2$$

where $\alpha_v = \sqrt{\frac{2\mu(v)^2}{m}\log\left(\frac{1}{\delta}\right)}$, and otherwise we have the same notations as Theorem 3.1.

For intuition here note that as $n \to \infty$ and $m \to n$, the parameters $\alpha_v, \alpha \to 0$ and the left hand side of Equation 3.3 approaches $\|v\|_2^2 - \|v - P_{\mathcal{S}} v\|_2^2 = \|P_{\mathcal{S}} v\|_2^2$. An open question is to develop a result which reduces to this case when $m = n$, no matter how large; of course presently we are basing our results on sampling with replacement, which precludes such a result. Also here we wish to note that all the results of this section are relative bounds: if $\|P_{\mathcal{S}} v\|_2^2$ is itself negligible, the result is accordingly less powerful.

Proof. By Lemma 3.2, we have with probability at least $1 - 2\delta$ that

$$(1 - \alpha_v)\frac{m}{n}\|v\|_2^2 \leq \|v_\Omega\|_2^2 \leq (1 + \alpha_v)\frac{m}{n}\|v\|_2^2 \,. \tag{3.4}$$

Combine this with Theorem 3.1 and we have the result. □

3.2.3 Angle Estimation from Incomplete Data

Since the residual norm is preserved, then we are also able to estimate the angle between v and its projection onto $\mathcal{S}$. Define the angle between the vector v and its projection into the subspaces $\mathcal{S}$ as θ:

$$\theta = \sin^{-1}\left(\frac{\|v - P_{\mathcal{S}}v\|_2}{\|v\|_2}\right) \tag{3.5}$$

and similarly define an estimate of θ given only the observed entries v_Ω:

$$\theta_\Omega = \sin^{-1}\left(\frac{\|v_\Omega - P_{\mathcal{S}_\Omega}v_\Omega\|_2}{\|v_\Omega\|_2}\right) \tag{3.6}$$

As before, let $v = x + y$, where $x \in \mathcal{S}$, $y \perp S$.

Corollary. *Let $\delta > 0$ and $m \geq \frac{8}{3}d\mu(\mathcal{S})\log\left(\frac{2d}{\delta}\right)$. Then with probability at least $1 - 6\delta$,*

$$C(m)\sin^2(\theta) \leq \sin^2(\theta_\Omega) \leq \frac{1+\alpha}{1-\alpha_v}\sin^2(\theta) \quad ,$$

where

$$\alpha_v = \sqrt{\frac{2\mu(v)^2}{m}\log\left(\frac{1}{\delta}\right)} \, ,$$

and

$$C(m) = \frac{m(1-\alpha) - d\mu(\mathcal{S})\frac{(1+\beta)^2}{(1-\gamma)}}{m(1+\alpha_v)} \, , \tag{3.7}$$

where α, β, and γ are defined as in Theorem 3.1 and α_v as in Corollary 3.2.2.

Before the proof we point out that $C(m) \nearrow 1$ as $m \to \infty$.

Proof. Take together Theorem 3.1 and the statement in Equation 3.4 which holds with probability at least $1 - 2\delta$. Using the union bound we therefore have that with probability at least $1 - 6\delta$,

$$C(m)\frac{\|v - P_{\mathcal{S}}v\|_2^2}{\|v\|_2^2} \leq \frac{\|v_\Omega - P_{\mathcal{S}_\Omega}v_\Omega\|_2^2}{\|v_\Omega\|_2^2} \leq \frac{1+\alpha}{1-\alpha_v}\frac{\|v - P_{\mathcal{S}}v\|_2^2}{\|v\|_2^2}$$

Upon substitution of 3.5 and 3.6, the proof is complete. □

3.3 Subspace Assignment

Subspace assignment is the problem of assigning a data vector to one of a collection of subspaces, i.e. a multiple hypothesis testing problem. This problem arises in any context where data are modeled not as a single linear subspace but as a union of subspaces, or in other words, as a collection of low-dimensional linear embeddings. The results in the previous section can be immediately extended to the problem of subspace assignment with missing data.

Modeling high-dimensional data with a union of subspaces is a useful generalization of subspace models [67], and has applications in machine learning, imaging, computer vision [32], and system identification [103]. We discuss estimation of a union of subspaces model with missing data in Chapter 5. An important problem which arises when modeling data with a union of subspaces is subspace clustering, or clustering vectors into groups that lie in or near the same subspace. The results here will therefore be useful in Chapter 5 when vectors must be assigned to one of a collection of subspaces.

Many of the subspace clustering algorithms have a subroutine which assigns data vectors to subspaces based on projection residuals [101]. For example, the k-planes clustering algorithm[1] [18, 1] is an alternating minimization algorithm which alternates between estimating subspace parameters given a clustering of the data and clustering the data using projections to assign the data points to fixed subspaces.

[1] This is also called k-subspaces in the literature, and we use the terminology interchangeably.

3.3.1 Problem Formulation

The subspace assignment problem can be defined as follows. Given a vector $v \in \mathbb{R}^n$ and subspaces $\mathcal{S}^0, \ldots, \mathcal{S}^k$, we wish to determine the subspace closest to v. With full data, we project v onto each subspace and choose the subspace with the smallest residual:

$$\text{argmin}_i \|v - P_{\mathcal{S}^i} v\|_2^2 \ , \ \ i = 1, \ldots, k \ ,$$

where $P_{\mathcal{S}^i}$ is the projection operator onto subspace $\mathcal{S}^i$.

The natural question for this thesis is, given an *incomplete* data vector, how do we determine to which of k subspaces the vector is closest? We begin by examining the problem of binary subspace assignment, i.e. $k = 2$. Let $v \in \mathbb{R}^n$ and let $\mathcal{S}^0 \subset \mathbb{R}^n$ and $\mathcal{S}^1 \subset \mathbb{R}^n$ be subspaces of dimension d_0 and d_1 respectively. Is v closer to $\mathcal{S}^0$ or $\mathcal{S}^1$? If we had complete data, we would compare the norm of the projection residual of v onto both $\mathcal{S}^0$ and $\mathcal{S}^1$:

$$\|v - P_{\mathcal{S}^0} v\|_2^2 \underset{\mathcal{S}^0}{\overset{\mathcal{S}^1}{\gtrless}} \|v - P_{\mathcal{S}^1} v\|_2^2 \ . \tag{3.8}$$

Now consider the situation when we only observe a set or multiset $\Omega \subset \{1, \ldots, n\}$ of indices of v. Denote the observed vector as v_Ω. Let the columns of an orthonormal matrix U span the d-dimensional subspace $\mathcal{S} \in \mathbb{R}^n$. Then we define the projection operator restricted to Ω as

$$P_{\mathcal{S}_\Omega} = U_\Omega \left(U_\Omega^T U_\Omega \right)^{-1} U_\Omega^T \ . \tag{3.9}$$

where the notation U_Ω denotes a restriction to the rows of U indicated by the multiset Ω. We base our subspace assignment on this projection residual:

$$\|v_\Omega - P_{\mathcal{S}_\Omega^0} v_\Omega\|_2^2 \overset{?}{<} \|v_\Omega - P_{\mathcal{S}_\Omega^1} v_\Omega\|_2^2 \ . \tag{3.10}$$

In what follows we show that with enough observations, the subspace assignment based on (3.10) will be the same as that for (3.8) with high probability.

3.3.2 Theoretical Analysis

Define the angle between the vector v and its projection into the two subspaces $\mathcal{S}^0, \mathcal{S}^1$ as θ_0 and θ_1:

$$\theta_0 = \sin^{-1}\left(\frac{\|v - P_{\mathcal{S}^0} v\|_2}{\|v\|_2}\right) \tag{3.11}$$

and θ_1 is defined similarly. Following the notation of [8], let $v = x_0 + y_0 = x_1 + y_1$, where $x_0 \in \mathcal{S}^0$, $y_0 \perp S^0$, $x_1 \in \mathcal{S}^1$, and $y_1 \perp S^1$. Let $\mu(\mathcal{S}) = \frac{n}{r} max_j \|P_{\mathcal{S}} e_j\|_2^2$, where e_j is the j^{th} canonical basis vector. Let $m := |\Omega|$ and choose a $\delta > 0$ as a confidence parameter. For notational simplicity and without loss of generality we focus on the situation when $\theta_0 < \theta_1$ and define

$$C(m) = \frac{m(1-\alpha_1) - d_1 \mu(\mathcal{S}^1) \frac{(1+\beta_1)^2}{(1-\gamma_1)}}{m(1+\alpha_0)} , \tag{3.12}$$

where $\alpha_1 = \sqrt{\frac{2\mu(y_1)^2}{m} \log\left(\frac{1}{\delta}\right)}$, $\beta_1 = \sqrt{2\mu(y_1) \log\left(\frac{1}{\delta}\right)}$, $\gamma_1 = \sqrt{\frac{8 d_1 \mu(\mathcal{S}^1)}{3m} \log\left(\frac{2d_1}{\delta}\right)}$, and $\alpha_0 = \sqrt{\frac{2\mu(y_0)^2}{m} \log\left(\frac{1}{\delta}\right)}$.

Notice that $C(m) \nearrow 1$ as $m \to \infty$.

Theorem 3.2. *Let $\delta > 0$ and $m \geq \frac{8}{3} d_1 \mu(\mathcal{S}^1) \log\left(\frac{2d_1}{\delta}\right)$. Assume that*

$$\sin^2(\theta_0) < C(m) \sin^2(\theta_1) . \tag{3.13}$$

Then with probability at least $1 - 4\delta$,

$$\|v_\Omega - P_{\mathcal{S}^0_\Omega} v_\Omega\|_2^2 < \|v_\Omega - P_{\mathcal{S}^1_\Omega} v_\Omega\|_2^2 .$$

Before the proof we consider consequences of the theorem. First we consider the situation where $\theta_0 = 0$, i.e., the vector v is *in* the hypothesized subspace $\mathcal{S}^0$. This particular case was proved in [3]. As long as $\theta_1 \neq 0$, the ratio $\sin^2(\theta_0)/\sin^2(\theta_1) = 0$. This in turn implies that the number of observations

required does not depend on θ_1 nor on the relationship of θ_0 to θ_1, and the condition (3.13) is simply that $C(m) > 0$. To guarantee $\theta_1 \neq 0$ for arbitrary $v \in \mathcal{S}^0$, we must have that $\mathcal{S}^0$ and $\mathcal{S}^1$ are linearly independent[2]. In other words, if $\mathcal{S}^0$ and $\mathcal{S}^1$ are linearly independent, and the vector is in either $\mathcal{S}^0$ or $\mathcal{S}^1$, the number of observations to guarantee the test works does not depend on the angle of v to the other subspace. If, on the other hand, $\mathcal{S}^0$ and $\mathcal{S}^1$ are not linearly independent, there are vectors in the two subspaces which are arbitrarily close to one another; for any fixed m there exists a vector in $\mathcal{S}^0$ for which the incomplete data projection residual would not be valid.

Now we consider the situation where v is not *in* the subspace, but is simply *closer*: $0 < \theta_0 < \theta_1$. Thus $\sin^2(\theta_0)/\sin^2(\theta_1) > 0$. As the gap $\theta_1 - \theta_0$ decreases, $\sin^2(\theta_0)/\sin^2(\theta_1) \nearrow 1$. Consequently, as this gap narrows, we must increase m to guarantee that the subspace assignment based on (3.10) gives the same result as that of (3.8).

Proof. From Theorem 1 of [8] and the union bound, the following two statements hold simultaneously with probability at least $1 - 4\delta$:

$$\|v_\Omega - P_{\mathcal{S}^0_\Omega} v_\Omega\|_2^2 \leq (1 + \alpha_0)\frac{m}{n}\|v - P_{\mathcal{S}^0} v\|_2^2$$

and

$$\frac{m(1-\alpha_1) - d_1\mu(\mathcal{S}^1)\frac{(1+\beta_1)^2}{(1-\gamma_1)}}{n}\|v - P_{\mathcal{S}^1} v\|_2^2 \leq \|v_\Omega - P_{\mathcal{S}^1_\Omega} v_\Omega\|_2^2 \,.$$

Thus if

$$\|v - P_{\mathcal{S}^0} v\|_2^2 < C(m)\|v - P_{\mathcal{S}^1} v\|_2^2 \,, \tag{3.14}$$

we have the conclusion of the theorem. But using (3.11), this statement is equivalent to our requirement that $\sin^2(\theta_0) < C(m)\sin^2(\theta_1)$, completing the proof. □

This result can be directly extended to the situation where there are multiple subspaces. Again

[2]Two subspaces are linearly independent if the dimension of their union is equal to the sum of their dimensions.

without loss of generality we focus on the situation where $\theta_0 < \theta_i, \forall i$, and define

$$C_i(m) = \frac{m(1-\alpha_i) - d_i\mu(\mathcal{S}^i)\frac{(1+\beta_i)^2}{(1-\gamma_i)}}{m(1+\alpha_0)} ,$$

where α_i, β_i, and γ_i are defined as in Equation 3.12 using d_i, $\mu(y_i)$ and $\mu(\mathcal{S}^i)$.

Corollary. *Let* $m \geq \frac{8}{3} max_{i\neq 0}\left(d_i\mu(\mathcal{S}^i)\log\left(\frac{2d_i}{\delta}\right)\right)$ *for fixed* $\delta > 0$*. Assume that*

$$\sin^2(\theta_0) < C_i(m)\sin^2(\theta_i) , \quad \forall i \neq 0 .$$

Then with probability at least $1 - 4(k-1)\delta$*,*

$$\|v_\Omega - P_{\mathcal{S}^0_\Omega} v_\Omega\|_2^2 < \|v_\Omega - P_{\mathcal{S}^i_\Omega} v_\Omega\|_2^2 , \quad \forall i \neq 0 .$$

3.4 Matched Subspace Detection

Now we consider a variation on the classical problem of Subspace Detection. Subspace Detection with missing data is similar to subspace assignment, but in this case we have one subspace $\mathcal{S}$ and the null hypothesis is the orthogonal complement of $\mathcal{S}$ (i.e. $\mathcal{S}^\perp$). Again we assume that only a small subset or multiset $\Omega \subset \{1, \ldots, n\}$ of the elements of v are observed (with or without noise), and based on these observations we want to test whether $v \in \mathcal{S}$.

For example, consider monitoring a large networked system such as a portion of the Internet. Measurement nodes in the network may have software that collects measurements such as upload and download rate, number of packets, or type of traffic given by the packet headers. In order to monitor the network, these measurements will be collected in a central place for compilation, modeling and analysis. The effective dimension of the state of such systems is often much lower than the extrinsic dimension of the network itself. Subspace detection, therefore, can be a useful tool for detecting changes

or anomalies [61]. The challenge is that it may be impossible to obtain every measurement from every point in the network due to resource constraints, node outages, etc.

Given a subspace $\mathcal{S}$ of dimension $d \ll n$, how many elements of v must be observed so that we can reliably decide if it belongs to $\mathcal{S}$? From Theorem 3.1 we have that the number of required measurements is $O(d \log d)$. This means that reliable matched subspace detectors can be constructed from very few measurements, making them scalable and applicable to large-scale testing problems.

We have the following detection set up. Our hypotheses are $\mathcal{H}_0 : v \in \mathcal{S}$ and $\mathcal{H}_1 : v \notin S$ and the test statistic we will use is

$$t(v_\Omega) = \|v_\Omega - P_{\mathcal{S}_\Omega} v_\Omega\|_2^2 \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} \eta$$

In the noiseless case, we can let $\eta = 0$; our result in Theorem 3.1 shows for $\delta > 0$, the probability of detection is $P_D = \mathbb{P}\left[t(v_\Omega) > 0 | \mathcal{H}_1\right] \geq 1 - 4\delta$ as long as m is large enough, and we also have that the probability of false alarm is zero, $P_{FA} = \mathbb{P}\left[t(v_\Omega) > 0 | \mathcal{H}_0\right] = 0$ since the projection error will be zero when $v \in \mathcal{S}$.

When we introduce noise we have the same hypotheses, but we compute the statistic on $\widetilde{v}_\Omega = v_\Omega + w$ where $w \sim \mathcal{N}(0, 1)$ is Gaussian white noise:

$$t(\widetilde{v}_\Omega) = \|\widetilde{v}_\Omega - P_{\mathcal{S}_\Omega} \widetilde{v}_\Omega\|_2^2 \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} \eta_\lambda$$

We choose η_λ to fix the probability of false alarm:

$$\mathbb{P}\left[t(\widetilde{v}_\Omega) > \eta_\lambda | \mathcal{H}_0\right] \leq \lambda = P_{FA}$$

Then we have from [92] that $t(\tilde{v}_\Omega)$ is distributed as a non-central χ^2 with d degrees of freedom and non-centrality parameter $\|v_\Omega - P_{\mathcal{S}_\Omega} v_\Omega\|_2^2$, and that P_D is monotonically increasing with the non-centrality parameter. Putting this together with Theorem 3.1 we see that as m grows, $\|v_\Omega - P_{\mathcal{S}_\Omega} v_\Omega\|_2^2$ grows and thus the probability of detection grows.

3.4.1 Comparison to a Zero-filling Estimator

Another heuristic approach that is often used in practice is to fill the vector v with zeros and then project onto the full subspace $\mathcal{S}$. We will denote the zero-filled vector as v_0, i.e. we have an $n \times 1$ vector v_0 with elements v_i if $i \in \Omega$ and zero if $i \notin \Omega$, for $i = 1, \ldots, n$. We then calculate the projection energy only on the observed entries:

$$\| v_\Omega - (P_\mathcal{S} v_0)_\Omega \|_2^2$$

We now show why the heuristic approach of zero-filling the incomplete vector v_Ω does not work. Even if $v \in \mathcal{S}$, the zero-filled vector v_0 does not necessarily lie in $\mathcal{S}$.

We calculate the projection energy only on the observed entries:

$$t_0(v_\Omega) = \| v_\Omega - (P_\mathcal{S} v_0)_\Omega \|_2^2 \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} \eta$$

Simple algebraic consideration reveals that $t_0(v_\Omega)|\mathcal{H}_0$ is positive. In fact, even in the absence of noise, the probability of false alarm can be arbitrarily large as $\|v\|_2^2$ increases. The value of $t_0(v_\Omega)|\mathcal{H}_0$, based on noiseless observations, is plotted as a function of the number of measurements in Figure 4 in Section 3.5.

We note that for unknown noise power or structured interference, these results can be extended using the GLRT [93].

3.5 Empirical Analysis

Detection First we examine the theory in Section 3.2.1 in simulation. As we said in Section 3.2.1, in simulations the large constants of the theorem are somewhat irrelevant. This plays out in the simulations shown in Figure 3, where we see that for very incoherent subspaces, $\|v_\Omega - P_{\mathcal{S}_\Omega} v_\Omega\|_2$ is always positive for $m > d\mu(\mathcal{S}) \log d$. The plots show the minimum, maximum and mean value of $\|v_\Omega - P_{\mathcal{S}_\Omega} v_\Omega\|_2$ over 100 simulations, for fixed $\mathcal{S}$ and fixed v such that $\|v\|_2^2 = 1$ and $v \in \mathcal{S}^\perp$. For each value of the sample size m, we sampled 100 different instances of Ω *without* replacement, giving us a realistic

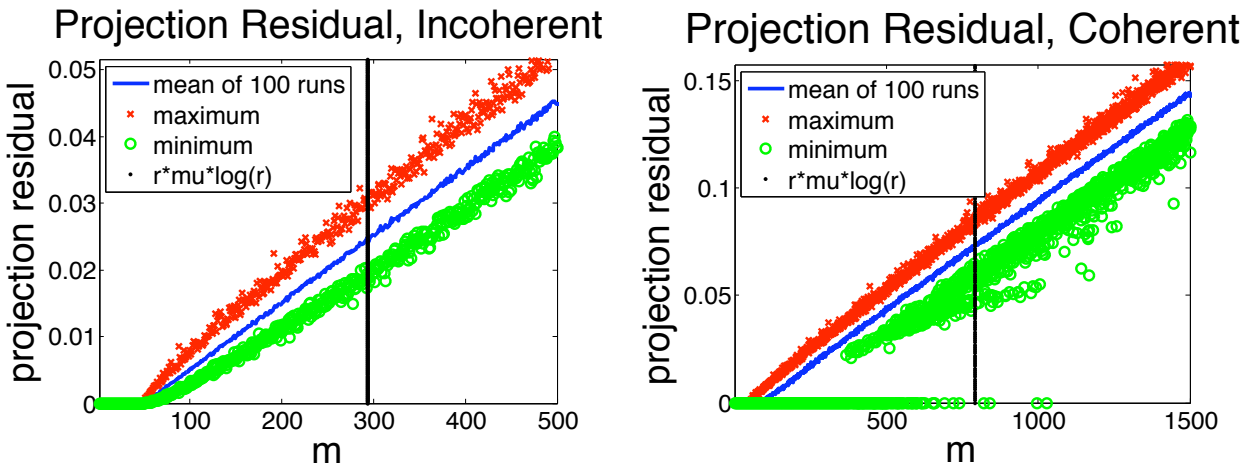


Figure 3: These plots show the projection residual $\|v_\Omega - P_{\mathcal{S}_\Omega} v_\Omega\|_2^2$ over 100 simulations. Each of the simulations has a fixed subspace $\mathcal{S}$, vector $v \in \mathcal{S}^\perp$ and sample size m, but different sample set Ω drawn *without* replacement. The problem size is $n = 10000$, $d = 50$. (a) Incoherent subspace (random Gaussian basis). $\mu(\mathcal{S}) \approx 1.5$, $\mu(y) \approx 13.6$. (b) Coherent subspace. $\mu(\mathcal{S}) \approx 4.1$, $\mu(y) \approx 47.0$.

idea of how much energy of v is captured by m samples. Our simulations for the Fourier basis and a basis made of orthogonalized Gaussian random vectors always showed the estimate to be positive for $m > d\mu(\mathcal{S}) \log d$, even for the worst-case simulation run. For more coherent subspaces, we often (but not always) see that the norm is positive as long as $m > d\mu(\mathcal{S}) \log d$.

Assignment Next we illustrate the output of the test given by (3.10) by showing its behavior in simulation. First we consider the most basic scenario where both subspaces are of the same dimension. Figure 5 shows the behavior of the comparison of projection residuals depending on the angle difference, $\theta_0 - \theta_1$. As the angle difference nears zero, even nearly complete vectors do not always result in the correct subspace being chosen.

Next we consider a scenario where the subspaces are of different dimension. Figure 6 shows the same plot of means as Figure 5, but the asymmetry is due to the imbalance of the subspace dimensions. In Figures 7 and 8 we examine the difference of the projection residual as m increases, for unbalanced subspace dimensions of $d_0 = 5$ and $d_1 = 20$. Both plots have a fixed vector v at a fixed angle to

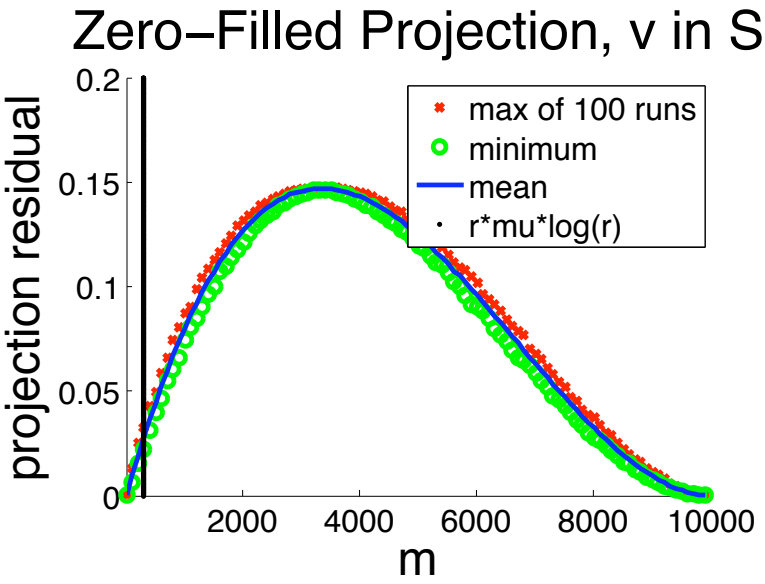


Figure 4: Simulation results for the zero-filling approach, $v \in \mathcal{S}$, $\|v\|_2^2 = 1$. The basis used is a random Gaussian basis, $d = 50$, $n = 10000$, $\mu(\mathcal{S}) \approx 1.5$, $\mu(y) \approx 17.9$. Note that the zero-filled residuals can be made arbitrarily large by increasing $\|v\|_2^2$.

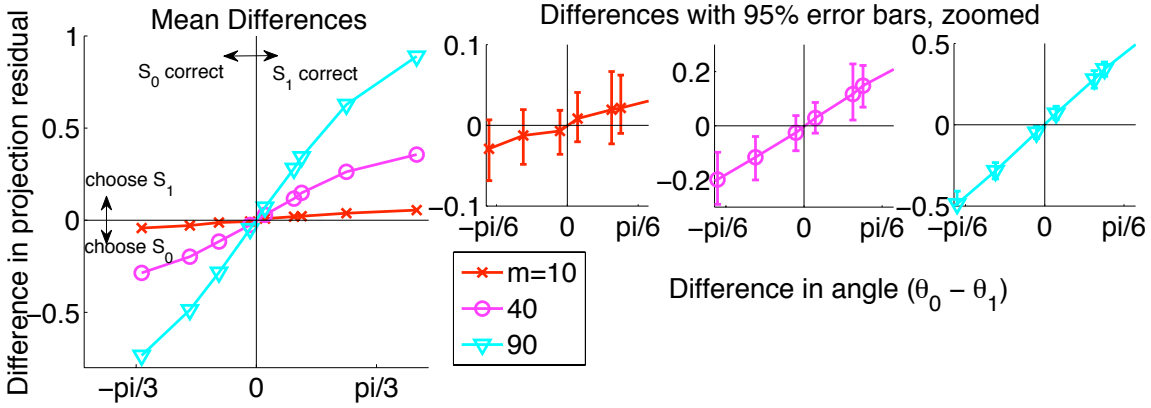


Figure 5: Simulation results for a binary subspace assignment with $n = 100$, $d_0 = 5$, $d_1 = 5$. Projection Residual Difference is defined as $\|v_\Omega - P_{\mathcal{S}^0_\Omega} v_\Omega\|_2^2 - \|v_\Omega - P_{\mathcal{S}^1_\Omega} v_\Omega\|_2^2$; thus, $\mathcal{S}^0$ is the chosen subspace when the residual is negative. The curves shown are averaged over 100 random sample sets of various sizes as denoted by the legend. The right three plots are zoomed around $\theta_0 - \theta_1 = 0$; 95% confidence intervals are shown.

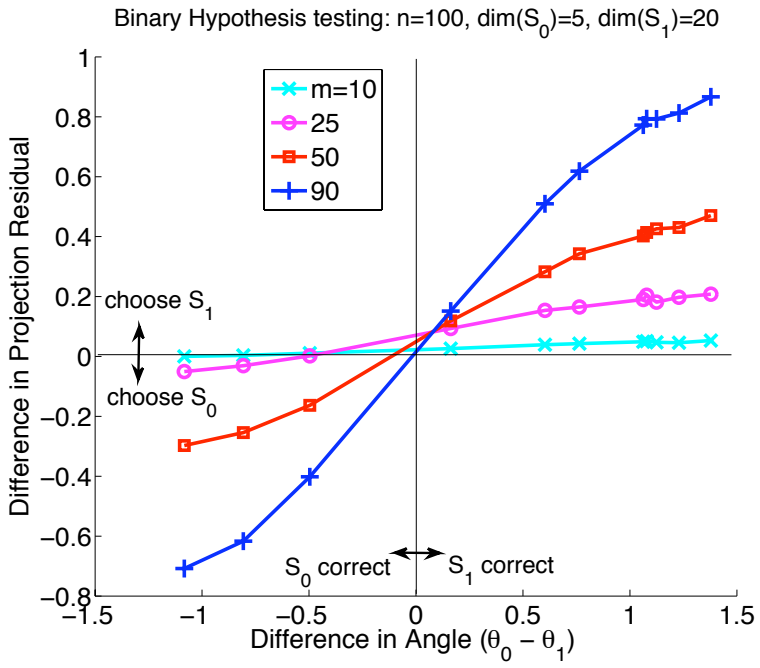


Figure 6: Simulation results for a binary subspace assignment with $n = 100$, $d_0 = 5$, $d_1 = 20$. Projection Residual Difference is defined as $\|v_\Omega - P_{\mathcal{S}^0_\Omega} v_\Omega\|_2^2 - \|v_\Omega - P_{\mathcal{S}^1_\Omega} v_\Omega\|_2^2$. The curves shown are averaged over 100 random sample sets of various sizes as denoted by the legend.

both subspaces, but the number of observations m is varied. Figure 7 shows the projection residual difference for two vectors v; one which is closer to $\mathcal{S}^0$ and one which is closer to $\mathcal{S}^1$. As is evident, the vector nearer $\mathcal{S}_1$ can be distinguished with a small number of observations. However, the vector closer to $\mathcal{S}^0$ looks as though it may be near either subspace until we have more than 20 observations.

Finally, Figure 8 shows a similar scenario but with a smaller difference in angle and a larger ambient dimension. Here were highlight the behavior of the residual near the measurement cutoff $d_1\mu_1 log(d_1)$. Though the mean residual difference, averaged over many possible sample sets, is almost always below zero, the worst-case difference $\|v_\Omega - P_{\mathcal{S}^0_\Omega} v_\Omega\|_2^2 - \|v_\Omega - P_{\mathcal{S}^1_\Omega} v_\Omega\|_2^2$ is often positive, in which case we would incorrectly select $\mathcal{S}^1$.

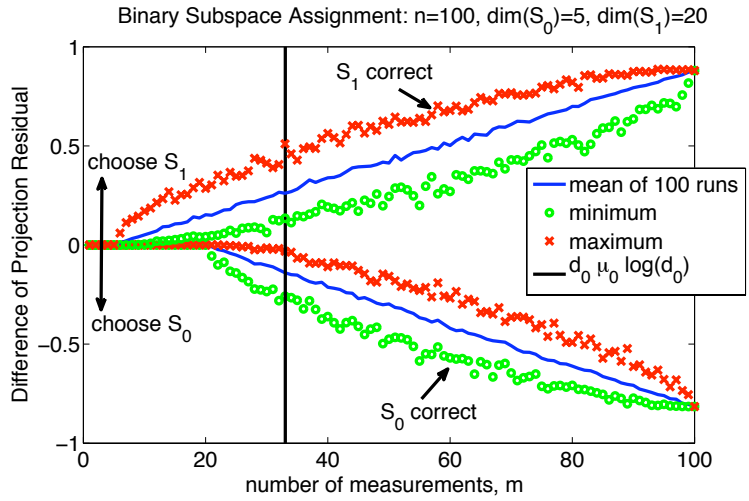


Figure 7: Simulation results for a binary subspace assignment with $n = 100$, $d_0 = 5$, $d_1 = 20$. The upper curves represent a case where the angle difference $\theta_0 - \theta_1 = 1.078$ degrees, thus resulting in $\mathcal{S}_1$ being the correct choice. The lower curves represent a case where the angle difference $\theta_1 - \theta_0 = 1.083$ degrees, thus resulting in $\mathcal{S}^0$ being the correct choice.

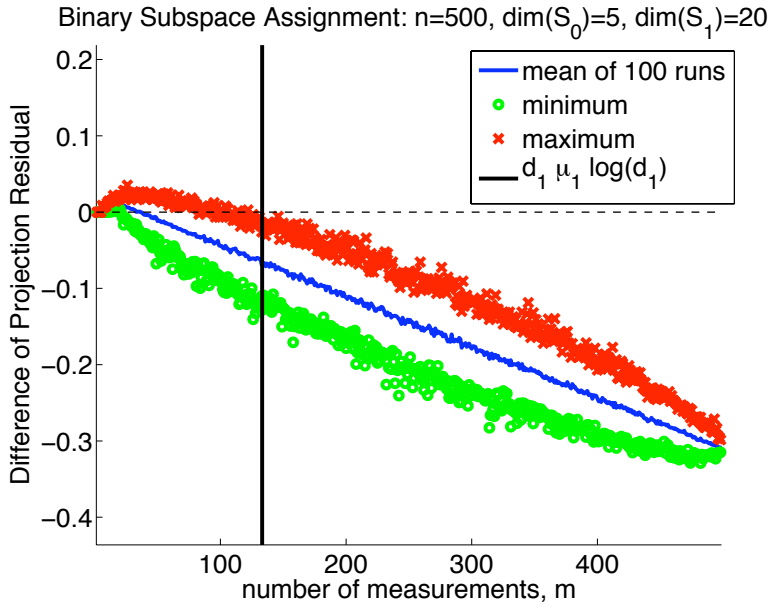


Figure 8: Simulation results for a binary subspace assignment with $n = 500$, $d_0 = 5$, $d_1 = 20$. The angle difference $\theta_1 - \theta_0 = 0.3156$ degrees, thus $\mathcal{S}^0$ is the correct choice. The approximate measurement requirement $d_1 \mu_1 \log(d_1)$ is the black line shown; beyond it the worst-case residual difference is in favor of $\mathcal{S}^0$.

3.6 Proofs

Proof. In order to prove Theorem 3.1, we split the quantity of interest into three terms and bound each with high probability. Consider $\|v_\Omega - P_{\mathcal{S}_\Omega} v_\Omega\|_2^2 = \|y_\Omega - P_{\mathcal{S}_\Omega} y_\Omega\|_2^2$ (Note this equality holds by Lemma 3.1). Let the d columns of U be an orthonormal basis for the subspace $\mathcal{S}$. We want to show that

$$\|y_\Omega - P_{\mathcal{S}_\Omega} y_\Omega\|_2^2 = \|y_\Omega\|_2^2 - y_\Omega^T U_\Omega \left(U_\Omega^T U_\Omega\right)^{-1} U_\Omega^T y_\Omega \tag{3.15}$$

is near $\frac{m}{n}\|y\|_2^2$ with high probability.

We first apply the three Lemmas of Section 3.2.1 to prove Theorem 3.1, and then we prove the lemmas in Section 3.6.2. Write the second term of Equation (3.15) as

$$y_\Omega^T U_\Omega \left(U_\Omega^T U_\Omega\right)^{-1} U_\Omega^T y_\Omega = \|W_\Omega U_\Omega^T y_\Omega\|_2^2$$

where $W_\Omega^T W_\Omega = \left(U_\Omega^T U_\Omega\right)^{-1}$. By Lemma 3.4, $U_\Omega^T U_\Omega$ is invertible under the assumptions of our theorem. Since it is additionally symmetric and positive semidefinite, its inverse is as well, and both have real eigenvalues. Therefore, the eigenvalues of W_Ω are the square roots of the eigenvalues of $W_\Omega^T W_\Omega$, and in particular the spectral norms are related by $\|W_\Omega^T W_\Omega\|_2 = \|W_\Omega\|_2^2$. Hence W_Ω has spectral norm bounded by the square root of the inverse of the smallest eigenvalue of $U_\Omega^T U_\Omega$. That is, we have

$$\begin{aligned}
\|W_\Omega U_\Omega^T y_\Omega\|_2^2 &\leq \|W_\Omega\|_2^2 \|U_\Omega^T y_\Omega\|_2^2 \\
&= \|W_\Omega^T W_\Omega\|_2 \|U_\Omega^T y_\Omega\|_2^2 \\
&= \| \left(U_\Omega^T U_\Omega\right)^{-1} \|_2 \|U_\Omega^T y_\Omega\|_2^2 .
\end{aligned}$$

$\| \left(U_\Omega^T U_\Omega\right)^{-1} \|_2$ is bounded by Lemma 3.4 and $\|U_\Omega^T y_\Omega\|_2$ is bounded by Lemma 3.3. Putting these two bounds together with the bounds in Lemma 3.2 and using the union bound, we have that with

probability at least $1 - 4\delta$

$$(1+\alpha)^2 \frac{m}{n}\|y\|_2^2 \geq \|y_\Omega\|_2^2 - \| \left(U_\Omega^T U_\Omega\right)^{-1} \|_2 \|U_\Omega^T y_\Omega\|_2^2$$
$$\geq (1-\alpha)^2 \frac{m}{n}\|y\|_2^2 - \frac{(\beta+1)^2 d\mu(\mathcal{S})}{(1-\gamma)n}\|y\|_2^2$$

giving us our bound. □

3.6.1 Useful Inequalities

We will need the following two large deviation bounds in the proofs of our Lemmas below.

Theorem 3.3 (McDiarmid's Inequality [74])**.** *Let $X_1, \ldots, X_n$ be independent random variables, and assume f is a function for which there exist t_i, $i = 1, \ldots, n$ satisfying*

$$\sup_{x_1,\ldots,x_n,\hat{x_i}} |f(x_1, \ldots, x_n) - f(x_1, \ldots, \hat{x_i}, \ldots, x_n)| \leq t_i$$

where $\hat{x_i}$ indicates replacing the sample value x_i with any other of its possible values. Call $f(X_1, \ldots, X_n) := Y$. Then for any $\epsilon > 0$,

$$\mathbb{P}\left[Y \geq \mathbb{E}\left[Y\right] + \epsilon\right] \leq \exp\left(\frac{-2\epsilon^2}{\sum_{i=1}^n t_i^2}\right) \tag{3.16}$$

$$\mathbb{P}\left[Y \leq \mathbb{E}\left[Y\right] - \epsilon\right] \leq \exp\left(\frac{-2\epsilon^2}{\sum_{i=1}^n t_i^2}\right) \tag{3.17}$$

Theorem 3.4 (Noncommutative Bernstein Inequality [46, 85])**.** *Let $X_1, \ldots, X_m$ be independent zero-mean square $d \times d$ random matrices. Suppose $\rho_k^2 = max\{\|\mathbb{E}[X_k X_k^T]\|_2, \|\mathbb{E}[X_k^T X_k]\|_2\}$ and $\|X_k\|_2 \leq M$ almost surely for all k. Then for any $\tau > 0$,*

$$\mathbb{P}\left[\left\|\sum_{k=1}^m X_k\right\|_2 > \tau\right] \leq 2r \exp\left(\frac{-\tau^2/2}{\sum_{k=1}^m \rho_k^2 + M\tau/3}\right)$$

3.6.2 Supporting Lemmas for Theorem 3.1 and Proofs

We now proceed with the proof of Lemmas 3.2, 3.3, and 3.4 which support the proof of Theorem 3.1 in Section 3.2.1. Our analysis is informed by that in [46, 85].

Proof of Lemma 3.2. To prove this we use McDiarmid's inequality from Theorem 3.3 for the function $f(X_1, \ldots, X_m) = \sum_{i=1}^{m} X_i$. The resulting inequality is more commonly referred to as Hoeffding's inequality.

We begin with the first inequality. Set $X_i = y^2_{\Omega(i)}$. We seek a good value for t_i. Since $y^2_{\Omega(i)} \leq \|y\|^2_\infty$ for all i, we have

$$\left| \sum_{i=1}^{m} X_i - \sum_{i \neq k} X_i - \hat{X}_k \right| \quad = \quad \left| X_k - \hat{X}_k \right| \leq 2\|y\|^2_\infty$$

We calculate $\mathbb{E}\left[\sum_{i=1}^{m} X_i\right]$ as follows. Define $\mathbb{I}_{\{\}}$ to be the indicator function, and assume that the samples are taken uniformly with replacement.

$$\begin{aligned} \mathbb{E}\left[\sum_{i=1}^{m} X_i\right] &= \mathbb{E}\left[\sum_{i=1}^{m} y^2_{\Omega(i)}\right] \\ &= \sum_{i=1}^{m} \mathbb{E}\left[\sum_{j=1}^{n} y_j^2 \mathbb{I}_{\{\Omega(i)=j\}}\right] = \frac{m}{n}\|y\|_2^2 \end{aligned}$$

Plugging into Equation (3.17), the left hand side is

$$\mathbb{P}\left[\sum_{i=1}^{m} X_i \leq \mathbb{E}\left[\sum_{i=1}^{m} X_i\right] - \epsilon\right] = \mathbb{P}\left[\sum_{i=1}^{m} X_i \leq \frac{m}{n}\|y\|_2^2 - \epsilon\right]$$

and letting $\epsilon = \alpha \frac{m}{n}\|y\|_2^2$, we then have that this probability is bounded by

$$\exp\left(\frac{-2\alpha^2 \left(\frac{m}{n}\right)^2 \|y\|_2^4}{4m\|y\|_\infty^4}\right)$$

Thus, the resulting probability bound is

$$\mathbb{P}\left[\|y_\Omega\|_2^2 \geq (1-\alpha)\frac{m}{n}\|y\|_2^2\right] \geq 1 - \exp\left(\frac{-\alpha^2 m\|y\|_2^4}{2n^2\|y\|_\infty^4}\right)$$

Substituting our definitions of $\mu(y)$ and α shows that the lower bound holds with probability at least $1-\delta$. The argument for the upper bound is identical after replacing Equation (3.16) instead of (3.17). The Lemma now follows by applying the union bound. □

Proof of Lemma 3.3. We use McDiarmid's inequality in a very similar fashion to the proof of Lemma 3.2. Let $X_i = y_{\Omega(i)}U_{\Omega(i)}$, where $\Omega(i)$ refers to the i^{th} sample index. Thus $y_{\Omega(i)}$ is a scalar, and the notation $U_{\Omega(i)}$ refers to an $d \times 1$ vector representing the transpose of the $\Omega(i)^{th}$ row of U.

Let our function $f(X_1, \ldots, X_m) = \|\sum_{i=1}^m X_i\|_2 = \|U_\Omega^T y_\Omega\|_2$. To find the t_i of the theorem we first need to bound $\|X_i\|$ for all i.

Observe that since U is orthogonal, and denoting $\Omega(i) = j$,

$$\begin{aligned}\|U_{\Omega(i)}\|_2^2 = \|U^T e_j\|_2^2 &= e_j^T UU^T e_j \\ &= e_j^T UU^T UU^T e_j \\ &= \|UU^T e_j\|_2^2\end{aligned}$$

We also have that $\|UU^T e_j\|_2 = \|P_{\mathcal{S}} e_j\|_2 \leq \sqrt{d\mu(\mathcal{S})/n}$ by assumption. Thus,

$$\begin{aligned}\|X_i\|_2 &= \|y_{\Omega(i)}U_{\Omega(i)}\|_2 \\ &\leq |y_{\Omega(i)}|\|U_{\Omega(i)}\|_2 \\ &\leq \|y\|_\infty\sqrt{d\mu(\mathcal{S})/n}\end{aligned}$$

Then observe $\left|f(X_1, \ldots, X_m) - f(X_1, \ldots, \hat{X}_k, \ldots, X_m)\right|$ is

$$\begin{aligned}\left|\left\|\sum_{i=1}^{m} X_i\right\|_2 - \left\|\sum_{i\neq k} X_i + \hat{X}_k\right\|_2\right| &\leq \left\|X_k - \hat{X}_k\right\|_2 \\ &\leq \|X_k\|_2 + \|\hat{X}_k\|_2 \\ &\leq 2\|y\|_\infty \sqrt{\frac{d\mu(\mathcal{S})}{n}}\,.\end{aligned}$$

Here, the first two inequalities follow from the triangle inequality. Next we calculate a bound for $\mathbb{E}\left[f(X_1,\ldots,X_m)\right] = \mathbb{E}\left[\|\sum_{i=1}^m X_i\|\right]$. Assume again that the samples are taken uniformly with replacement. We have

$$\sum_{k=1}^{d} U_{jk}^2 = \|U_j\|_2^2 = \|P_{\mathcal{S}} e_j\|_2^2 \leq \frac{d}{n}\mu(\mathcal{S})\,,$$

from which we can see that

$$\begin{aligned}
&\mathbb{E}\left[\left\|\sum_{i=1}^{m} X_i\right\|_2^2\right] = \mathbb{E}\left[\left\|U_\Omega^T y_\Omega\right\|_2^2\right] \\
&= \mathbb{E}\left[\sum_{k=1}^{d}\sum_{i=1}^{m} U_{\{\Omega(i),k\}} y_{\Omega(i)} \sum_{l=1}^{m} U_{\{\Omega(l),k\}} y_{\Omega(l)}\right] \\
&= \sum_{k=1}^{d} \mathbb{E}\left[\sum_{i=1}^{m}\sum_{j=1}^{n} U_{jk}^2 y_j^2 \mathbb{I}_{\{\Omega(i)=j\}}\right] \qquad (3.18)\\
&= \sum_{k=1}^{d}\sum_{i=1}^{m}\sum_{j=1}^{n} U_{jk}^2 y_j^2 \mathbb{E}\left[\mathbb{I}_{\{\Omega(i)=j\}}\right] \\
&= \sum_{k=1}^{d} m\left(\sum_{j=1}^{n} U_{jk}^2 y_j^2\right)\frac{1}{n} \qquad (3.19)\\
&= \frac{m}{n}\sum_{j=1}^{n}\left(\sum_{k=1}^{d} U_{jk}^2\right) y_j^2 \\
&\leq \frac{m}{n}\frac{d\mu(\mathcal{S})}{n}\|y\|_2^2
\end{aligned}$$

The step (3.18) follows because the cross terms cancel by orthogonality ($U^T y = 0$ since $y \in \mathcal{S}^\perp$). The step (3.19) is because of our assumption that sampling is uniform with replacement.

Since $\mathbb{E}\left[\|X\|_2\right] \leq \mathbb{E}\left[\|X\|_2^2\right]^{1/2}$ by Jensen's inequality, we have that $\mathbb{E}\left[\|\sum_{i=1}^m X_i\|_2\right] \leq \sqrt{\frac{m}{n}}\sqrt{\frac{d\mu(\mathcal{S})}{n}}\|y\|_2$. Letting $\epsilon = \beta\sqrt{\frac{m}{n}}\sqrt{\frac{d\mu(\mathcal{S})}{n}}\|y\|_2$ and plugging into Equation (3.16), we then have that the probability is bounded by

$$\exp\left(\frac{-2\beta^2\frac{m}{n}\frac{d\mu(\mathcal{S})}{n}\|y\|_2^2}{4m\|y\|_\infty^2\frac{d\mu(\mathcal{S})}{n}}\right)$$

Thus, the resulting probability bound is

$$\mathbb{P}\left[\|U_\Omega y_\Omega\|_2^2 \geq (1+\beta)^2\frac{md\mu(\mathcal{S})}{n^2}\|y\|_2^2\right] \leq \exp\left(\frac{-\beta^2\|y\|_2^2}{2n\|y\|_\infty^2}\right)$$

Substituting our definitions of $\mu(y)$ and β shows that the lower bound holds with probability at least $1-\delta$, completing the proof. □

Proof of Lemma 3.4. We use the Noncommutative Bernstein Inequality as follows. Let $X_k = U_{\Omega(k)}U_{\Omega(k)}^T - \frac{1}{n}I_d$, where the notation $U_{\Omega(k)}$ is as before, i.e. is the transpose of the $\Omega(k)^{th}$ row of U, and I_d is the $d \times d$ identity matrix. Note that this random variable is zero mean.

We must compute ρ_k^2 and M. Since $\Omega(k)$ is chosen uniformly with replacement, the X_k are identically distributed, and ρ does not depend on k. For ease of notation we will denote $U_{\Omega(k)}$ as U_k.

Using the fact that for positive semi-definite matrices, $\|A-B\|_2 \leq \max\{\|A\|_2, \|B\|_2\}$, and recalling again that $\|U_k\|_2^2 = \|U^T e_k\|_2^2 = \|P_{\mathcal{S}} e_k\|_2^2 \leq d\mu(\mathcal{S})/n$, we have

$$\left\|U_k U_k^T - \frac{1}{n}I_d\right\|_2 \leq \max\left\{\frac{d\mu(\mathcal{S})}{n}, \frac{1}{n}\right\}$$

and we let $M := d\mu(\mathcal{S})/n$.

For ρ, we note

$$\begin{aligned}
\left\|\mathbb{E}\left[X_k X_k^T\right]\right\|_2 &= \left\|\mathbb{E}\left[X_k^T X_k\right]\right\|_2 \\
&= \left\|\mathbb{E}\left[\left(U_k U_k^T - \frac{1}{n}I_d\right)^2\right]\right\|_2 \\
&= \left\|\mathbb{E}\left[U_k U_k^T U_k U_k^T - \frac{2}{n}U_k U_k^T + \frac{1}{n^2}I_d\right]\right\|_2 \\
&= \left\|\mathbb{E}\left[U_k U_k^T U_k U_k^T\right] - \frac{1}{n^2}I_d\right\|_2 \qquad (3.20)
\end{aligned}$$

Step (3.20) is achieved as follows. First apply linearity of expectation to the preceding line to get

$$\left\| \mathbb{E}\left[U_k U_k^T U_k U_k^T\right] - \frac{2}{n}\mathbb{E}\left[U_k U_k^T\right] + \frac{1}{n^2} I_d \right\|_2 .$$

We can write the middle term as

$$\begin{aligned} \frac{2}{n}\mathbb{E}\left[U_k U_k^T\right] &= \frac{2}{n}\mathbb{E}\left[\sum_{j=1}^{n} U_j U_j^T \mathbb{I}_{\{\Omega(k)=j\}}\right] \\ &= \frac{2}{n} U^T U \frac{1}{n} \\ &= \frac{2}{n^2} I_d \end{aligned} \tag{3.21}$$

where step 3.21 follows simply from the definition of U_j as the transpose of the j^{th} row of U.

For the next steps we note that

$$\begin{aligned} \|E[U_k (U_k^T U_k) U_k^T]\|_2 &= \left\|E[U_k (\|U_k\|_2^2) U_k^T]\right\|_2 \\ &= \|U_k\|_2^2 \|E[U_k U_k^T]\|_2 \\ &\leq \frac{d\mu(\mathcal{S})}{n} \|E[U_k U_k^T]\|_2 \\ &= \frac{d\mu(\mathcal{S})}{n^2} \|I_d\|_2 = \frac{d\mu(\mathcal{S})}{n^2} \end{aligned}$$

Bringing all this together with the fact that $\|A - B\|_2 \leq \max\{\|A\|_2, \|B\|_2\}$ for positive semi-definite marices, we have

$$\begin{aligned} \left\|\mathbb{E}\left[X_k X_k^T\right]\right\|_2 &= \left\| \mathbb{E}\left[U_k U_k^T U_k U_k^T\right] - \frac{1}{n^2} I_d \right\|_2 \\ &\leq \max\left\{ \left\|\mathbb{E}\left[U_k U_k^T U_k U_k^T\right]\right\|, \frac{1}{n^2} \right\} \\ &\leq \max\left\{ \frac{d\mu(\mathcal{S})}{n} \|E[U_k U_k^T]\|_2, \frac{1}{n^2} \right\} \\ &= \max\left\{ \frac{d\mu(\mathcal{S})}{n^2}, \frac{1}{n^2} \right\} \\ &= \frac{d\mu(\mathcal{S})}{n^2} . \end{aligned}$$

Thus we let $\rho^2 := d\mu(\mathcal{S})/n^2$.

Now we can apply the Noncommutative Bernstein Inequality, Theorem 3.4. First we restrict τ to be such that $M\tau \leq m\rho^2$ to simplify the denominator of the exponent. Then we get that

$$2r \exp\left(\frac{-\tau^2/2}{m\rho^2 + M\tau/3}\right) \leq 2r \exp\left(\frac{-\tau^2/2}{\frac{4}{3}m\frac{d\mu(\mathcal{S})}{n^2}}\right)$$

and thus

$$\mathbb{P}\left[\left\|\sum_{k\in\Omega}\left(U_k U_k^T - \frac{1}{n}I_d\right)\right\| > \tau\right] \leq 2d \exp\left(\frac{-3n^2\tau^2}{8md\mu(\mathcal{S})}\right)$$

Now take $\tau = \gamma m/n$ with γ defined in the statement of Theorem 3.1. Since $\gamma < 1$ by assumption, $M\tau \leq m\rho^2$ holds and we have

$$\mathbb{P}\left[\left\|\sum_{k\in\Omega}\left(U_k U_k^T - \frac{1}{n}I_d\right)\right\|_2 \leq \frac{m}{n}\gamma\right] \geq 1 - \delta$$

We note that $\left\|\sum_{k\in\Omega} U_k U_k^T - \frac{m}{n}I_d\right\|_2 \leq \frac{m}{n}\gamma$ implies that the minimum singular value of $\sum_{k\in\Omega} U_k U_k^T$ is at least $(1-\gamma)\frac{m}{n}$.

To see this, let λ_i be the i^{th} eigenvalue and consider that

$$\begin{aligned}
\sigma_{max}^2\left(\sum_{k\in\Omega} U_k U_k^T - \frac{m}{n}I_d\right) &= \lambda_{max}\left(\left(\sum_{k\in\Omega} U_k U_k^T - \frac{m}{n}I_d\right)^T\left(\sum_{k\in\Omega} U_k U_k^T - \frac{m}{n}I_d\right)\right) \\
&= \max_i\left(\lambda_i\left(\sum_{k\in\Omega} U_k U_k^T - \frac{m}{n}I_d\right)^2\right) \qquad (3.22) \\
&= \max_i\left(\lambda_i\left(\sum_{k\in\Omega} U_k U_k^T\right) - \frac{m}{n}\right)^2
\end{aligned}$$

where Step 3.22 is due to the fact that $\sum_{k\in\Omega} U_k U_k^T$ is a symmetric matrix. Then we have that

$$\begin{aligned}
\sigma^2_{max}\left(\sum_{k\in\Omega} U_k U_k^T - \frac{m}{n} I_d\right) &= \max_i \left(\lambda_i\left(\sum_{k\in\Omega} U_k U_k^T\right) - \frac{m}{n}\right)^2 \leq \left(\frac{m}{n}\gamma\right)^2 \\
&\Longrightarrow \quad \left|\lambda_i\left(\sum_{k\in\Omega} U_k U_k^T\right) - \frac{m}{n}\right| \leq \frac{m}{n}\gamma \quad \forall i \\
&\Longrightarrow \quad \lambda_i\left(\sum_{k\in\Omega} U_k U_k^T\right) \geq \frac{m}{n}(1-\gamma) \quad \forall i
\end{aligned}$$

This in turn implies that $\left\|\left(\sum_{k\in\Omega} U_k U_k^T\right)^{-1}\right\|_2 \leq \frac{n}{(1-\gamma)m}$, which completes the proof. □

Chapter 4

Subspace Estimation with Missing Data

The popular problem of matrix completion [22, 23] can be viewed as static subspace estimation with missing data. To address this problem using the tools from Chapter 3, we now develop GROUSE (Grassmannian Rank-One Update Subspace Estimation), a subspace identification and tracking algorithm that builds high quality estimates from very sparsely sampled vectors [6]. GROUSE implements an incremental gradient procedure with computational complexity linear in dimensions of the problem, and is therefore scalable to very high-dimensional applications. An additional feature of GROUSE is that it can be immediately adapted to an 'online' version of the matrix completion problem, where one aims to recover a low-rank matrix from streaming column vectors and from small random subsets of the entries of those columns. GROUSE is not only remarkably efficient for online matrix completion, but additionally enables incremental updates as columns are added or entries are incremented over time. These features are particularly attractive for maintaining databases of user preferences and collaborative filtering.

In this chapter we explore the theoretical properties of GROUSE as well as its behavior in numerical simulation.

4.1 Problem Formulation

Using the tools of the incomplete data projection from Chapter 3, we can adapt the classical subspace estimation and tracking algorithms to deal with missing data. Several classical incremental subspace estimation algorithms have the vector residual as a centerpiece; in Section 4.2.2 we discuss how our

algorithm has relationships to both incremental SVD algorithms and gradient descent subspace estimation algorithms.

We can handle missing data using the tools from Chapter 3; the residual on entries seen is the indicator of the goodness of fit. It can also indicate what directions are missing from the subspace estimate, at least on the entries observed.

We aim to estimate an d-dimensional subspace of $\mathbb{R}^n$ denoted by $\mathcal{S}[t]$. At every time t, we observe a vector $v_t \in \mathcal{S}[t]$ at locations $\Omega_t \subset \{1, \dots n\}$. The subspace $\mathcal{S}[t]$ may evolve over time, and we address that more specifically in Chapter 6, Section 6.1. We will measure the error of our subspace using the squared Euclidean distance from our current subspace estimate $\mathcal{S}_{\text{est}}[t]$ to the observed vector v_t only on the coordinates revealed in the set in Ω_t. We can compute this distance explicitly for any subspace S using our formulation in Section 3.1.

Let U be any matrix whose columns span $\mathcal{S}$. Let U_{Ω_t} denote the submatrix of U consisting of the rows indexed by Ω_t. For a vector $v \in \mathbb{R}^n$, let v_{Ω_t} be the vector of only the observed entries of v; that is $v_{\Omega_t} \in \mathbb{R}^{|\Omega_t|}$ has the entries of v indexed by Ω_t. Then we have our cost function:

$$F(\mathcal{S}; t) = \min_a \|v_{\Omega_t} - U_{\Omega_t} a\|^2 \tag{4.1}$$

We will use this definition in Section 4.2 to derive our algorithm. If the matrix $U_{\Omega_t}^T U_{\Omega_t}$ has full rank, then we must have that $a = (U_{\Omega_t}^T U_{\Omega_t})^{-1} U_{\Omega_t}^T v_{\Omega_t}$ achieves the minimum in 4.1. Thus,

$$F(\mathcal{S}; t) = v_{\Omega_t}^T (I - U_{\Omega_t} (U_{\Omega_t}^T U_{\Omega_t})^{-1} U_{\Omega_t}^T) v_{\Omega_t} \, .$$

In the special case where the subspace is time-invariant, that is $\mathcal{S}[t] = \mathcal{S}_0$ for some fixed subspace S_0, then it is natural to consider the average cost function

$$\bar{F}(\mathcal{S}) := \sum_{t=1}^{T} \min_a \|v_{\Omega_t} - U_{\Omega_t} a\|^2 \, . \tag{4.2}$$

The average cost function will allow us to estimate the steady-state behavior of our algorithm. Indeed, in the static case, our algorithm will be guaranteed to converge to a stationary point of $\bar{F}(\mathcal{S})$.

4.1.1 Relation to Matrix Completion

In the scenario where the subspace does not evolve over time and we only observe vectors on a finite time horizon, then the cost function 4.2 is identical to the matrix completion optimization problem studied in [57, 33]. To see the equivalence, let $\Omega = \{(k,t) \ : \ k \in \Omega_t \, 1 \leq t \leq T$, and let $V = [v_1, \ldots, v_T]$. Then

$$\begin{aligned} \bar{F}(\mathcal{S}) &= \sum_{t=1}^{T} \min_a \|v_{\Omega_t} - U_{\Omega_t} a\|^2 \\ &= \min_{A \in \mathbb{R}^{d \times T}} \sum_{(i,j) \in \Omega} (V - UA)_{ij}^2 \end{aligned}$$

That is, the global optimization problem can be written as $\min_{U,A} \sum_{(i,j)\in\Omega}(V - UA)_{ij}^2$, which is precisely the starting point for the algorithms and analyses in [57, 33]. The authors in [57] use a gradient descent algorithm to jointly minimize both U and A while [33] minimizes this cost function by first solving for A and then taking a gradient step with respect to U. In the present work, we consider optimizing this cost function one column at a time. We show that by using our online algorithm, where each measurement v_t corresponds to a random column of the matrix V, we achieve state-of-the-art or better performance on matrix completion problems (see Section 4.5.3).

4.2 Algorithm: Grassmannian Rank-One Update Subspace Estimation

The set of all subspaces of $\mathbb{R}^n$ of dimension d is denoted $\mathfrak{G}(n,d)$ and is called the *Grassmannian*. The Grassmannian is a compact Riemannian manifold, and its geodesics can be explicitly computed [37]. An element $S \in \mathfrak{G}(n,d)$ can be represented by any $n \times d$ matrix U whose columns form an orthonormal basis for S. Our algorithm derives from an application of incremental gradient descent [11] on the Grassmannian. We first compute a gradient of the cost function F, and then follow this gradient along a short geodesic curve in the Grassmannian.

We follow the program developed in [37]. To compute the gradient of F on the Grassmannian, we

first need to compute the partial derivatives of F with respect to the components of U. From Lemma 3.4, the matrix $U_{\Omega_t}^T U_{\Omega_t}$ has full rank provided with probability $1-\delta$ provided that $|\Omega_t| \geq \frac{8}{3} d\mu(S) \log\left(\frac{2d}{\delta}\right)$, and hence the cost function 4.1 is differentiable almost everywhere with the same probability. Let Δ_{Ω_t} be the $n \times n$ diagonal matrix which has 1 in the j^{th} diagonal entry if $j \in \Omega_t$ and has 0 otherwise[1]. We can rewrite

$$F(\mathcal{S}; t) = \min_a \|\Delta_{\Omega_t}(v_t - Ua)\|^2 ;$$

taking the derivative of F with respect to the elements of U, as derived in the Appendix A.1, the result is

$$\begin{aligned} \frac{dF}{dU} &= -2(\Delta_{\Omega_t}(v_t - Uw))w^T \\ &= -2v_\perp w^T \end{aligned} \tag{4.3}$$

where $v_\perp := \Delta_{\Omega_t}(v_t - Uw)$ denotes the (zero padded) residual vector and w is the least-squares solution in 4.1.

Using Equation (2.70) in [37], we can calculate the gradient on the Grassmannian from this partial derivative

$$\begin{aligned} \nabla F &= (I - UU^T)\frac{dF}{dU} \\ &= -2(I - UU^T)v_\perp w^T = -2v_\perp w^T . \end{aligned}$$

The final equality follows because the residual vector $v_\perp$ is orthogonal to all of the columns of U. This can be verified from the definitions of $v_\perp$ and w.

A gradient step along the geodesic with tangent vector $-\nabla F$ is given by Equation (2.65) in [37], which we repeat here. Let the singular value decomposition of $\nabla F = YSZ^T$. Then for a step of length η in the direction of ∇F is given by

$$U(\eta) = UZ\cos(S\eta)Z^T + Y\sin(S\eta)Z^T .$$

[1]We note here that we are now considering Ω to be a subset of $\{1, \ldots, n\}$ and not a multiset. This is meant to be general for *any* subset of entries; but to leverage the tools from Chapter 3 we could assume the entries could be sampled without replacement or with independent Bernoulli random sampling.

It is trivial to compute the singular value decomposition of ∇F, as it is rank one. The sole non-zero singular value is $\sigma = 2||v_\perp||||w||$ and the corresponding left and right singular vectors are $\frac{v_\perp}{\|v_\perp\|}$ and $\frac{w}{\|w\|}$ respectively. Let $x_2, \ldots x_d$ be an orthonormal set orthogonal to $v_\perp$ and $y_2, \ldots, y_d$ be an orthonormal set orthogonal to w. Then

$$-2v_\perp w^T = \begin{bmatrix} -\frac{v_\perp}{\|v_\perp\|} & x_2 & \ldots & x_d \end{bmatrix} \times \mathrm{diag}(\sigma, 0, \ldots, 0) \times \begin{bmatrix} \frac{w}{\|w\|} & y_2 & \ldots & y_d \end{bmatrix}^T$$

forms an SVD for the gradient. Now using (2.65) from [37], we find that for $\eta > 0$, a step of length η in the direction ∇F is given by (See Appendix A.2):

$$\begin{aligned} U(\eta) &= U + \frac{(\cos(\sigma\eta) - 1)}{\|w\|^2} U w w^T + \sin(\sigma\eta) \frac{v_\perp}{\|v_\perp\|} \frac{w^T}{\|w\|} \\ &= U + \left(\sin(\sigma\eta) \frac{v_\perp}{\|v_\perp\|} + (\cos(\sigma\eta) - 1) \frac{v_{||}}{\|v_{||}\|} \right) \frac{w^T}{\|w\|} \end{aligned}$$

where $v_{||} := Uw$, the predicted value of the projection of the vector v_t onto $\mathcal{S}$.

This geodesic update rule is remarkable for a number of reasons. First of all, it consists only of a rank-one modification of the current subspace basis U, which involves low computational complexity. Second, compared to other algorithms for subspace estimation with missing data, GROUSE is highly computationally efficient is because it avoids the computation of the SVD altogether. Third, the term $\frac{\sin(\sigma\eta)}{\|v_\perp\|\|w\|} = \frac{\sin(\sigma\eta)}{\sigma}$ is on the order of η when $\sigma\eta$ is small. That is, for small values of σ and η this expression looks like a normal step along the gradient direction $-2v_\perp w^T$ given by 4.3. Stationary points exist when $\sigma = 0$. That is, there is no update either when $\|v_\perp\| = 0$, implying that the data vector v was already part of the estimated subspace; or when $\|w\| = 0$, implying that the estimated subspace is orthogonal to v_{Ω_t}. In Section 4.2.2 we discuss how this iterate relates to more familiar iterative algorithms from linear algebra which use full information; In Section 4.3, we explore further insights into this update equation.

4.2.1 Complexity

The GROUSE algorithm simply follows geodesics along the gradients of F with a prescribed set of step-sizes η. The full computation is summarized in Algorithm 1. Each step of GROUSE can be performed efficiently with standard linear algebra packages. Computing the weights in Step 2 of Algorithm 1 requires solving a least squares problem in $|\Omega_t|$ equations and d unknowns. Such a system is solvable in at most $O(|\Omega_t|d^2)$ flops in the worst case. Predicting the component of v that lies in the current subspace requires a matrix vector multiply that can be computed in $O(nd)$ flops. Computing the residual then only requires $O(|\Omega_t|)$ flops, as we will always have zeros in the entries indexed by the complement of Ω_t. Computing the norms of $v_\perp$ and $v_{||}$ can be done in $O(n)$ flops. The final subspace update consists of adding a rank one matrix to an $n \times d$ matrix and can be computed in $O(nd)$ flops. Totaling all of these computation times gives an overall complexity estimate of $O(nd + |\Omega_t|d^2)$ flops per subspace update.

Algorithm 1 Grassmannian Rank-One Update Subspace Estimation

Require: An $n \times d$ orthogonal matrix U_0. A sequence of vectors v_t, each observed in entries Ω_t. A set of stepsizes η_t.

1: **for** $t = 1, \ldots, T$ **do**
2: **Estimate weights:** $w = \arg\min_a \|\Delta_{\Omega_t}(v_t - U_t a)\|^2$
3: **Predict full vector:** $v_{||} = U_t w$
4: **Compute residual:** $v_\perp = \Delta_{\Omega_t}(v_t - v_{||})$
5: **Update subpace**:
$$U_{t+1} = U_t + \left((\cos(\sigma\eta_t) - 1)\frac{v_{||}}{\|v_{||}\|} + \sin(\sigma\eta_t)\frac{v_\perp}{\|v_\perp\|}\right)\frac{w^T}{\|w\|}$$
where $\sigma = \|v_\perp\|\|v_{||}\|$
6: **end for**

4.2.2 Comparison to methods that have no missing data

We discuss two ways to interpret the GROUSE algorithm in relation to well-known algorithms: incremental update of the SVD and subspace tracking using an LMS algorithm. Other related work can be found in Section 2.

Linear Algebraic View

GROUSE can be understood as an adaptation of an incremental update to a QR or SVD factorization. Most batch subspace identification algorithms that rely on the eigenvalue decomposition, the singular value decomposition, or their more efficient counterparts such as the QR decomposition or the Lanczos method, can be adapted for on-line updates and tracking of the principal subspace. A comprehensive survey of these methods can be found in [31].

Suppose we fully observe the vector v at each increment. Given an estimated basis, U_{est} for the unknown subspace S, we would update our estimate for S by computing the component of v that is orthogonal to U. We would then append this new orthogonal component to our basis U, and use an update rule based on the magnitude of v that does not lie in the span of U. Brand [19] has shown that this method can be used to compute an efficient incremental update of an SVD using optimized algorithms for computing rank one modifications of matrix factorizations [48].

In lieu of being able to exactly compute the component of v_t that is orthogonal to our current subspace estimate, GROUSE computes this component only on the entries in Ω_t. In Section 3.2.1 we had indications that the estimate for $v_\perp$ computed by Algorithm 1 in a single iteration is a good proxy for full-data residual. One can also verify that the GROUSE update rule corresponds to forming the matrix $[U, v_\perp]$ and then truncating the last column of the matrix

$$[U, v_\perp] R_\eta$$

where R_η denotes the $(d+1) \times (d+1)$ rotation matrix

$$R_\eta = \begin{bmatrix} I - \frac{ww'}{\|w\|}(1 - \cos(\eta\sigma)) & -\frac{w}{\|w\|}\sin(\eta\sigma) \\ \frac{w}{\|w\|}\sin(\eta\sigma) & \cos(\eta\sigma) \end{bmatrix}.$$

That is, our algorithm computes a mixture of the current subspace estimate and the incomplete data residual. This mixture is determined both by the stepsize and $\sigma = \|v_\perp\|\|v_{||}\|$.

Gradient-descent based methods

Here we discuss in particular the least-mean-square (LMS)-type algorithm derived in [110, 109] and studied in [34]. In Section 2 we also discuss the regularized least-squares (RLS)-type[2] variant called Projection Approximation Subspace Tracking [109].

The author of [109] first introduces the LMS algorithm using exactly the same cost function as given in Equation 4.1 but with full data and not requiring orthonormality of the column space. Equation (A3) of [109] gives the gradient, which we now repeat here, using notation W for the subspace variable; the columns of W span the subspace but have no orthogonality constraint. Substituting the correlation matrix $C = \mathbb{E}[vv^T]$ with its instantaneous estimate $\widehat{C} = vv^T$, they derive the gradient as

$$\frac{1}{2}\nabla F = [-2vv^T + vv^T WW^T + WW^T vv^T]W \quad .$$

If W were orthogonal, we could use GROUSE notation U instead, and this would in fact be equivalent to the gradient we have derived for GROUSE: $2[-vv^T U + UU^T vv^T U] = (-v + Uw)w^T = -2v_\perp w^T$. The LMS update is given in [34] as Equation (5) as $W_{t+1} = W_t + \lambda_t[-2v_t v_t^T + v_t v_t^T W_t W_t^T + W_t W_t^T v_t v_t^T]W_t$ with step-size λ_t, and [34] proves that even without the orthogonality constraint, W_t will converge to a matrix with orthonormal columns that span the subspace when $\lambda = \lambda(t) \overset{t\to\infty}{\longrightarrow} 0$ for stationary signals (i.e. when the subspace is static).

4.3 Proof of Convergence with no missing data

Now we prove that the GROUSE algorithm for subspace estimation converges to the true subspace in the asymptotic limit, with *no missing data and no noise*.

Let $\mathcal{S}_{true}$ represent the true d-dimensional subspace of $\mathbb{R}^n$, and let $U_{true} \in \mathbb{R}^{n\times d}$ be an orthonormal matrix whose columns span $\mathcal{S}_{true}$. We receive a sequence of vectors $v_1, v_2, \ldots, v_t, \cdots \in \mathcal{S}_{true}$,

[2] LMS and RLS are terms used in signal processing and adaptive filtering. See [91] Chapters 10-14 and Section 2.2.2.

and from that, using Algorithm 1, we generate a sequence of matrices $U_0, U_1, \ldots, U_t, \ldots$, which correspond[3] to subspace estimates $\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_t, \ldots$. We are interested in proving that this sequence converges to $\mathcal{S}_{true}$. Precisely, let the columns of orthonormal matrix $U_t \in \mathbb{R}_{n\times d}$ span the subspace $\mathcal{S}_t$. Then we are interested in showing that

$$\lim_{t\to\infty} \det\left(U_{true}^T U_t U_t^T U_{true}\right) = 1 \ . \tag{4.4}$$

In this section we analyze Algorithm 2, the full-data version of Algorithm 1, with respect to this criterion. To see why this is equivalent to convergence of a sequence of subspaces, consider the following Lemma.

Lemma 4.1. *The following three statements are equivalent, and are equivalent to 4.4.*

$$\lim_{t\to\infty} U_t U_t^T = U_{true} U_{true}^T \tag{4.5}$$

$$\lim_{t\to\infty} \sum_i \sigma_i (U_t^T U_{true})^2 = d \tag{4.6}$$

$$\lim_{t\to\infty} \sum_i \lambda_i (U_{true}^T U_t U_t^T U_{true}) = d \ . \tag{4.7}$$

Proof. We start by considering the meaning of 4.5. Since projection matrices uniquely define subspaces, then the entry-wise convergence of these matrices is a precise characterization of convergence of the corresponding subspaces.

Next note that for $i = 1, \ldots, d$, the singular values $\sigma_i(U_t^T U_{true}) \in [0, 1]$. This can be seen with the fact that for two $n \times n$ projection matrices A and B, $\|AB\|_2 \leq \|A\|_2 \|B\|_2 = 1$; i.e. the maximum singular value of the product of projection matrices is 1, and thus all the singular values are between zero and one.

[3] We remind the reader that the subspace which is spanned by the orthonormal columns of U_t is uniquely identified by the projection matrix $U_t U_t^T$. Any two (non-unique) orthonormal matrices whose columns span $\mathcal{S}_t$ will have the same unique resulting projection matrix.

The proof of the fact that Equations 4.5 and 4.6 are equivalent can be found using [28, 45]. Then 4.6 and 4.7 are equivalent by the relationship of singular values and eigenvalues, and 4.4 is equivalent by the definition of the determinant as the product of the eigenvalues. □

Algorithm 2 Full-data GROUSE *for analysis purposes only*

Require: An $n \times d$ orthogonal matrix U_0 such that all principal angles between U_0 and U_{true} are less than $\pi/2$. A sequence of vectors $v_t \in \mathcal{S}_{true} \subset \mathbb{R}^n$. A set of stepsizes η_t.

1: **for** $t = 1, \ldots, T$ **do**
2: **Estimate weights:** $w = \arg\min_a \|U_t a - v_t\|^2$
3: **Predict full vector:** $v_{||} = U_t w$
4: **Rewrite the basis:** Let the first column of a new matrix $\widetilde{U}_t$ be $v_{||}/\|v_{||}\|$, and fill out the rest of the columns using Gram-Schmidt such that $\widetilde{U}_t$ is orthonormal and spans the same subspace as U_t.
5: **Re-estimate weights:** $\widetilde{w}(1) = \|w\|_2$, and $w(i) = 0$ for $i \neq 1$.
6: **Compute residual:** $v_\perp = v_t - v_{||}$
7: **Update subpace**: $U_{t+1} = \widetilde{U}_t + \left((\cos(\gamma\eta_t) - 1) \frac{v_{||}}{\|v_{||}\|} + \sin(\gamma\eta_t) \frac{v_\perp}{\|v_\perp\|} \right) \frac{\widetilde{w}^T}{\|\widetilde{w}\|}$
where $\gamma = \|v_\perp\| \|v_{||}\|$
8: **end for**

This algorithm given in 2 is equivalent to Algorithm 1 when all data are observed. Steps 4 and 5 are new, and Step 7 uses $\widetilde{U}_t$ and $\widetilde{w}$ as opposed to U_t and w. However, since U_t and $\widetilde{U}_t$ span the same subspace, these changes do not change the algorithm. In fact it is unnecessary to execute steps 4 and 5, so one would not do this in practice, we are just writing it like this for purposes of analysis. Also note $v_{||} = U_t w = \widetilde{U}_t \widetilde{w}$. Finally, note that we changed notation to $\gamma = \|v_\perp\| \|v_{||}\|$, so as to avoid any confusion with the singular values $\sigma_i(U_t^T U_{true})$.

Let $y = \left(\cos(\gamma\eta_t) \frac{v_{||}}{\|v_{||}\|} + \sin(\gamma\eta_t) \frac{v_\perp}{\|v_\perp\|} \right)$ and use $\widetilde{U}(i)$ to denote the i^{th} column of $\widetilde{U}$. Then the update step is simply

$$U_{t+1} = \widetilde{U}_t - \frac{v_{||} \widetilde{w}^T}{\|v_{||}\| \|\widetilde{w}\|} + y \frac{\widetilde{w}^T}{\|\widetilde{w}\|} \tag{4.8}$$

Since $\widetilde{w}(i) = 0$ for $i \neq 1$, we have $\frac{\widetilde{w}^T}{\|\widetilde{w}\|} = [1\ 0 \ldots 0]^T$. This means that the update in Step 7 is simply replacing the first column of $\widetilde{U}_t$:

$$U_{t+1} = \begin{bmatrix} | & | & & | \\ \mathbf{0} & \widetilde{U}_t(2) & \cdots & \widetilde{U}_t(d) \\ | & | & & | \end{bmatrix} + \begin{bmatrix} | & | & & | \\ y & \mathbf{0} & \cdots & \mathbf{0} \\ | & | & & | \end{bmatrix} \tag{4.9}$$

or written slightly differently:

$$U_{t+1}(i) = \begin{cases} y & i = 1 \\ \widetilde{U}_t(i) & i \neq 1 \end{cases} . \tag{4.10}$$

4.3.1 Monotonic Increase of the Determinant

We use Equation 4.4 and show that $\det\left(U_{true}^T U_t U_t^T U_{true}\right)$ increases with every step of Algorithm 2. Note that $\det\left(U_{true}^T U_t U_t^T U_{true}\right) = \det\left(U_t^T U_{true}\right)^2$. We therefore wish to show that, except when $U_t = U_{t+1}$,

$$\left|\det\left(U_{t+1}^T U_{true}\right)\right| > \left|\det\left(U_t^T U_{true}\right)\right| .$$

We re-write U_{true} in the same way that we did for U_t in Step 4 of Algorithm 2 by orthonormalizing around v_t. That is, the first column of $\widetilde{U}_{true}$ is $v_t/\|v_t\|$, and the rest of the columns are populated using Gram-Schmidt such that $\widetilde{U}_{true}$ is orthonormal and spans the same subspace as U_{true}. We can then look at $U_t^T \widetilde{U}_{true}$ instead to show equivalently that:

$$\left|\det\left(U_{t+1}^T \widetilde{U}_{true}\right)\right| > \left|\det\left(\widetilde{U}_t^T \widetilde{U}_{true}\right)\right| .$$

We first state a Lemma that we will need for to prove this inequality holds.

This Lemma examines the relationship of $\left|y^T \frac{v_t}{\|v_t\|}\right|$ and $\left|\frac{v_{||}^T}{\|v_{||}\|} \frac{v_t}{\|v_t\|}\right|$. We wish to show that $\left|\frac{v_{||}^T v_t}{\|v_{||}\|\|v_t\|}\right| < \left|y^T \frac{v_t}{\|v_t\|}\right|$; i.e. that y is closer to v_t than is $v_{||}$. In the illustration in Figure 9, the shaded region gives the range for y (equivalently $\gamma\eta_t$) which will guarantee this inequality is satisfied. Let ψ be the angle between y and v_t, and choose $\gamma\eta_t \in (\phi - \pi/2, \phi + \pi/2)$ to restrict $\psi \in (0, \pi/2)$. Note that the

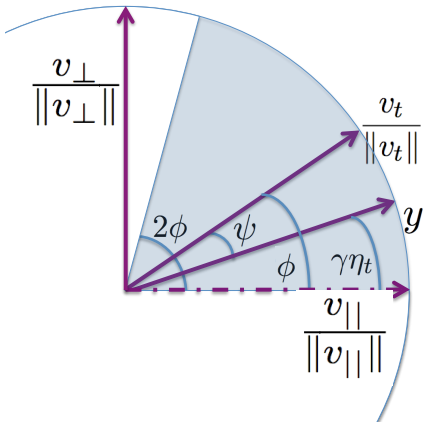


Figure 9: The relationship of various vectors and angles in the update and proofs. As the step size $\gamma\eta_t$ is swept from 0 to 2π, the vector y traces out the unit circle that contains both $v_\perp/\|v_\perp\|$ and $v_{||}/\|v_{||}$. The shaded region shows the range for y, i.e. for $\gamma\eta_t$, that will result in increasing the determinant in Theorem 4.1.

constraint we will impose in the Lemma that $\eta_t \in \left(0, \frac{2}{\|v_t\|^2}\right)$ is more restrictive than this. Therefore $y^T \frac{v_t}{\|v_t\|} > 0$, and we can drop the absolute value signs. Let ϕ be the angle between $v_{||}$ and v_t, that is $\cos(\phi) = \frac{v_{||}^T v_t}{\|v_{||}\|\|v_t\|}$. Note also $\phi \in (0, \pi/2)$ since $v_{||}$ is the projection of v_t, and so also $\frac{v_{||}^T v_t}{\|v_{||}\|\|v_t\|} > 0$.

Lemma 4.2. *Restrict* $\psi \in (0, \pi/2)$. *Then* $0 < \eta_t < \frac{2}{\|v_t\|^2}$ *implies* $\frac{v_{||}^T v_t}{\|v_{||}\|\|v_t\|} < y^T \frac{v_t}{\|v_t\|}$.

Proof. It is sufficient to prove that $\phi > \psi$: this implies $\cos(\phi) < \cos(\psi)$ and $\frac{v_{||}^T v_t}{\|v_{||}\|\|v_t\|} < y^T \frac{v_t}{\|v_t\|}$.

In order to prove $\phi > \psi$, it is equivalent to prove that $\gamma\eta_t < 2\phi$ (See Figure 9).

First consider $\phi \in [\frac{\pi}{4}, \frac{\pi}{2})$. For $\phi = \frac{\pi}{4}$, we have $\|v_\perp\| = \|v_{||}\| = \sqrt{\frac{\|v_t\|^2}{2}}$ and thus $\gamma = \|v_\perp\|\|v_{||}\| = \frac{\|v_t\|^2}{2}$. And for $\phi > \frac{\pi}{4}$, $\gamma < \frac{\|v_t\|^2}{2}$. So by the assumption of the Lemma, we have that

$$\eta_t < \frac{2}{\|v_t\|^2} < \frac{\pi}{\|v_t\|^2} < \frac{\pi}{2\gamma} < \frac{2\phi}{\gamma} \ .$$

Now consider $\phi \in [0, \frac{\pi}{4})$. In this region, $\sin(\phi) < \phi$. Note that $\sin(\phi) = \frac{\|v_\perp\|}{\|v_t\|}$.

$$\frac{2\phi}{\gamma} > \frac{2\sin(\phi)}{\gamma} = \frac{2}{\|v_\perp\|\|v_{||}\|}\frac{\|v_\perp\|}{\|v_t\|} > \frac{2}{\|v_t\|^2} > \eta_t \ ,$$

where we use that $\|v_{||}\| < \|v_t\|$ and again use the assumption of the Lemma. □

Now we are able to state our Theorem of monotonic increase of the determinant.

Theorem 4.1. *Let* $\eta_t \in \left(0, 2/\|v_t\|^2\right)$. *Then unless* $U_t = U_{t+1}$,

$$\left|det\left(U_{t+1}^T \widetilde{U}_{true}\right)\right| > \left|det\left(\widetilde{U}_t^T \widetilde{U}_{true}\right)\right| \quad .$$

Proof. We start by writing the products $\widetilde{U}_t^T \widetilde{U}_{true}$ and $U_{t+1}^T \widetilde{U}_{true}$ as follows.

$$\begin{aligned}
\widetilde{U}_t^T \widetilde{U}_{true} &= \begin{bmatrix} | & | & & | \\ v_{||}/\|v_{||}\| & \widetilde{U}_t(2) & \cdots & \widetilde{U}_t(d) \\ | & | & & | \end{bmatrix}^T \begin{bmatrix} | & | & & | \\ v_t/\|v_t\| & \widetilde{U}_{true}(2) & \cdots & \widetilde{U}_{true}(d) \\ | & | & & | \end{bmatrix} \\
U_{t+1}^T \widetilde{U}_{true} &= \begin{bmatrix} | & | & & | \\ y & \widetilde{U}_t(2) & \cdots & \widetilde{U}_t(d) \\ | & | & & | \end{bmatrix}^T \begin{bmatrix} | & | & & | \\ v_t/\|v_t\| & \widetilde{U}_{true}(2) & \cdots & \widetilde{U}_{true}(d) \\ | & | & & | \end{bmatrix}
\end{aligned}$$

Let A_t be the matrix $\begin{bmatrix} \widetilde{U}_t(2) & \cdots & \widetilde{U}_t(d) \end{bmatrix}$, and let B be the matrix $\begin{bmatrix} \widetilde{U}_{true}(2) & \cdots & \widetilde{U}_{true}(d) \end{bmatrix}$. Then $\widetilde{U}_t^T \widetilde{U}_{true} = \begin{bmatrix} v_{||}/\|v_{||}\| & A_t \end{bmatrix}^T \begin{bmatrix} v_t/\|v_t\| & B \end{bmatrix}$ and $U_{t+1}^T \widetilde{U}_{true} = \begin{bmatrix} y & A_t \end{bmatrix}^T \begin{bmatrix} v_t/\|v_t\| & B \end{bmatrix}$, and our block matrices are:

$$\begin{aligned}
\widetilde{U}_t^T \widetilde{U}_{true} &= \left[\begin{array}{c|c} \frac{v_{||}}{\|v_{||}\|}^T \frac{v_t}{\|v_t\|} & \frac{v_{||}}{\|v_{||}\|}^T B \\ \hline \mathbf{0} & A_t^T B \end{array}\right] \qquad (4.11) \\
U_{t+1}^T \widetilde{U}_{true} &= \left[\begin{array}{c|c} y^T \frac{v_t}{\|v_t\|} & y^T B \\ \hline \mathbf{0} & A_t^T B \end{array}\right]
\end{aligned}$$

The bottom left corner is $\mathbf{0}$ because $v_t \perp A_t$ by definition of the projection of v_t on $\widetilde{U}_t$.

$$\begin{aligned}
\det\left(\widetilde{U}_t^T \widetilde{U}_{true}\right) &= \left|\left[\begin{array}{c|c} \frac{v_{||}^T}{\|v_{||}\|}\frac{v_t}{\|v_t\|} & \frac{v_{||}^T}{\|v_{||}\|}B \\ \hline \mathbf{0} & A_t^T B \end{array}\right]\right| \\
&= \det\left(A_t^T B\right) \det\left(\frac{v_{||}^T}{\|v_{||}\|}\frac{v_t}{\|v_t\|} - \frac{v_{||}^T}{\|v_{||}\|}B(A_t^T B)^{-1}\cdot\mathbf{0}\right) \\
&= \det(A_t^T B)\frac{v_{||}^T}{\|v_{||}\|}\frac{v_t}{\|v_t\|}
\end{aligned}$$

and equivalently

$$\det\left(U_{t+1}^T \widetilde{U}_{true}\right) = \det(A_t^T B) y^T \frac{v_t}{\|v_t\|}$$

From here we use the result of Lemma 4.2 that $\eta_t < 2/\|v_t\|^2$ implies that $\left|y^T \frac{v_t}{\|v_t\|}\right| > \left|\frac{v_{||}^T}{\|v_{||}\|}\frac{v_t}{\|v_t\|}\right|$, completing the proof. □

4.3.2 Stationary Points

We seek stationary points for which the update step will have $U_t = U_{t+1}$ for all $t > T$ for some finite T. We require that the data vectors v_t are generated from the true subspace in such a way that there exists a τ so that for every collection of τ consecutive vectors $v_{t+1}, \ldots, v_{t+\tau}$,

$$\text{span}\left(v_{t+1}, \ldots, v_{t+\tau}\right) = \mathcal{S}_{true}\ . \tag{4.12}$$

This will hold, for example, with probability 1 for $\tau = d$ if the vectors v_t are drawn i.i.d. from an absolutely continuous distribution on $\mathcal{S}_{true}$.

Lemma 4.3. *Let U be a stationary point of Algorithm 2; i.e., $U_t = U_{t+1} = U$ for τ consecutive updates. Then either*

$$\textit{det}(U^T U_{true}) = 0 \ \ ,$$

or

$$\textit{det}(U^T U_{true}) = 1 \tag{4.13}$$

and note that all U which satisfy 4.13 span the subspace $\mathcal{S}_{true}$.

Proof. Stationary points exist wherever $U_{t+1} = U_t$ with our assumption in 4.12 whenever $U_{t+1} = U_t$ for some consecutive vectors $v_{t+1}, \ldots, v_{t+\tau}$. This is true when $w = 0$ or $v_{||} = v_t$ for $v_{t+1}, \ldots, v_{t+\tau}$.

For our global optimal point, $v_{||} = v_t$ for $v_{t+1}, \ldots, v_{t+\tau}$. Thus all the eigenvalues of $U^T U_{true}$ are one and $\det(U^T U_{true}) = 1$.

The next obvious collection of stationary points are those where $w = 0$ for all $v_{t+1}, \ldots, v_{t+\tau}$; that is, $U \perp U_{true}$. Finally, the remaining stationary points are those where $v_{||} = v_t$ for some of $v_{t+1}, \ldots, v_{t+\tau}$ and $w = 0$ for the others. For all these non-optimal stationary points, at least one vector or component of span(U) is orthogonal to $\mathcal{S}_{true}$. In this case, at least one eigenvalue of $U^T U_{true}$ will be zero. □

4.3.3 Convergence to the Global Optimal Stationary Point

As long as U_0 is initialized in such a way that the eigenvalues of $U_0^T U_{true}$ are all non-zero, then Theorem 4.1 and Lemma 4.3 imply that Algorithm 2 converges to the global optimal stationary point. This will hold if U_0 is drawn randomly from any distribution with zero measure on the stationary points of Lemma 4.3, which includes is any uniform distribution on the Grassmannian.

Theorem 4.2. *Algorithm 2 (and therefore Algorithm 1 with $\Omega_t = \{1, \ldots, n\}$, $\forall t$) converges to the global optimum point where $\textit{det}(U_t^T U_{true}) = 1$.*

Proof. We know from [12] that, being a gradient method, Algorithm 2 will converge to a stationary

point, as long as the step sizes are diminishing but not summable:

$$\eta_t \to 0, \qquad \sum_t \eta_t = \infty \ .$$

Since $\det(U_0^T U_{true}) > 0$, and the determinant is monotonically increasing, then we are guaranteed that the algorithm converges to the one stationary point with nonzero determinant, the global optimal point. □

4.4 Discussion of Convergence with missing data

We conjecture that GROUSE also converges with missing data under certain assumptions on the sampling. Obviously it is at least necessary that there exists a finite τ such that $0 < \|v_{\Omega_t} - P_{\mathcal{S}_{\Omega_t}} v_{\Omega_t}\|_2 < \|v_{\Omega_t}\|_2$ for at least one vector in every consecutive τ vectors, or else no GROUSE update would occur in that window. Whether this is sufficient, and what distributions on the data and the observation sets achieve such a condition, remains a crucial open question.

We could potentially extend the reasoning of the previous section in order to address whether the general GROUSE algorithm in Algorithm 1 converges with missing data. We could use the same final argument of Section 4.3.3 as long as we can similarly identify the stationary points of the missing-data cost function and show the monotonic increase of the GROUSE update step. Here we discuss our understanding of the potential extension of the previous arguments, and potential obstacles when addressing each of these issues.

4.4.1 Stationary Points

We start by considering the stationary points of the missing-data cost function. Again we require for a stationary point that $U_t = U_{t+1}$ for some τ consecutive vectors. We must have some way of requiring "enough" observations in our data vectors.

With missing data, it is possible that $v_{||} = v_t$ for all the data vectors even when the estimate given

by GROUSE is not the global optimal point. For example, consider an example where $d = 1$ and the number of samples $|\Omega_i| = 1$ for every vector v_i. Then it is possible to fit every vector to any randomly initialized subspace U_0, and the algorithm will never update. In this example any randomly drawn initial subspace is a stationary point of the algorithm with $v_{||} = v_t$.

It is also difficult to characterize the non-optimal stationary points where $w = 0$ when there are missing data. It is technically possible that the data vectors, on their observed coordinates, are all orthogonal to the estimated subspace U_t for τ consecutive vectors, even when the fully-observed vectors would not be orthogonal. This creates new stationary points which could trap the algorithm. Despite this, we see that with enough measurements the algorithm always does well in simulation; we believe there may be some simple assumptions under which this will not happen.

4.4.2 Monotonic Increase of the Determinant

Here we can analyze the algorithm with missing data using exactly the same approach as before; however, when we write our block matrices as in Equation 4.12 we have a slightly less simple expression.

With missing data our vector y is the vector from Algorithm 1:

$$y = \cos(\sigma\eta_t)\frac{v_{||}}{\|v_{||}\|} + \sin(\sigma\eta_t)\frac{v_\perp}{\|v_\perp\|}$$

where $v_\perp = 0$ on Ω^c.

The block matrices are thus:

$$\widetilde{U}_t^T \widetilde{U}_{true} = \left[\begin{array}{c|c} \frac{v_{||}}{\|v_{||}\|}^T \frac{v_t}{\|v_t\|} & \frac{v_{||}}{\|v_{||}\|}^T B \\ \hline A_t^T \frac{v_t}{\|v_t\|} & A_t^T B \end{array}\right] \tag{4.14}$$

$$U_{t+1}^T \widetilde{U}_{true} = \left[\begin{array}{c|c} y^T \frac{v_t}{\|v_t\|} & y^T B \\ \hline A_t^T \frac{v_t}{\|v_t\|} & A_t^T B \end{array}\right]$$

The bottom left corner is no longer $\mathbf{0}$, because $v_{\Omega_t} \perp A_t$ after the projection, but the same cannot be said for v_t. Continuing we have that

$$\begin{aligned} \det\left(\widetilde{U}_t^T \widetilde{U}_{true}\right) &= \left|\left[\begin{array}{c|c} \frac{v_{||}^T}{\|v_{||}\|} \frac{v_t}{\|v_t\|} & \frac{v_{||}^T}{\|v_{||}\|} B \\ \hline A_t^T \frac{v_t}{\|v_t\|} & A_t^T B \end{array}\right]\right| \\ &= \det\left(A_t^T B\right) \det\left(\frac{v_{||}^T}{\|v_{||}\|}\frac{v_t}{\|v_t\|} - \frac{v_{||}^T}{\|v_{||}\|} B (A_t^T B)^{-1} A_t^T \frac{v_t}{\|v_t\|}\right) \end{aligned}$$

and equivalently

$$\det\left(U_{t+1}^T \widetilde{U}_{true}\right) = \det\left(A_t^T B\right) \det\left(y^T \frac{v_t}{\|v_t\|} - y^T B (A_t^T B)^{-1} A_t^T \frac{v_t}{\|v_t\|}\right) .$$

Defining $D_t = \frac{v_t}{\|v_t\|} - B(A_t^T B)^{-1} A_t^T \frac{v_t}{\|v_t\|}$, we have

$$\det\left(\widetilde{U}_t^T \widetilde{U}_{true}\right) = \det\left(A_t^T B\right) \det\left(\frac{v_{||}^T}{\|v_{||}\|} D_t\right) \tag{4.15}$$

$$\det\left(U_{t+1}^T \widetilde{U}_{true}\right) = \det\left(A_t^T B\right) \det\left(y^T D_t\right) . \tag{4.16}$$

These expressions provide a starting point for future investigation into requirements that will guarantee convergence with missing data.

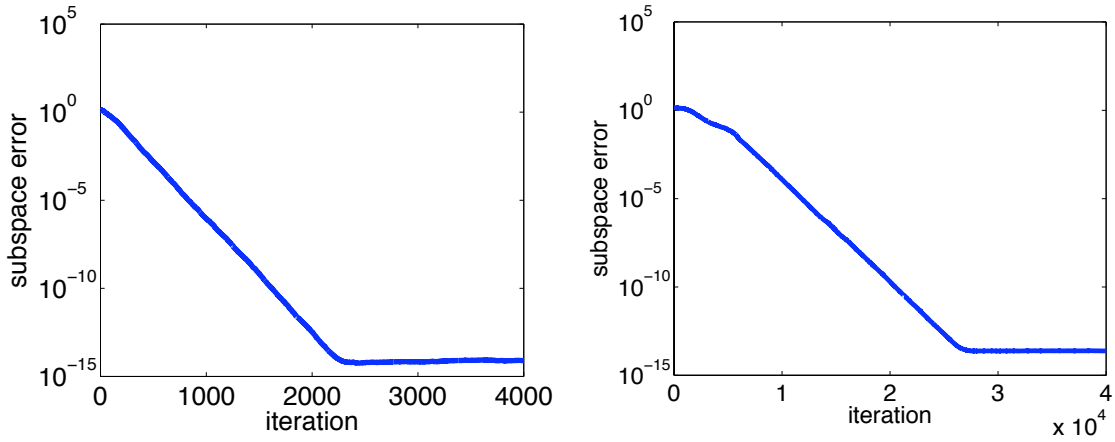


Figure 10: Missing data convergence for an incoherent subspace. For both, the inherent dimension $d = 5$ and the number of samples per vector $|\Omega| = 20$. On the left $n = 100$, on the right $n = 1000$.

4.4.3 Empirical Evidence

In conclusion, we provide two empirical results which are representative and which motivate further work on the question of convergence. In both we have plotted the Frobenius norm of the difference between the projection matrices (i.e. as in Equation 4.5 as opposed to the determinant), and so we expect to see the cost function decrease to zero.

In Figure 10, we have simulated a true subspace using orthonormalized Gaussian vectors, and therefore the resulting subspace is incoherent. The ambient dimension $n = 100$ (left) and $n = 1000$ (right), inherent dimension $d = 5$, and the number of samples per vector $|\Omega| = 20$. We see the cost function decreasing quickly to numerical precision. In Figure 11, the true subspace is a 5-dimensional subspace made of the first five coordinate vectors, i.e. the standard basis vectors. In this case, it takes many vectors before convergence; however we conjecture that we are still guaranteed convergence of the algorithm since the measurements are drawn uniformly, and thus over time the true subspace is properly sampled.

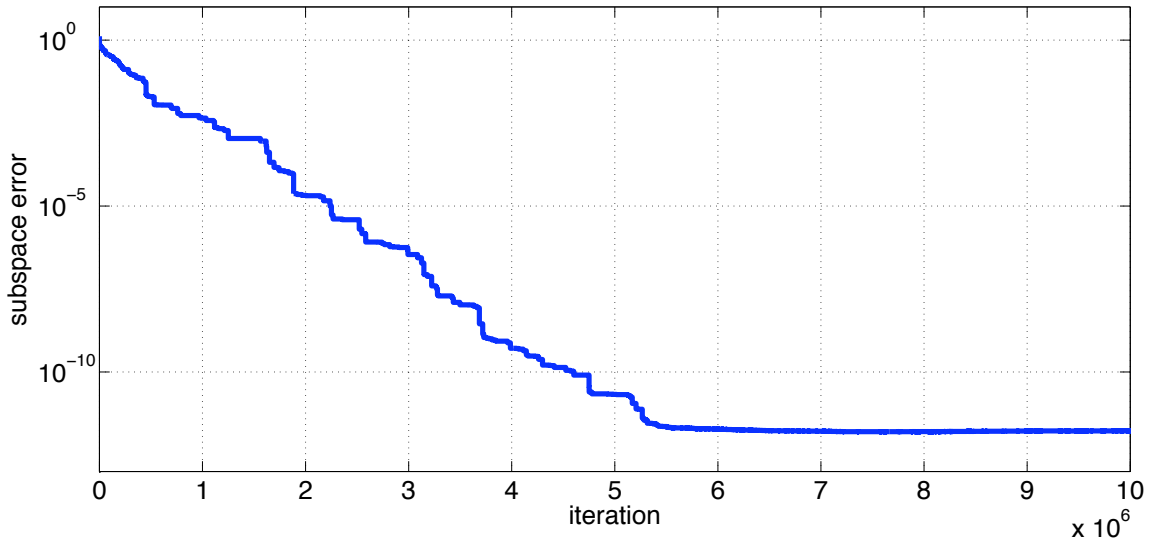


Figure 11: Missing data convergence when the true subspace is the first five standard basis vectors. $n = 100$, $d = 5$, $|\Omega| = 20$.

4.5 Empirical Analysis

GROUSE is implemented in Matlab using only basic operations. A single update of a 10 dimensional subspace of $\mathbb{R}^{100000}$ from a measurement density 10^{-3} takes less than two hundredths of a second on a MacBook laptop.

4.5.1 Static Subspace Estimation

We first consider the problem of identifying a fixed subspace. In all of the following experiments, the full data dimension is $n = 700$, the rank of the underlying subspace is $d = 10$, and the sampling density is 0.17 unless otherwise noted.

We generated a series of iid vectors v_t according to generative model:

$$v_t = U_{\text{true}}\alpha + \beta$$

where U_{true} is an $n \times d$ matrix whose d orthonormal columns spam the subspace, α is a $d \times 1$ vector whose entries are realizations of iid $\mathcal{N}(0, 1)$ random variables, and β is an $n \times 1$ vector whose entries

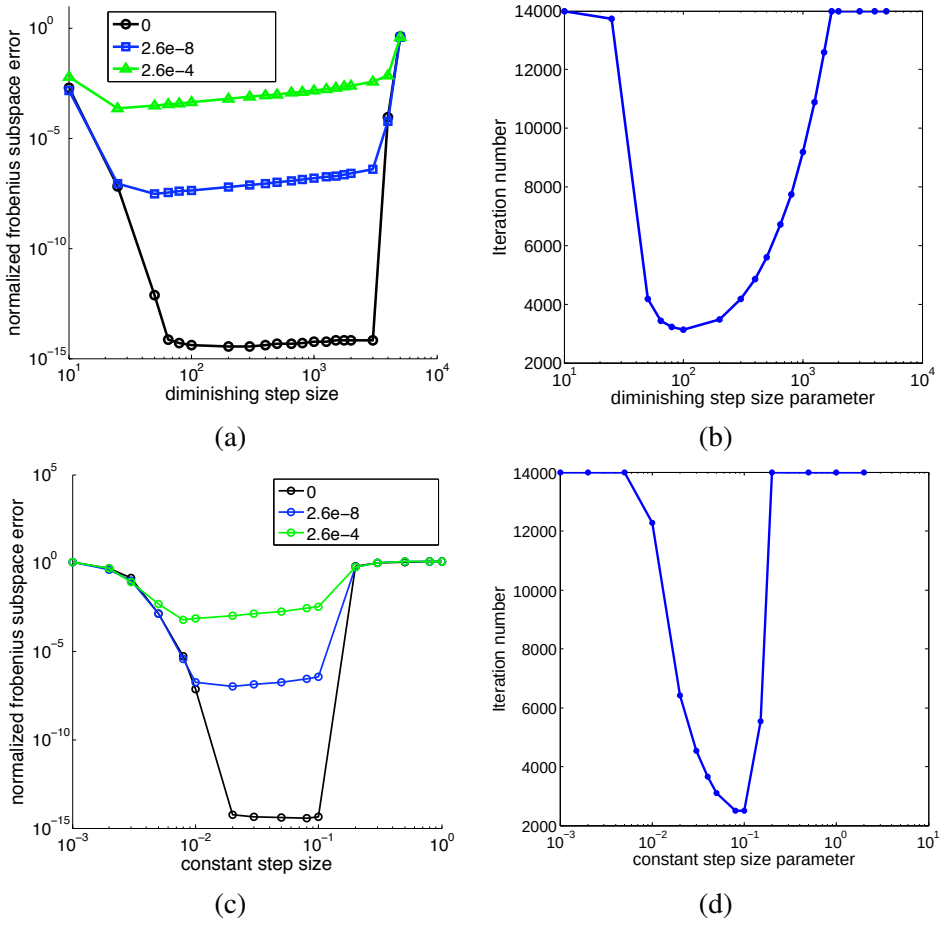


Figure 12: (a) Performance sensitivity to noise for a diminishing stepsize policy $\eta_t = C/t$. Results are displayed for three values of the noise magnitude, ω. (b) The number of iterations required to achieve an error of 10^{-6} as a function of C. (c) and (d) are the same except that the stepsize policy is here $\eta_t = C$.

are iid $\mathcal{N}(0, \omega^2)$. Here $\mathcal{N}(0, \omega^2)$ denotes the Gaussian distribution with mean zero and variance $\omega^2 > 0$ which governs the SNR of our data.

We implemented GROUSE (see Algorithm 1 above) with a stepsize rule of $\eta_t := C/t$ for some constant $C > 0$. Figure 12(a) shows the steady state error of the tracked subspace with varying values of C and the noise variance. All the data points reflect the error performance at $t = 14000$. We see that GROUSE converges for C ranging over an order of magnitude, however with additive noise the smaller stepsizes yield smaller errors. When there is no noise, i.e., $\omega^2 = 0$, the error performance is near the level of machine precision and is flat for the whole range of converging stepsizes. In Figure 12(b) we

show the number of input vectors after which the algorithm converges to an error of less than 10^{-6}. The results are consistent with Figure 12(a), demonstrating that smaller stepsizes in the suitable range take fewer vectors until convergence. We only ran the algorithm up to time $t = 14000$, so the data points for the smallest and the largest few stepsizes only reflect that the algorithm did not yet reach the desired error in the allotted time. Figures 12(c) and 12(d) repeat identical experiments with a constant stepsize policy, $\eta_t = C$. We again see a wide range of stepsizes for which GROUSE converges, though the region of stability is narrower in this case.

We note that the norm of the residual $\|v_\perp\|$ provides an excellent indicator for whether tracking is successful. A shown in Figure 24(a), the error to the true subspace is closely approximated by $\|v_\perp\|/\|v_t\|$. This confirms the theoretical analysis in [8] which proves that this residual norm is an accurate estimator of the true subspace error when the number of samples is appropriately large.

4.5.2 Static Subspaces and LMS Comparison

Here we compare GROUSE to the LMS algorithm described in Section 4.2.2. We first compare with fully observed vectors, and then we alter LMS to use the incomplete data residual. However even with the smallest amount of data missing, the altered LMS algorithm fails to identify the subspace appropriately.

Algorithm LMS implements LMS subspace estimation as given in [34], but using the incomplete data residual; in order to avoid a divergence of the entries of $W \to \infty$, we implemented a low-complexity column normalization step. We also implemented another version which orthogonalizes W after every update.

For these simulations, our parameters were $n = 100$, $d = 5$, and the number of vectors seen was $T = 5000$. No noise was added to the vectors; i.e. all $v_t \in S_{true}$. Results with full data are seen in Figure 13, which is averaged over 6 random problem instances. These results show that the performance of the two algorithms are similar until data are missing.

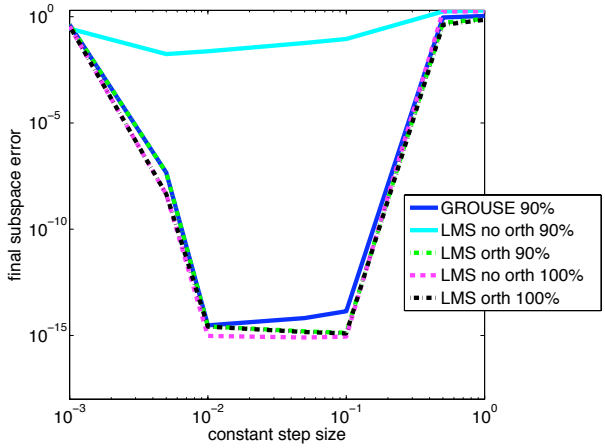


Figure 13: LMS versus GROUSE on subspace error after $T = 5000$ vectors, for ambient dimension $n = 100$ and subspace dimension $d = 5$. Even with just a small number of entries missing, the non-orthonormalized version of LMS completely fails.

In order to further investigate the behavior of LMS with missing data, we made the following two plots. We looked at LMS with fully observed vectors and 90% observed vectors; we also noted that since LMS does not orthogonalize the matrix W, the implementation of the solution to $W_{\Omega_t} w = v_{\Omega_t}$ may create different solutions, so we implemented it both with the backslash operator in MATLAB as well as with the pseudoinverse operator. As can be seen in Figure 14, when W is not orthonormalized and data are missing, the LMS algorithm fails to converge. Also, interestingly, we can see in Figure 15 that with both 100% and 90% sampling, the smallest singular value of W heads toward one after an initial dip, but only with 100% sampling does W become orthogonal.

4.5.3 Subspace Estimation for Matrix Completion

As described in Section 4.1.1, matrix completion can be thought of as a subspace identification problem where we aim to identify the column space of the unknown low-rank matrix. Once the column space is identified, we can project the incomplete columns onto that subspace in order to complete the matrix. We have examined GROUSE in this context with excellent results. Our approach is to do descent on the column vectors in random order, and allow the algorithm to pass over those same incomplete columns

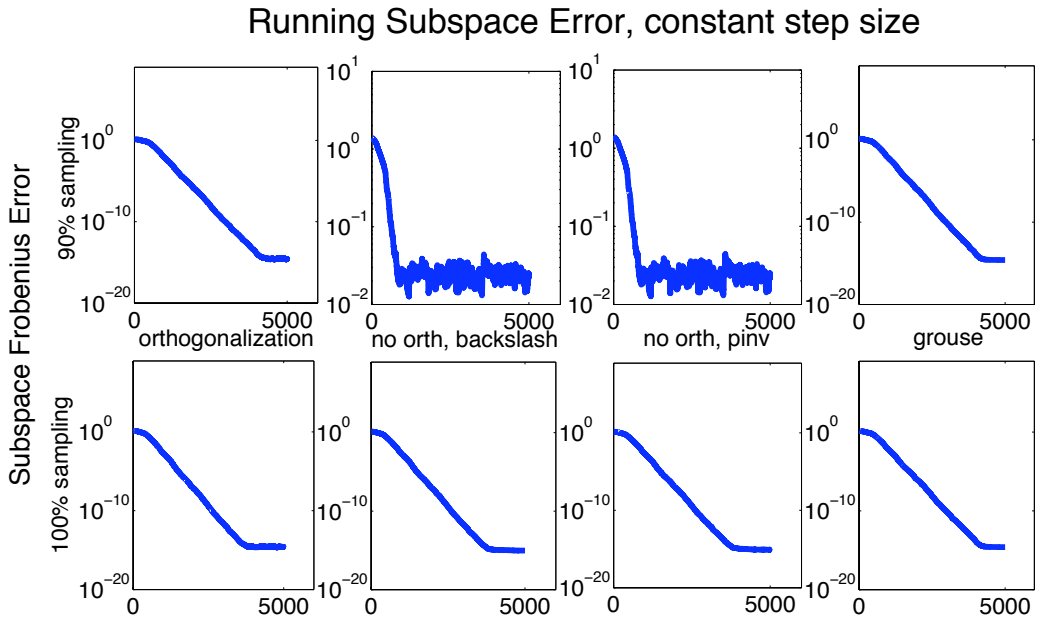


Figure 14: LMS versus GROUSE on vector-by-vector subspace error, for ambient dimension $n = 100$ and subspace dimension $d = 5$. Note the y-axis scale on the top center two plots is different from the others. Even with just a small number of entries missing, the non-orthonormalized version of LMS completely fails.

a few times.

Our simulation set-up aimed to complete 700×700 dimensional matrices of rank 10 sampled with density 0.17. We generated the low-rank matrix by generating two factors Y_L and Y_R with i.i.d. Gaussian entries, and added normally distributed noise with variance ω^2. The robustness to step-size and time to recovery are shown in Figure 16.

In Figure 17 we show a comparison of five matrix completion algorithms and GROUSE (labeled as Stochastic Gradient). Namely, we compare to the performance of OptSpace [57], FPCA [68], SVT [21], SDPLR [20, 86], and NNLS [99]. We downloaded each of these MATLAB codes from the original developer's websites when possible. We use the same random matrix model as in Figure 16. GROUSE is faster than all other algorithms, and achieves higher quality reconstructions on many instances. We subsequently compared against NNLS, the fastest batch method, on very large problems. Both GROUSE

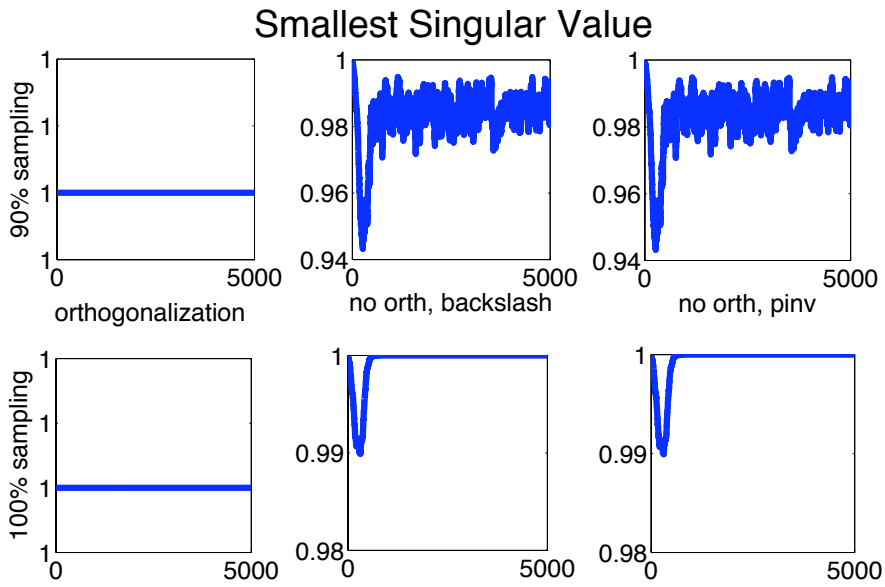


Figure 15: Since the LMS matrix variable W does not have to be orthonormal, we looked at the value of the smallest singular value to ensure numerical stability. We found evidence that W becomes orthogonal in the limit [34] when the vector is fully sampled, but not when there are missing observations.

and NNLS achieved excellent reconstruction, but GROUSE was twice as fast.

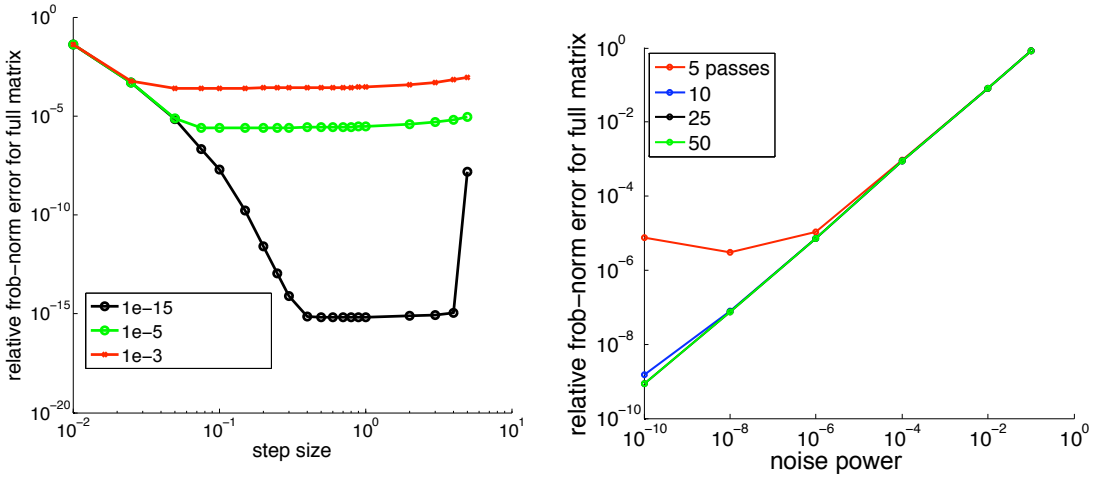


Figure 16: (a) Performance sensitivity to noise parameter ω and stepsize for matrix completion. Here we use a diminishing stepsize $\eta = C/t$. (b) Here we plot the time to get to a desired Frobenius norm error on the hidden matrix. We see that GROUSE converges to the noise floor after at most 10 passes over the columns of the matrix.

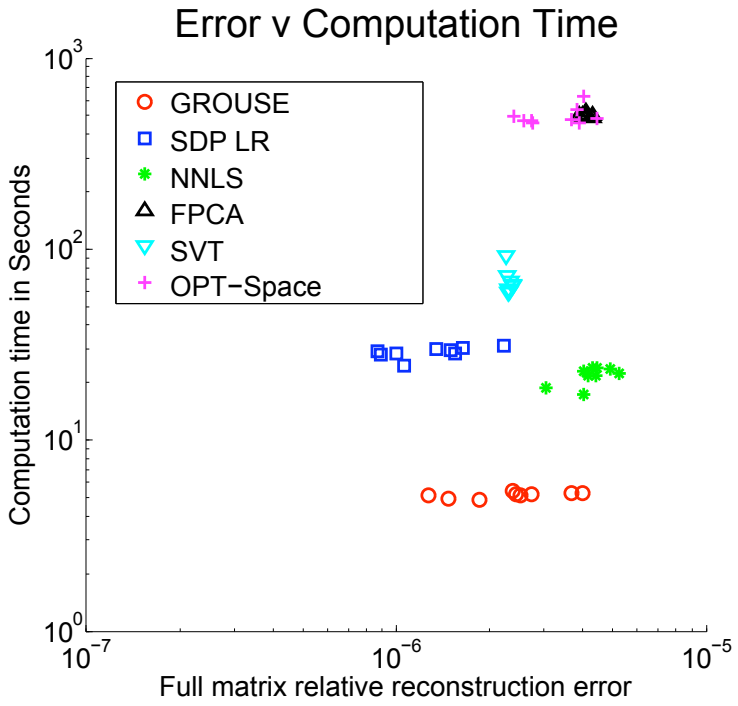


Figure 17: This figure compares five matrix completion algorithms against GROUSE.

Problem parameters				GROUSE			NNLS	
n_r	n_c	r	dens	rel.err.	passes	time	rel. err.	time (s)
5000	20000	5	0.006	1.10e-4	2	14.8	4.12e-04	66.9
5000	20000	10	0.012	1.5e-3	2	21.1	1.79e-4	81.2
6000	18000	5	0.006	1.44e-5	3	21.3	3.17e-4	66.8
6000	18000	10	0.011	8.24e-5	3	31.1	2.58e-4	68.8
7500	15000	5	0.005	5.71e-4	4	36.0	3.09e-4	62.6
7500	15000	10	0.013	1.41e-5	4	38.3	1.67e-4	68.0

Figure 18: This table gives a comparison of GROUSE and NNLS on large random matrix completion problems.

Appendix A

Details of the GROUSE derivation

A.1 Derivative of F

In Section 4.2 we used the derivative of $F(S;t) = \min_a \|\Delta_{\Omega_t}(v_t - Ua)\|^2$ to develop the GROUSE algorithm. Here we show how to reach the derivative, first using matrix derivative identities [70] without justification. We then give a complete treatment using the Fréchet derivative.

In Section 4.2, $\frac{dF}{dU}$ is given as $2v_\perp w^T$ (Equation 4.3). We derive this as follows:

$$\begin{aligned}
\frac{dF}{dU} &= \frac{d}{dU}\left(v_\Omega^T v_\Omega - v_\Omega^T U_\Omega \left(U_\Omega^T U_\Omega\right)^{-1} U_\Omega^T v_\Omega\right) \\
&= \frac{d}{dU}\left(-\text{Trace}\left(v_\Omega^T U_\Omega \left(U_\Omega^T U_\Omega\right)^{-1} U_\Omega^T v_\Omega\right)\right) \qquad \text{(A.1)} \\
&= -\frac{d}{dU}\text{Trace}\left(\left(U^T\Sigma_\Omega U\right)^{-1}\left(U^T\Sigma_\Omega v v^T \Sigma_\Omega U\right)\right) \qquad \text{(A.2)} \\
&= 2\Sigma_\Omega U\left(U^T\Sigma_\Omega U\right)^{-1}\left(U^T\Sigma_\Omega v v^T\Sigma_\Omega U\right)\left(U^T\Sigma_\Omega U\right)^{-1} \\
&\quad -2\left(\Sigma_\Omega v v^T\Sigma_\Omega U\right)\left(U^T\Sigma_\Omega U\right)^{-1} \qquad \text{(A.3)}
\end{aligned}$$

where Step A.1 holds because the Trace of a scalar is itself, Step A.2 uses the cyclic property of the trace, and Step A.3 is calculated using rules of matrix derivatives which can be found in [70]. Note also that we have transposed the result to be consistent with derivative notation in [37]. Defining the projection weights to be

$$w = (U_\Omega^T U_\Omega)^{-1} U_\Omega^T v_\Omega = \left(U^T\Sigma_\Omega U\right)^{-1} U^T\Sigma_\Omega v$$

we have

$$\begin{aligned}\frac{dF}{dU} &= 2\Sigma_\Omega U w w^T - 2\Sigma_\Omega v w^T \\ &= \begin{cases} 2(U_\Omega w - v_\Omega)w^T & \text{on } \Omega \\ 0 & otherwise \end{cases} \\ &= 2v_\perp w^T\end{aligned}$$

where we define $v_\perp = (U_\Omega w - v_\Omega)$ on Ω and 0 otherwise.

In order to develop Step A.3 above more carefully, here we use the Fréchet derivative. For a function $F(U)$, the derivative $\frac{dF}{dU}$ is the function such that

$$\left\langle \frac{dF}{dU}, X \right\rangle = \frac{d}{d\epsilon} F(U + \epsilon X)\Big|_{\epsilon=0} .$$

Thus we look at $\frac{d}{d\epsilon}F(U+\epsilon X)\big|_{\epsilon=0}$, and manipulate it such that it is the inner product of some function g with X, $\langle g, X\rangle$. We can read off that function g as the derivative $\frac{dF}{dU}$.

$$F(U+\epsilon X) = -\text{Tr}\left(v^T\Sigma_\Omega(U+\epsilon X)\left((U+\epsilon X)^T\Sigma_\Omega(U+\epsilon X)\right)^{-1}(U+\epsilon X)^T\Sigma_\Omega v\right)$$

$$= -\text{Tr}\left(v^T\Sigma_\Omega(U+\epsilon X)\left(A\left(I-B\right)\right)^{-1}(U+\epsilon X)^T\Sigma_\Omega v\right)$$

where we define $A = U^T\Sigma_\Omega U$ and

$$B = -\epsilon A^{-1}X^T\Sigma_\Omega U - \epsilon A^{-1}U^T\Sigma_\Omega X - \epsilon^2 A^{-1}X^T\Sigma_\Omega X \ .$$

We continue

$$
\begin{aligned}
F(U+\epsilon X) &= -\text{Tr}(v^T\Sigma_\Omega(U+\epsilon X)(I-B)^{-1}A^{-1}(U+\epsilon X)^T\Sigma_\Omega v) \\
&= -\text{Tr}(v^T\Sigma_\Omega(U+\epsilon X)\sum_{k=0}^{\infty}B^kA^{-1}(U+\epsilon X)^T\Sigma_\Omega v) \\
&= -\text{Tr}(v^T\Sigma_\Omega(U+\epsilon X)\left(B^0+B^1+\sum_{k=2}^{\infty}B^k\right)A^{-1}(U+\epsilon X)^T\Sigma_\Omega v) \\
&= -\text{Tr}(v^T\Sigma_\Omega UA^{-1}U^T\Sigma_\Omega v+\epsilon v^T\Sigma_\Omega UA^{-1}X^T\Sigma_\Omega v+\epsilon v^T\Sigma_\Omega XA^{-1}U^T\Sigma_\Omega v \\
&\quad +\epsilon^2 v^T\Sigma_\Omega XA^{-1}X^T\Sigma_\Omega v+v^T\Sigma_\Omega UBA^{-1}U^T\Sigma_\Omega v+\epsilon v^T\Sigma_\Omega UBA^{-1}X^T\Sigma_\Omega v \\
&\quad +\epsilon v^T\Sigma_\Omega XBA^{-1}U^T\Sigma_\Omega v+\epsilon^2 v^T\Sigma_\Omega XBA^{-1}X^T\Sigma_\Omega v \\
&\quad +v^T\Sigma_\Omega(U+\epsilon X)\sum_{k=2}^{\infty}B^kA^{-1}(U+\epsilon X)^T\Sigma_\Omega v)
\end{aligned}
$$

We rearrange so that all terms with an ϵ^2 or higher are together; we pull out the ϵ^2 and replace the other variables simply by C. We also replace B with its definition:

$$
\begin{aligned}
= -\text{Tr}\big(v^T\Sigma_\Omega UA^{-1}U^T\Sigma_\Omega v \quad & +\epsilon v^T\Sigma_\Omega UA^{-1}X^T\Sigma_\Omega v+\epsilon v^T\Sigma_\Omega XA^{-1}U^T\Sigma_\Omega v \\
& +v^T\Sigma_\Omega UBA^{-1}U^T\Sigma_\Omega v+\epsilon^2 C\big)
\end{aligned}
$$

$$
\begin{aligned}
= \quad -\text{Tr}\big(&v^T\Sigma_\Omega UA^{-1}U^T\Sigma_\Omega v+\epsilon v^T\Sigma_\Omega UA^{-1}X^T\Sigma_\Omega v+\epsilon v^T\Sigma_\Omega XA^{-1}U^T\Sigma_\Omega v \\
&-\epsilon v^T\Sigma_\Omega UA^{-1}X^T\Sigma_\Omega UA^{-1}U^T\Sigma_\Omega v-\epsilon v^T\Sigma_\Omega UA^{-1}U^T\Sigma_\Omega XA^{-1}U^T\Sigma_\Omega v+\epsilon^2 C\big)
\end{aligned}
$$

At this point, we can take the derivative with respect to ϵ:

$$\begin{aligned}
\frac{d}{d\epsilon}F(U+\epsilon X) &= -\text{Tr}(v^T\Sigma_\Omega U A^{-1}X^T\Sigma_\Omega v + v^T\Sigma_\Omega X A^{-1}U^T\Sigma_\Omega v \\
&\quad -v^T\Sigma_\Omega U A^{-1}X^T\Sigma_\Omega U A^{-1}U^T\Sigma_\Omega v \\
&\quad -v^T\Sigma_\Omega U A^{-1}U^T\Sigma_\Omega X A^{-1}U^T\Sigma_\Omega v + 2\epsilon C) \qquad \text{(A.4)} \\
&= -\text{Tr}(A^{-1}U^T\Sigma_\Omega v v^T\Sigma_\Omega X) - \text{Tr}(A^{-1}U^T\Sigma_\Omega v v^T\Sigma_\Omega X) \\
&\quad +\text{Tr}(A^{-1}U^T\Sigma_\Omega v v^T\Sigma_\Omega U A^{-1}U^T\Sigma_\Omega X) \qquad \text{(A.5)} \\
&\quad +\text{Tr}(A^{-1}U^T\Sigma_\Omega v v^T\Sigma_\Omega U A^{-1}U^T\Sigma_\Omega X) - \text{Tr}(2\epsilon C) \qquad \text{(A.6)} \\
&= \text{Tr}(\left[-2A^{-1}U^T\Sigma_\Omega v v^T\Sigma_\Omega + 2A^{-1}U^T\Sigma_\Omega v v^T\Sigma_\Omega U A^{-1}U^T\Sigma_\Omega\right] X) \qquad \text{(A.7)} \\
&\quad -\text{Tr}(2\epsilon C)
\end{aligned}$$

Step A.4 and A.7 use linearity of the Trace. Step A.6 also uses that $\text{Tr}(A) = \text{Tr}(A^T)$, the cyclic property of the Trace, and the fact that A^{-1} is symmetric. Setting $\epsilon = 0$, the final term drops off. Taking the transpose to be consistent with the derivative notation of [37], we can read our derivative from within the brackets:

$$\begin{aligned}
\frac{d}{dU}F(U) &= \left[-2(U_\Omega^TU_\Omega)^{-1}U_\Omega^Tv_\Omega v_\Omega^T + 2(U_\Omega^TU_\Omega)^{-1}U_\Omega^Tv_\Omega v_\Omega^TU_\Omega(U_\Omega^TU_\Omega)^{-1}U_\Omega^T\right]^T \\
&= \left[2\left((U_\Omega^TU_\Omega)^{-1}U_\Omega^Tv_\Omega\right)\left(v_\Omega^TU_\Omega(U_\Omega^TU_\Omega)^{-1}U_\Omega^T - v_\Omega^T\right)\right]^T \\
&= 2v_\perp w^T
\end{aligned}$$

where again we define the projection weights to be $w = (U_\Omega^TU_\Omega)^{-1}U_\Omega^Tv_\Omega = \left(U^T\Sigma_\Omega U\right)^{-1}U^T\Sigma_\Omega v$ and $v_\perp = (U_\Omega w - v_\Omega)$ on Ω and 0 otherwise.

A.2 Update Step Derivation

We have from Section 4.2 that

$$\nabla F = -2v_\perp w^T\,.$$

We wish to find an update such that we take a geodesic step on the Grassmannian in the direction of the negative gradient. Let $\sigma = \|v_\perp\|\|w\|$ and let the SVD of the negative gradient $v_\perp w^T$ be written as in Section 4.2:

$$v_\perp w^T = YSZ^T = \begin{bmatrix} \frac{v_\perp}{\|v_\perp\|} & x_2 & \dots & x_d \end{bmatrix} \times \mathrm{diag}(\sigma, 0, \dots, 0) \times \begin{bmatrix} \frac{w}{\|w\|} & y_2 & \dots & y_d \end{bmatrix}^T .$$

Using (2.65) from [37], we find that for $\eta > 0$, a step of length η in the direction $v_\perp w^T$ is given by

$$\begin{aligned} U(\eta) &= UZ\cos(S\eta)Z^T + Y\sin(S\eta)Z^T \\ &= UZ\cos(\mathrm{diag}(\sigma\eta, 0, \dots, 0))Z^T + Y\sin(\mathrm{diag}(\sigma\eta, 0, \dots, 0))Z^T \\ &= UZ\,\mathrm{diag}(\cos(\sigma\eta), 1, \dots, 1)Z^T + Y\,\mathrm{diag}(\sin(\sigma\eta), 0, \dots, 0)Z^T \\ &= UZ\left[\mathrm{diag}(1, 1, \dots, 1) + \mathrm{diag}(\cos(\sigma\eta) - 1, 0, \dots, 0)\right]Z^T + Y\,\mathrm{diag}(\sin(\sigma\eta), 0, \dots, 0)Z^T \\ &= UZIZ^T + UZ\,\mathrm{diag}(\cos(\sigma\eta) - 1, 0, \dots, 0)Z^T + Y\,\mathrm{diag}(\sin(\sigma\eta), 0, \dots, 0)Z^T \\ &= U + (\cos(\sigma\eta) - 1)U\frac{w}{\|w\|}\frac{w^T}{\|w\|} + \sin(\sigma\eta)\frac{v_\perp}{\|v_\perp\|}\frac{w^T}{\|w\|} \end{aligned}$$

which then gives us the GROUSE update equation

$$\begin{aligned} U(\eta) &= U + \frac{(\cos(\sigma\eta) - 1)}{\|w\|^2}Uww^T + \sin(\sigma\eta)\frac{v_\perp}{\|v_\perp\|}\frac{w^T}{\|w\|} \\ &= U + \left(\sin(\sigma\eta)\frac{v_\perp}{\|v_\perp\|} + (\cos(\sigma\eta) - 1)\frac{v_{||}}{\|v_{||}\|}\right)\frac{w^T}{\|w\|} \end{aligned}$$

where $v_{||} := Uw$, the predicted value of the projection of the vector v_t onto the span of U.

Chapter 5

Estimating Unions of Subspaces with Missing Data

The work in this chapter is joint with Brian Eriksson and Robert Nowak [3] and with Arthur Szlam, Benjamin Recht, and Rob Nowak [7].

Modeling high-dimensional data with a union of subspaces is a useful generalization of subspace models [67], and has applications in machine learning, imaging, network topology identification [3], computer vision [32], and system identification [103].

A union of k r-dimensional subspaces itself spans a subspace which has dimension at most kr. However, the union of subspaces model is often more flexible than the kr-dimensional subspace. Consider the illustration in Figure 19; though the two lines (two one-dimensional subspaces) are a simpler model than the two-dimensional plane, the set they define is non-convex.

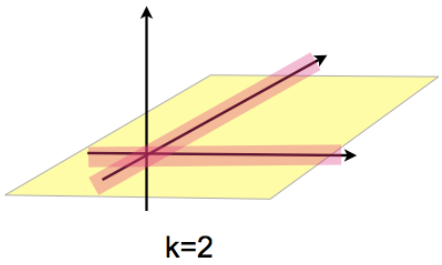


Figure 19: Illustration of two 1-d subspaces and one 2-d subspace; though the two pink lines (two one-dimensional subspaces) are a simpler model than the yellow two-dimensional plane, the set they define is non-convex.

In this chapter we formulate the problem of estimating a union of subspaces from data vectors

which have missing values. We present a methodology in Section 5.2 for solving this problem, and in Section 5.3 we prove that this methodology will succeed in estimating the union of subspaces with high probability. We then present another algorithm in Section 5.4 which is more computationally efficient, and we compare the two in numerical experiments in Section 5.5.

5.1 Problem Formulation

Consider a real-valued $n \times N$ dimensional matrix $\boldsymbol{X}$. Assume that the columns of $\boldsymbol{X}$ lie in the union of at most k subspaces of $\mathbb{R}^n$, each having dimension at most $r < n$ and assume that $N > kn$. We are especially interested in "high-rank" situations in which the total rank (the rank of the union of the subspaces) may be n. Our goal is to complete $\boldsymbol{X}$ based on an observation of a small random subset of its entries. We propose a novel method for this matrix completion problem. In the applications we have in mind N may be arbitrarily large, and so we will focus on quantifying the probability that a given column is perfectly completed, rather than the probability that whole matrix is perfectly completed (*i.e.,* every column is perfectly completed). Of course it is possible to translate between these two quantifications using a union bound, but that bound becomes meaningless if N is extremely large.

Suppose the entries of $\boldsymbol{X}$ are observed uniformly at random with probability p_0. Let Ω denote the set of indices of observed entries and let $\boldsymbol{X}_\Omega$ denote the observations of $\boldsymbol{X}$. Our main result shows that under a mild set of assumptions each column of $\boldsymbol{X}$ can be perfectly recovered from $\boldsymbol{X}_\Omega$ with probability at least $1 - 20kn^{2-2\beta^{1/2}} \log n$, $\beta > 1$, using a computationally efficient procedure if

$$p_0 \geq C\,\beta\,\frac{r}{n}\log^2(n) \tag{5.1}$$

where $C > 1$ is a constant depending on the usual incoherence conditions as well as the geometrical arrangement of subspaces and the distribution of the columns in the subspaces.

5.1.1 Connections to Low-Rank Completion

Low-rank matrix completion theory [85] shows that an $n \times N$ matrix of rank r can be recovered from incomplete observations, as long as the number of entries observed (with locations sampled uniformly at random) exceeds $rN \log^2 N$ (within a constant factor and assuming $n \leq N$). It is also known that, in the same setting, completion is impossible if the number of observed entries is less than a constant times $rN \log N$ [23]. These results imply that if the rank of $\boldsymbol{X}$ is close to n, then all of the entries are needed in order to determine the matrix.

Here we consider a matrix whose columns lie in the union of at most k subspaces of $\mathbb{R}^n$. Restricting the rank of each subspace to at most r, then the rank of the full matrix our situation could be as large as kr, yielding the requirement $krN \log^2 N$ using current matrix completion theory. In contrast, the bound in (5.1) implies that the completion of each column is possible from a constant times $rN \log^2 n$ entries sampled uniformly at random. Exact completion of every column can be guaranteed by replacing $\log^2 n$ with $\log^2 N$ is this bound, but since we allow N to be very large we prefer to state our result in terms of per-column completion. Our method, therefore, improves significantly upon conventional low-rank matrix completion, especially when k is large. This does not contradict the lower bound in [23], because the matrices we consider are not arbitrary high-rank matrices, rather the columns must belong to a union of rank $\leq r$ subspaces.

5.1.2 Connections to Subspace Clustering

Let $x_1, \ldots, x_N \in \mathbb{R}^n$ and assume each x_i lies in one of at most k subspaces of $\mathbb{R}^n$. Subspace clustering is the problem of learning the subspaces from $\{x_i\}_{i=1}^N$ and assigning each vector to its proper subspace; cf. [101] for a overview. This is a challenging problem, both in terms of computation and inference, but provably probably correct subspace clustering algorithms now exist [56, 102, 63]. Here we consider the problem of *high rank matrix completion*, which is essentially equivalent to subspace clustering with missing data. This problem has been looked at in previous works [47, 104], but to the best of

our knowledge our method and theoretical bounds are novel. Note that our sampling probability bound (5.1) requires that only slightly more than r out of n entries are observed in each column, so the matrix may be highly incomplete.

5.1.3 Related Work

The proof of the main result draws on ideas from matrix completion theory, subspace learning and detection with missing data, and subspace clustering. One key ingredient in our approach is the celebrated results on low-rank Matrix Completion [85, 23, 22]. Unfortunately, in many real-world problems where missing data is present, particularly when the data is generated from a union of subspaces, these matrices can have very large rank values (*e.g.,* networking data in [40]). Thus, these prior results will require effectively all the elements be observed to accurately reconstruct the matrix.

Our work builds upon the results of [8], which quantifies the deviation of an incomplete vector norm with respect to the incoherence of the sampling pattern. While this work also examines subspace detection using incomplete data, it assumes complete knowledge of the subspaces.

While research that examines subspace learning has been presented in [112, 29], the work in this chapter differs by the concentration on learning from incomplete observations (*i.e.,* when there are missing elements in the matrix), and by the methodological focus (*i.e.,* nearest neighbor clustering versus a multiscale Singular Value Decomposition approach).

5.2 Algorithm: Estimating a Union of Subspaces from Missing Data

We present an algorithm for estimating a union of subspaces from incomplete vectors. It involves several relatively intuitive steps, outlined here. We go into detail for each of these steps in Section 5.3. This work is joint with Brian Eriksson and Rob Nowak [3].

Local Neighborhoods. A subset of columns of $\boldsymbol{X}_\Omega$ are selected uniformly at random. These are called

seeds. A set of nearest neighbors is identified for each seed from the remainder of $\boldsymbol{X}_\Omega$. In Section 5.3.1, we show that nearest neighbors can be reliably identified, even though a large portion of the data are missing, under the usual incoherence assumptions.

Local Subspaces. The subspace spanned by each seed and its neighborhood is identified using matrix completion. If matrix completion fails (*i.e.,* if the resulting matrix does not agree with the observed entries and/or the rank of the result is greater than r), then the seed and its neighborhood are discarded. In Section 5.3.2 we show that when the number of seeds and the neighborhood sizes are large enough, then with high probability all k subspaces are identified. We may also identify additional subspaces which are unions of the true subspaces, which leads us to the next step. An example of these neighborhoods is shown in Figure 20.

Subspace Refinement. The set of subspaces obtained from the matrix completions is pruned to remove all but k subspaces. The pruning is accomplished by simply discarding all subspaces that are spanned by the union of two or more other subspaces. This can be done efficiently, as is shown in Section 5.3.3.

Full Matrix Completion. Each column in $\boldsymbol{X}_\Omega$ is assigned to its proper subspace and completed by projection onto that subspace, as described in Section 5.3.4. Even when many observations are missing, it is possible to find the correct subspace and the projection using results from subspace detection with missing data [8]. The result of this step is a completed matrix $\widehat{\boldsymbol{X}}$ such that each column is correctly completed with high probability.

The mathematical analysis will be presented in the next few sections, organized according to these steps. After proving the main result, experimental results are presented in the final section.

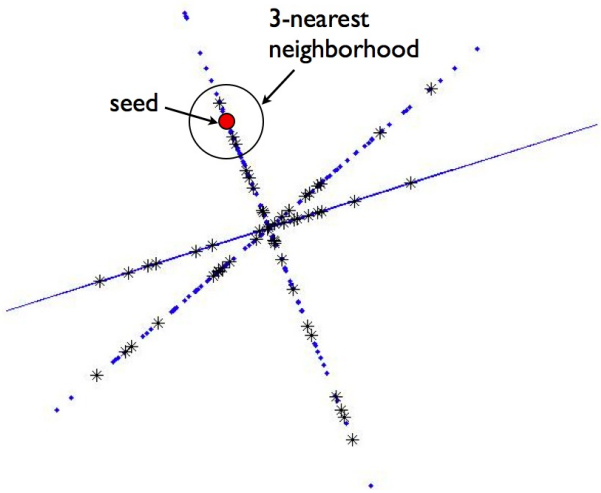


Figure 20: Example of nearest-neighborhood selecting points on from a single subspace. For illustration, samples from three one-dimensional subspaces are depicted as small dots. The large dot is the seed. The subset of samples with significant observed support in common with that of the seed are depicted by $*$'s. If the density of points is high enough, then the nearest neighbors we identify will belong to the same subspace as the seed. In this case we depict the ball containing the 3 nearest neighbors of the seed with significant support overlap.

5.3 Theoretical Analysis

We start by describing the key assumptions in our analysis, and giving our main result. The notion of incoherence plays a key role in matrix completion and subspace recovery from incomplete observations.

Definition 3. *The* coherence *of an r-dimensional subspace $\mathcal{S} \subseteq \mathbb{R}^n$ is*

$$\mu(\mathcal{S}) := \frac{n}{r} \max_j \|P_{\mathcal{S}} e_j\|_2^2$$

where $P_{\mathcal{S}}$ is the projection operator onto $\mathcal{S}$ and $\{e_j\}$ are the canonical unit vectors for $\mathbb{R}^n$.

Note that $1 \leq \mu(\mathcal{S}) \leq n/r$. The coherence of single vector $x \in \mathbb{R}^n$ is $\mu(x) = \frac{n\|x\|_\infty^2}{\|x\|_2^2}$, which is precisely the coherence of the one-dimensional subspace spanned by x. With this definition, we can state the main assumptions we make about the matrix $\boldsymbol{X}$.

A1. The columns of $\boldsymbol{X}$ lie in the union of at most k subspaces, with $k = o(n^d)$ for some $d > 0$. The

subspaces are denoted by $\mathcal{S}_1, \ldots, \mathcal{S}_k$ and each has rank at most $r < n$. The ℓ_2-norm of each column is ≤ 1.

A2. The coherence of each subspace is bounded above by μ_0. The coherence of each column is bounded above by μ_1 and for any pair of columns, x_1 and x_2, the coherence of $x_1 - x_2$ is also bounded above by μ_1.

A3. Let $0 < \epsilon_0 < 1$ and $\mathcal{S}_{i,\epsilon_0}$ denote the subset of points in $\mathcal{S}_i$ at least ϵ_0 distance away from any other subspace. There exists a constant $0 < \nu_0 \leq 1$, depending on ϵ_0, such that

(i) The probability that a column selected uniformly at random belongs to $\mathcal{S}_{i,\epsilon_0}$ is at least ν_0/k.

(ii) If $x \in \mathcal{S}_{i,\epsilon_0}$, then the probability that a column selected uniformly at random belongs to the ball of radius ϵ_0 centered at x is at least $\nu_0 \epsilon_0^r/k$.

Additionally, if $x \in \mathcal{S}_i$, then the probability that $x \in \mathcal{S}_j$, $j \neq i$ is zero.

The value of ν_0 depends on the geometrical arrangement of the subspaces and the distribution of the columns within the subspaces. If the subspaces are not too close to each other, and the distributions within the subspaces are fairly uniform, then typically ν_0 will be not too close to 0. We define three key quantities, the confidence parameter δ_0, the required number of "seed" columns s_0, and a quantity ℓ_0 related to the neighborhood formation process (see Algorithm 3 in Section 5.3.1):

$$\begin{aligned} \delta_0 &:= n^{2-2\beta^{1/2}} \log n \text{ , for some } \beta > 1 \text{ ,} \\ s_0 &:= \left\lceil \frac{k(\log k + \log 1/\delta_0)}{(1-e^{-4})\nu_0} \right\rceil \text{ ,} \\ \ell_0 &:= \left\lceil \max\left\{ \frac{2k}{\nu_0(\frac{\epsilon_0}{\sqrt{3}})^r} \text{ , } \frac{8k\log(s_0/\delta_0)}{n\nu_0(\frac{\epsilon_0}{\sqrt{3}})^r} \right\} \right\rceil \text{ .} \end{aligned} \tag{5.2}$$

We can now state our main result.

Theorem 5.1. *Let $\boldsymbol{X}$ be an $n \times N$ matrix satisfying* **A1**-**A3**. *Suppose that each entry of $\boldsymbol{X}$ is observed independently with probability p_0. If*

$$p_0 \geq \frac{128\,\beta \max\{\mu_1^2, \mu_0\}}{\nu_0} \frac{r \log^2(n)}{n}$$

and

$$N \quad \geq \quad \ell_0 n (2\delta_0^{-1} s_0 \ell_0 n)^{\mu_0^2 \log p_0^{-1}}$$

then each column of $\boldsymbol{X}$ *can be perfectly recovered with probability at least* $1 - (6 + 15 s_0)\, \delta_0$*, using the methodology sketched above (and detailed later in the following sections).*

The requirements on sampling are essentially the same as those for standard low-rank matrix completion, apart from requirement that the total number of columns N is sufficiently large. This is needed to ensure that each of the subspaces is sufficiently represented in the matrix. The requirement on N is polynomial in n for fixed p_0, which is easy to see based on the definitions of δ_0, s_0, and ℓ_0 (see further discussion at the end of Section 5.3.1).

Perfect recovery of each column is guaranteed with probability that decreases linearly in s_0, which itself is linear in k (ignoring log factors). This is expected since this problem is more difficult than k individual low-rank matrix completions. We state our results in terms of a per-column (rather than full matrix) recovery guarantee. A full matrix recovery guarantee can be given by replacing $\log^2 n$ with $\log^2 N$. This is evident from the final completion step discussed in Lemma 5.8, below. However, since N may be quite large (perhaps arbitrarily large) in the applications we envision, we chose to state our results in terms of a per-column guarantee.

The details of the methodology and lemmas leading to the theorem above are developed in the subsequent sections following the four steps of the methodology outlined above. In certain cases it will be more convenient to consider sampling the locations of observed entries uniformly at random *with replacement* rather than without replacement, as assumed above. The following lemma will be useful for translating bounds derived assuming sampling with replacement to our situation (the same sort of relation is noted in Proposition 3.1 in [85]).

Lemma 5.1. *Draw* m *samples independently and uniformly from* $\{1, \ldots, n\}$ *and let* Ω' *denote the resulting subset of unique values. Let* Ω_m *be a subset of size* m *selected uniformly at random from*

$\{1, \ldots, n\}$. *Let E denote an event depending on a random subset of $\{1, \ldots, n\}$. If $\mathbb{P}(E(\Omega_m))$ is a non-increasing function of m, then $\mathbb{P}(E(\Omega')) \geq \mathbb{P}(E(\Omega_m))$.*

Proof. For $k = 1, \ldots, m$, let Ω_k denote a subset of size k sampled uniformly at random from $\{1, \ldots, n\}$, and let $m' = |\Omega'|$.

$$\begin{aligned}
\mathbb{P}(E(\Omega')) &= \sum_{k=0}^{m} \mathbb{P}\left(E(\Omega') \,|\, m' = k\right) \mathbb{P}(m' = k) \\
&= \sum_{k=0}^{m} \mathbb{P}(E(\Omega_k)) \mathbb{P}(m' = k) \\
&\geq \mathbb{P}(E(\Omega_m)) \sum_{k=0}^{m} \mathbb{P}(m' = k) \, .
\end{aligned}$$

□

5.3.1 Local Neighborhoods

In this first step, s columns of $\mathbf{X}_\Omega$ are selected uniformly at random and a set of "nearby" columns are identified for each, constituting a local neighborhood of size n. All bounds that hold are designed with probability at least $1 - \delta_0$, where δ_0 is defined in (5.2) above. The s columns are called "seeds." The required size of s is determined as follows.

Lemma 5.2. *Assume* **A3** *holds. If the number of chosen seeds,*

$$s \geq \frac{k(\log k + \log 1/\delta_0)}{(1 - e^{-4})\nu_0} \, ,$$

then with probability greater than $1 - \delta_0$ for each $i = 1, \ldots, k$, at least one seed is in $\mathcal{S}_{i,\epsilon_0}$ and each seed column has at least

$$\eta_0 := \frac{64\,\beta \max\{\mu_1^2, \mu_0\}}{\nu_0} \, r \, \log^2(n) \tag{5.3}$$

observed entries.

Proof. First note that from Theorem 5.1, the expected number of observed entries per column is at least

$$\eta = \frac{128\,\beta \max\{\mu_1^2, \mu_0\}}{\nu_0}\; r\; \log^2(n)$$

Therefore, the number of observed entries $\widehat{\eta}$ in a column selected uniformly at random is probably not significantly less. More precisely, by Chernoff's bound we have

$$\mathbb{P}(\widehat{\eta} \leq \eta/2) \;\leq\; \exp(-\eta/8) \;<\; e^{-4}\,.$$

Combining this with **A3**, we have the probability that a randomly selected column belongs to $\mathcal{S}_{i,\epsilon_0}$ and has $\eta/2$ or more observed entries is at least ν_0'/k, where $\nu_0' := (1 - e^{-4})\nu_0$. Then, the probability that the set of s columns does not contain a column from $\mathcal{S}_{i,\epsilon_0}$ with at least $\eta/2$ observed entries is less than $(1 - \nu_0'/k)^s$. The probability that the set does not contain at least one column from $\mathcal{S}_{i,\epsilon_0}$ with $\eta/2$ or more observed entries, for $i = 1, \ldots, k$ is less than $\delta_0 = k(1 - \nu_0'/k)^s$. Solving for s in terms of δ_0 yields

$$s \;=\; \frac{\log k + \log 1/\delta_0}{\log\left(\frac{k/\nu_0'}{k/\nu_0' - 1}\right)}$$

The result follows by noting that $\log(x/(x-1)) \geq 1/x$, for $x > 1$. □

Next, for each seed we must find a set of n columns from the same subspace as the seed. This will be accomplished by identifying columns that are ϵ_0-close to the seed, so that if the seed belongs to $\mathcal{S}_{i,\epsilon_0}$, the columns must belong to the same subspace. Clearly the total number of columns N must be sufficiently large so that n or more such columns can be found. We will return to the requirement on N a bit later, after first dealing with the following challenge.

Since the columns are only partially observed, it may not be possible to determine how close each is to the seed. We address this by showing that if a column and the seed are both observed on enough common indices, then the incoherence assumption **A2** allows us reliably estimate the distance.

Lemma 5.3. *Assume* **A2** *and let $y = x_1 - x_2$, where x_1 and x_2 are two columns of $\boldsymbol{X}$. Assume there is a common set of indices of size $q \leq n$ where both x_1 and x_2 are observed. Let ω denote this common*

set of indices and let y_ω denote the corresponding subset of y. Then for any $\delta_0 > 0$, if the number of commonly observed elements

$$q \;\geq\; 8\mu_1^2 \log(2/\delta_0)\;,$$

then with probability at least $1 - \delta_0$

$$\frac{1}{2}\|y\|_2^2 \;\leq\; \frac{n}{q}\|y_\omega\|_2^2 \;\leq\; \frac{3}{2}\|y\|_2^2\;.$$

Proof. Note that $\|y_\omega\|_2^2$ is the sum of q random variables drawn uniformly at random without replacement from the set $\{y_1^2, y_2^2, \ldots, y_n^2\}$, and $\mathbb{E}\|y_\omega\|_2^2 = \frac{q}{n}\|y\|_2^2$. We will prove the bound under the assumption that, instead, the q variables are sampled with replacement, so that they are independent. By Lemma 5.1, this will provide the desired result. Note that if one variable in the sum $\|y_\omega\|_2^2$ is replaced with another value, then the sum changes in value by at most $2\|y\|_\infty^2$. Therefore, McDiramid's Inequality shows that for $t > 0$

$$\mathbb{P}\left(\left|\|y_\omega\|_2^2 - \frac{q}{n}\|y\|_2^2\right| \geq t\right) \;\leq\; 2\,\exp\left(\frac{-t^2}{2q\|y\|_\infty^4}\right)\;,$$

or equivalently

$$\mathbb{P}\left(\left|\frac{n}{q}\|y_\omega\|_2^2 - \|y\|_2^2\right| \geq t\right) \;\leq\; 2\,\exp\left(\frac{-qt^2}{2n^2\|y\|_\infty^4}\right)\;.$$

Assumption **A2** implies that $n^2\|y\|_\infty^4 \leq \mu_1^2\|y\|_2^4$, and so we have

$$\mathbb{P}\left(\left|\frac{n}{q}\|y_\omega\|_2^2 - \|y\|_2^2\right| \geq t\right) \;\leq\; 2\,\exp\left(\frac{-qt^2}{2\mu_1^2\|y\|_2^4}\right)\;.$$

Taking $t = \frac{1}{2}\|y\|_2^2$ yields the result. □

Suppose that $x_1 \in \mathcal{S}_{i,\epsilon_0}$ (for some i) and that $x_2 \notin \mathcal{S}_i$, and that both x_1, x_2 observe $q \geq 2\mu_0^2 \log(2/\delta_0)$ common indices. Let y_ω denote the difference between x_1 and x_2 on the common support set. If the *partial distance* $\frac{n}{q}\|y_\omega\|_2^2 \leq \epsilon_0^2/2$, then the result above implies that with probability at least $1 - \delta_0$

$$\|x_1 - x_2\|_2^2 \;\leq\; 2\,\frac{n}{q}\,\|y_\omega\|_2^2 \;\leq\; \epsilon_0^2.$$

On the other hand if $x_2 \in \mathcal{S}_i$ and $\|x_1 - x_2\|_2^2 \leq \epsilon_0^2/3$, then with probability at least $1 - \delta_0$

$$\frac{n}{q}\,\|y_\omega\|_2^2 \;\leq\; \frac{3}{2}\|x_1 - x_2\|_2^2 \;\leq\; \epsilon_0^2/2\,.$$

Using these results we will proceed as follows. For each seed we find all columns that have at least $t_0 > 2\mu_0^2 \log(2/\delta_0)$ observations at indices in common with the seed (the precise value of t_0 will be specified in a moment). Assuming that this set is sufficiently large, we will select ℓn these columns uniformly at random, for some integer $\ell \geq 1$. In particular, ℓ will be chosen so that with high probability at least n of the columns will be within $\epsilon_0/\sqrt{3}$ of the seed, ensuring that with probability at least δ_0 the corresponding partial distance of each will be within $\epsilon_0/\sqrt{2}$. That is enough to guarantee with the same probability that the columns are within ϵ_0 of the seed. Of course, a union bound will be needed so that the distance bounds above hold uniformly over the set of $s\ell n$ columns under consideration, which means that we will need each to have at least $t_0 := 2\mu_0^2 \log(2s\ell n/\delta_0)$ observations at indices in common with the corresponding seed. All this is predicated on N being large enough so that such columns exist in $\boldsymbol{X}_\Omega$. We will return to this issue later, after determining the requirement for ℓ. For now we will simply assume that $N \geq \ell n$.

Lemma 5.4. *Assume* **A3** *and for each seed x let T_{x,ϵ_0} denote the number of columns of $\boldsymbol{X}$ in the ball of radius $\epsilon_0/\sqrt{3}$ about x. If the number of columns selected for each seed, ℓn, such that,*

$$\ell \;\geq\; \max\left\{ \frac{2k}{\nu_0(\frac{\epsilon_0}{\sqrt{3}})^r}\,,\; \frac{8k\log(s/\delta_0)}{n\nu_0(\frac{\epsilon_0}{\sqrt{3}})^r} \right\}\,,$$

then $\mathbb{P}\left(T_{x,\epsilon_0} \leq n\right) \;\leq\; \delta_0$ for all s seeds.

Proof. The probability that a column chosen uniformly at random from $\boldsymbol{X}$ belongs to this ball is at least $\nu_0(\epsilon_0/\sqrt{3})^r/k$, by Assumption **A3**. Therefore the expected number of points is

$$\mathbb{E}[T_{x,\epsilon_0}] \geq \frac{\ell n \nu_0(\frac{\epsilon_0}{\sqrt{3}})^r}{k}\,.$$

By Chernoff's bound for any $0 < \gamma < 1$

$$\mathbb{P}\left(T_{x,\epsilon_0} \leq (1-\gamma)\frac{\ell n \nu_0(\frac{\epsilon_0}{\sqrt{3}})^r}{k}\right) \leq \exp\left(-\frac{\gamma^2}{2}\frac{\ell n \nu_0(\frac{\epsilon_0}{\sqrt{3}})^r}{k}\right)\,.$$

Algorithm 3 - Local Neighborhood Procedure

Input: n, k, μ_0, ϵ_0, ν_0, η_0, $\delta_0 > 0$.

$$\begin{aligned} s_0 &:= \left\lceil \frac{k(\log k + \log 1/\delta_0)}{(1-e^{-4})\nu_0} \right\rceil \\ \ell_0 &:= \left\lceil \max\left\{ \frac{2k}{\nu_0(\frac{\epsilon_0}{\sqrt{3}})^r}, \frac{8k\log(s_0/\delta_0)}{n\nu_0(\frac{\epsilon_0}{\sqrt{3}})^r} \right\} \right\rceil \\ t_0 &:= \lceil 2\mu_0^2 \log(2s_0\ell_0 n/\delta_0) \rceil \end{aligned}$$

Steps:

1. Select s_0 "seed" columns uniformly at random and discard all with less than η_0 observations
2. For each seed, find all columns with t_0 observations at locations observed in the seed
3. Randomly select $\ell_0 n$ columns from each such set
4. Form local neighborhood for each seed by randomly selecting n columns with partial distance less than $\epsilon_0/\sqrt{2}$ from the seed

---

Take $\gamma = 1/2$ which yields

$$\mathbb{P}\left(T_{x,\epsilon_0} \leq \frac{\ell n \nu_0 (\frac{\epsilon_0}{\sqrt{3}})^r}{2k} \right) \leq \exp\left(-\frac{\ell n \nu_0 (\frac{\epsilon_0}{\sqrt{3}})^r}{8k} \right).$$

We would like to choose ℓ so that $\frac{\ell n \nu_0 (\frac{\epsilon_0}{\sqrt{3}})^r}{2k} \geq n$ and so that $\exp\left(-\frac{\ell n \nu_0 (\frac{\epsilon_0}{\sqrt{3}})^r}{8k} \right) \leq \delta_0/s$ (so that the desired result fails for one or more of the s seeds is less than δ_0). The first condition leads to the requirement $\ell \geq \frac{2k}{\nu_0(\frac{\epsilon_0}{\sqrt{3}})^r}$. The second condition produces the requirement $\ell \geq \frac{8k\log(s/\delta_0)}{n\nu_0(\frac{\epsilon_0}{\sqrt{3}})^r}$. □

We can now formally state the procedure for finding local neighborhoods in Algorithm 3. Recall that the number of observed entries in each seed is at least η_0, per Lemma 5.2.

Lemma 5.5. *If N is sufficiently large and $\eta_0 > t_0$, then the Local Neighborhood Procedure in Algorithm 3 produces at least n columns within ϵ_0 of each seed, and at least one seed will belong to each of $\mathcal{S}_{i,\epsilon_0}$, for $i = 1, \ldots, k$, with probability at least $1 - 3\delta_0$.*

Proof. Lemma 5.2 states that if we select s_0 seeds, then with probability at least $1 - \delta_0$ there is a seed in each $\mathcal{S}_{i,\epsilon_0}$, $i = 1, \ldots, k$, with at least η_0 observed entries, where η_0 is defined in (5.3). Lemma 5.4

implies that if $\ell_0 n$ columns are selected uniformly at random for each seed, then with probability at least $1 - \delta_0$ for each seed at least n of the columns will be within a distance $\epsilon_0/\sqrt{3}$ of the seed. Each seed has at least η_0 observed entries and we need to find $\ell_0 n$ other columns with at least t_0 observations at indices where the seed was observed. Provided that $\eta_0 \geq t_0$, this is certainly possible if N is large enough. It follows from Lemma 5.3 that $\ell_0 n$ columns have at least t_0 observations at indices where the seed was also observed, then with probability at least $1 - \delta_0$ the partial distances will be within $\epsilon_0/\sqrt{2}$, which implies the true distances are within ϵ_0. The result follows by the union bound. □

Finally, we quantify just how large N needs to be. Lemma 5.4 also shows that we require at least

$$N \;\geq\; \ell n \;\geq\; \max\left\{ \frac{2kn}{\nu_0(\frac{\epsilon_0}{\sqrt{3}})^r} \,,\; \frac{8k\log(s/\delta_0)}{\nu_0(\frac{\epsilon_0}{\sqrt{3}})^r} \right\} .$$

However, we must also determine a lower bound on the probability that a column selected uniformly at random has at least t_0 observed indices in common with a seed. Let γ_0 denote this probability, and let p_0 denote the probability of observing each entry in $\boldsymbol{X}$. Note that our main result, Theorem 5.1, assumes that

$$p_0 \geq \; \frac{128\,\beta\max\{\mu_1^2, \mu_0\}}{\nu_0} \; \frac{r\,\log^2(n)}{n} \,.$$

Since each seed has at least η_0 entries observed, γ_0 is greater than or equal to the probability that a Binomial(η_0, p_0) random variable is at least t_0. Thus,

$$\gamma_0 \;\geq\; \sum_{j=t_0}^{\eta_0} \binom{\eta_0}{j} p_0^j (1-p_0)^{\eta_0 - j} \,.$$

This implies that the expected number of columns with t_0 or more observed indices in common with a seed is at least $\gamma_0 N$. If $\widetilde{n}$ is the actual number with this property, then by Chernoff's bound, $\mathbb{P}(\widetilde{n} \leq \gamma_0 N/2) \leq \exp(-\gamma_0 N/8)$. So $N \geq 2\ell_0\gamma_0^{-1} n$ will suffice to guarantee that enough columns can be found for each seed with probability at least $1 - s_0\exp(-\ell_0 n/4) \geq 1 - \delta_0$ since this will be far larger than $1 - \delta_0$, since δ_0 is polynomial in n.

To take this a step further, a simple lower bound on γ_0 is obtained as follows. Suppose we consider only a t_0-sized subset of the indices where the seed is observed. The probability that another column selected at random is observed at all t_0 indices in this subset is $p_0^{t_0}$. Clearly $\gamma_0 \geq p_0^{t_0} = \exp(t_0 \log p_0) \geq (2s_0\ell_0 n)^{2\mu_0^2 \log p_0}$. This yields the following sufficient condition on the size of N:

$$N \;\geq\; \ell_0 n (2s_0\ell_0 n/\delta_0)^{2\mu_0^2 \log p_0^{-1}} \,.$$

From the definitions of s_0 and ℓ_0, this implies that if $2\mu_0^2 \log p_0$ is a fixed constant, then a sufficient number of columns will exist if $N = O(\text{poly}(kn/\delta_0))$. For example, if $\mu_0^2 = 1$ and $p_0 = 1/2$, then $N = O((kn)/\delta_0)^{2.4})$ will suffice; i.e., N need only grow polynomially in n. On the other hand, in the extremely undersampled case p_0 scales like $\log^2(n)/n$ (as n grows and r and k stay constant) and N will need to grow almost exponentially in n, like $n^{\log n - 2\log\log n}$.

5.3.2 Local Subspace Completion

For each of our local neighbor sets, we will have an incompletely observed $n \times n$ matrix; if all the neighbors belong to a single subspace, the matrix will have rank $\leq r$. First, we recall the following result from low-rank matrix completion theory [85].

Lemma 5.6. *Consider an $n \times n$ matrix of rank $\leq r$ and row and column spaces with coherences bounded above by some constant μ_0. Then the matrix can be exactly completed if*

$$m' \geq 64 \max\left(\mu_1^2, \mu_0\right) \beta r n \log^2(2n) \tag{5.4}$$

entries are observed uniformly at random, for constants $\beta > 0$ and with probability $\geq 1 - 6\,(2n)^{2-2\beta}\log n - n^{2-2\beta^{1/2}}$.

We wish to apply these results to our local neighbor sets, but we have three issues we must address: First, the sampling of the matrices formed by local neighborhood sets is not uniform since the set is selected based on the observed indices of the seed. Second, given Lemma 5.2 we must complete not

one, but s_0 (see Algorithm 3) incomplete matrices simultaneously with high probability. Third, some of the local neighbor sets may have columns from more than one subspace. Let us consider each issue separately.

First consider the fact that our incomplete submatrices are not sampled uniformly. The non-uniformity can be corrected with a simple thinning procedure. Recall that the columns in the seed's local neighborhood are identified first by finding columns with sufficient overlap with each seed's observations. To refer to the seed's observations, we will say "the support of the seed."

Due to this selection of columns, the resulting neighborhood columns are highly sampled on the support of the seed. In fact, if we again use the notation q for the minimum overlap between two columns needed to calculate distance, then these columns have at least q observations on the support of the seed. Off the support, these columns are still sampled uniformly at random with the same probability as the entire matrix. Therefore we focus only on correcting the sampling pattern on the support of the seed.

Let t be the cardinality of the support of a particular seed. Because all entries of the entire matrix are sampled independently with probability p_0, then for a randomly selected column, the random variable which generates t is binomial. For neighbors selected to have at least q overlap with a particular seed, we denote t' as the number of samples overlapping with the support of the seed. The probability density for t' is positive only for $j = q, \ldots, t$,

$$\mathbb{P}(t' = j) = \frac{\binom{t}{j} p_0^j (1-p_0)^{t-j}}{\rho}$$

where $\rho = \sum_{j=q}^{t} \binom{t}{j} p_0^j (1-p_0)^{t-j}$.

In order to thin the common support, we need two new random variables. The first is a bernoulli, call it Y, which takes the value 1 with probability ρ and 0 with probability $1-\rho$. The second random variable, call it Z, takes values $j = 0, \ldots, q-1$ with probability

$$\mathbb{P}(Z = j) = \frac{\binom{t}{j} p_0^j (1-p_0)^{t-j}}{1-\rho}$$

Define $t'' = t'Y + Z(1-Y)$. The density of t'' is

$$\mathbb{P}(t''=j) = \begin{cases} \mathbb{P}(Z=j)(1-\rho) & j=0,\ldots,q-1 \\ \mathbb{P}(t'=j)\rho & j=q,\ldots,t \end{cases} \tag{5.5}$$

which equal to the desired binomial distribution. Thus, the thinning is accomplished as follows. For each column draw an independent sample of Y. If the sample is 1, then the column is not altered. If the sample is 0, then a realization of Z is drawn, which we denote by z. Select a random subset of size z from the observed entries in the seed support and discard the remainder. We note that the seed itself should not be used in completion, because there is a dependence between the sample locations of the seed column and its selected neighbors which cannot be eliminated.

Now after thinning, we have the following matrix completion guarantee for each neighborhood matrix.

Lemma 5.7. *Assume all s_0 seed neighborhood matrices are thinned according to the discussion above, have rank $\leq r$, and the matrix entries are observed uniformly at random with probability,*

$$p_0 \geq \frac{128\,\beta \max\{\mu_1^2, \mu_0\}}{\nu_0}\,\frac{r\,\log^2(n)}{n} \tag{5.6}$$

Then with probability $\geq 1 - 12 s_0 n^{2-2\beta^{1/2}} \log n$, all s_0 matrices can be perfectly completed.

Proof. First, we find that if each matrix has

$$m' \geq 64 \max\left(\mu_1^2, \mu_0\right) \beta r n \log^2(2n)$$

entries observed uniformly at random (with replacement), then with probability $\geq 1 - 12 s_0 n^{2-2\beta^{1/2}} \log n$, all s_0 matrices are perfectly completed. This follows by Lemma 5.6, the observation that

$$6\,(2n)^{2-2\beta} \log n + n^{2-2\beta^{1/2}} \leq 12 n^{2-2\beta^{1/2}} \log n\,,$$

and a simple application of the union bound.

But, under our sampling assumptions, the number of entries observed in each seed neighborhood matrix is random. Thus, the total number of observed entries in each is guaranteed to be sufficiently

large with high probability as follows. The random number of entries observed in an $n \times n$ matrix is $\widehat{m} \sim$ Binomial(p_0, n^2). By Chernoff's bound we have $\mathbb{P}(\widehat{m} \leq n^2 p_0/2) \leq \exp(-n^2 p_0/8)$. By the union bound we find that $\widehat{m} \geq m'$ entries are observed in each of the s_0 seed matrices with probability at least $1 - \exp(-n^2 p_0/8 + \log s_0)$ if $p_0 \geq \frac{128\,\beta \max\{\mu_1^2, \mu_0\}}{\nu_0} \frac{r \log^2(n)}{n}$.

Since $n^2 p_0 > rn \log^2 n$ and $s_0 = O(k(\log k + \log n))$, this probability tends to zero exponentially in n as long as $k = o(e^n)$, which holds according to Assumption **A1**. Therefore this holds with probability at least $1 - 12 s_0 n^{2-2\beta^{1/2}} \log n$. □

Finally, let us consider the third issue, the possibility that one or more of the points in the neighborhood of a seed lies in a subspace different than the seed subspace. When this occurs, the rank of the submatrix formed by the seed's neighbor columns will be larger than the dimension of the seed subspace. Without loss of generality assume that we have only two subspaces represented in the neighbor set, and assume their dimensions are r' and r''. First, in the case that $r' + r'' > r$, when a rank $\geq r$ matrix is completed to a rank r matrix, with overwhelming probability there will be errors with respect to the observations as long as the number of samples in each column is $O(r \log r)$, which is assumed in our case; see [8]. Thus we can detect and discard these candidates. Secondly, in the case that $r' + r'' \leq r$, we still have enough samples to complete this matrix successfully with high probability. However, since we have drawn enough seeds to guarantee that every subspace has a seed with a neighborhood entirely in that subspace, we will find that this problem seed is redundant. This is determined in the Subspace Refinement step.

5.3.3 Subspace Refinement

Each of the matrix completion steps above yields a low-rank matrix with a corresponding column subspace, which we will call the *candidate* subspaces. While the true number of subspaces will not be known in advance, since $s_0 = O(k(\log k + \log(1/\delta_0))$, the candidate subspaces will contain the true subspaces with high probability (see Lemma 5.4). We must now deal with the algorithmic issue of

determining the true set of subspaces.

We first note that, since the points are assumed to be drawn from a continuous distributions on the subspaces, with probability 1 a set of points of size $\geq r$ all drawn from a single subspace $\mathcal{S}$ of dimension $\leq r$ will span $\mathcal{S}$. In fact, any $b < r$ points will span a b-dimensional subspace of the r-dimensional subspace $\mathcal{S}$.

Assume that $r < n$, since otherwise it is clearly necessary to observe all entries. Therefore, if a seed's nearest neighborhood set is confined to a single subspace, then the columns in span their subspace. And if the seed's nearest neighborhood contains columns from two or more subspaces, then the matrix will have rank larger than that of any of the constituent subspaces. Thus, if a certain candidate subspace is spanned by the union of two or more smaller candidate subspaces, then it follows that that subspace is not a true subspace (since we assume that none of the true subspaces are contained within another).

This observation suggests the following subspace refinement procedure. The s_0 matrix completions yield $s \leq s_0$ candidate column subspaces; s may be less than s_0 since completions that fail are discarded as described above. First sort the estimated subspaces in order of rank from smallest to largest (with arbitrary ordering of subspaces of the same rank), which we write as $\mathcal{S}_{(1)}, \ldots, \mathcal{S}_{(s)}$. We will denote the final set of estimated subspaces as $\widehat{\mathcal{S}}_1, \ldots, \widehat{\mathcal{S}}_k$. The first subspace $\widehat{\mathcal{S}}_1 := \mathcal{S}_{(1)}$, a lowest-rank subspace in the candidate set. Next, $\widehat{\mathcal{S}}_2 = \mathcal{S}_{(2)}$ if and only if $\mathcal{S}_{(2)}$ is not contained in $\widehat{\mathcal{S}}_1$. Following this simple sequential strategy, suppose that when we reach the candidate $\mathcal{S}_{(j)}$ we have so far determined $\widehat{\mathcal{S}}_1, \ldots, \widehat{\mathcal{S}}_i$, $i < j$. If $\mathcal{S}_{(j)}$ is not in the span of $\cup_{\ell=1}^{i} \widehat{\mathcal{S}}_\ell$, then we set $\widehat{\mathcal{S}}_{i+1} = \mathcal{S}_{(j)}$, otherwise we move on to the next candidate. In this way, we can proceed sequentially through the rank-ordered list of candidates, and we will identify all true subspaces.

If there is more than one subspace of a particular dimension, these subspaces must be pruned by combinatorially by looking at which collection of subspaces is minimal, in the sense that the fewest subspaces are included in the true collection so as to span all the subspaces, both included and excluded.

5.3.4 Subspace Assignment

At this point, we have identified the true subspaces, and all N columns lie in the span of one of those subspaces. For ease of presentation, we assume that the number of subspaces is exactly k. However if columns lie in the span of fewer than k, then the procedure above will produce the correct number. To complete the full matrix, we proceed one column at a time. For each column of $\boldsymbol{X}_\Omega$, we determine the correct subspace to which this column belongs, and we then complete the column using that subspace. We can do this with high probability due to results from [8, 7].

The first step is that of subspace assignment, determining the correct subspace to which this column belongs. In [7], it is shown that given k subspaces, an incomplete vector can be assigned to its closest subspace with high probability given enough observations. In the situation at hand, we have a special case of the results of [7] because we are considering the more specific situation where our incomplete vector lies exactly in one of the candidate subspaces, and we have an upper bound for both the dimension and coherence of those subspaces.

Lemma 5.8. *Let $\{\mathcal{S}_1, \ldots, \mathcal{S}_k\}$ be a collection of k subspaces of dimension $\leq r$ and coherence parameter bounded above by μ_0. Consider column vector x with index set $\Omega \in \{1, \ldots, n\}$, and define $P_{\Omega,\mathcal{S}_j} = U^j_\Omega \left(\left(U^j_\Omega\right)^T U^j_\Omega \right)^{-1} \left(U^j_\Omega\right)^T$, where U^j is the orthonormal column span of $\mathcal{S}_j$ and U^j_Ω is the column span of $\mathcal{S}_j$ restricted to the observed rows, Ω. Without loss of generality, suppose the column of interest $x \in \mathcal{S}_1$. If* **A3** *holds, and the probability of observing each entry of x is independent and Bernoulli with parameter*

$$p_0 \geq \frac{128\,\beta \max\{\mu_1^2, \mu_0\}}{\nu_0} \, \frac{r \, \log^2(n)}{n} \, .$$

Then with probability at least $1 - (3(k-1)+2)\delta_0$,

$$\|x_\Omega - P_{\Omega,\mathcal{S}_1} x_\Omega\|_2^2 = 0 \tag{5.7}$$

and for $j = 2, \ldots, k$

$$\|x_\Omega - P_{\Omega,\mathcal{S}_j} x_\Omega\|_2^2 \; > \; 0 \, . \tag{5.8}$$

Proof. We wish to use results from [8, 7], which require a fixed number of measurements $|\Omega|$. By Chernoff's bound

$$\mathbb{P}\left(|\Omega| \leq \frac{np_0}{2}\right) \leq \exp\left(\frac{-np_0}{8}\right).$$

Note that $np_0 > 16r\beta\log^2 n$, therefore $\exp\left(\frac{-np_0}{8}\right) < (n^{-2\beta})^{\log n} < \delta_0$; in other words, we observe $|\Omega| > np_0/2$ entries of x with probability $1-\delta_0$. This set Ω is selected uniformly at random among all sets of size $|\Omega|$, but using Lemma 5.1 we can assume that the samples are drawn uniformly with replacement in order to apply results of [8, 7].

Now we show that $|\Omega| > np_0/2$ samples selected uniformly with replacement implies that

$$|\Omega| > \max\left\{\frac{8r\mu_0}{3}\log\left(\frac{2r}{\delta_0}\right), \frac{r\mu_0(1+\xi)^2}{(1-\alpha)(1-\gamma)}\right\} \tag{5.9}$$

where $\xi, \alpha > 0$ and $\gamma \in (0,1)$ are defined as $\alpha = \sqrt{\frac{2\mu_1^2}{|\Omega|}\log\left(\frac{1}{\delta_0}\right)}$, $\xi = \sqrt{2\mu_1\log\left(\frac{1}{\delta_0}\right)}$, and $\gamma = \sqrt{\frac{8r\mu_0}{3|\Omega|}\log\left(\frac{2r}{\delta_0}\right)}$.

We start with the second term in the max of (5.9). Substituting δ_0 and the bound for p_0, one can show that for $n \geq 15$ both $\alpha \leq 1/2$ and $\gamma \leq 1/2$. This makes $(1+\xi)^2/(1-\alpha)(1-\gamma) \leq 4(1-\xi)^2 \leq 8\xi^2$ for $\xi > 2.5$, i.e. for $\delta_0 < 0.04$.

We finish this argument by noting that $8\xi^2 = 16\mu_1\log(1/\delta_0) < np_0/2$; there is in fact an $O(r\log(n))$ gap between the two. Similarly for the first term in the max of (5.9), $\frac{8}{3}r\mu_0\log\left(\frac{2r}{\delta_0}\right) < np_0/2$; here the gap is $O(\log(n))$.

Now we prove (5.7), which follows from [8]. With $|\Omega| > \frac{8}{3}r\mu_0\log\left(\frac{2r}{\delta_0}\right)$, we have that $U_\Omega^T U_\Omega$ is invertible with probability at least $1-\delta_0$ according to Lemma 3 of [8]. This implies that

$$U^T x = \left(U_\Omega^T U_\Omega\right)^{-1} U_\Omega^T x_\Omega\,. \tag{5.10}$$

Call $a_1 = U^T x$. Since $x \in \mathcal{S}$, $Ua_1 = x$, and a_1 is in fact the unique solution to $Ua = x$. Now consider the equation $U_\Omega a = x_\Omega$. The assumption that $U_\Omega^T U_\Omega$ is invertible implies that $a_2 = \left(U_\Omega^T U_\Omega\right)^{-1} U_\Omega^T x_\Omega$

exists and is the unique solution to $U_\Omega a = x_\Omega$. However, $U_\Omega a_1 = x_\Omega$ as well, meaning that $a_1 = a_2$. Thus, we have

$$\|x_\Omega - P_{\Omega,\mathcal{S}_1} x_\Omega\|_2^2 = \|x_\Omega - U_\Omega U^T x\|_2^2 = 0$$

with probability at least $1 - \delta_0$.

Now we prove (5.8), paralleling Theorem 1 in [7]. We use Assumption **A3** to ensure that $x \notin \mathcal{S}_j$, $j = 2, \ldots, k$. This along with (5.9) and Theorem 1 from [8] guarantees that

$$\|x_\Omega - P_{\Omega,\mathcal{S}_j} x_\Omega\|_2^2 \geq \frac{|\Omega|(1-\alpha) - r\mu_0 \frac{(1+\xi)^2}{1-\gamma}}{n} \|x - P_{\mathcal{S}_j} x\|_2^2 > 0$$

for each $j = 2, \ldots, k$ with probability at least $1 - 3\delta_0$. With a union bound this holds simultaneously for all $k-1$ alternative subspaces with probability at least $1 - 3(k-1)\delta_0$. When we also include the events that (5.7) holds and that $|\Omega| > np_0/2$, we get that the entire theorem holds with probability at least $1 - (3(k-1)+2)\delta_0$. □

Finally, denote the column to be completed by x_Ω. To complete x_Ω we first determine which subspace it belongs to using the results above. For a given column we can use the *incomplete data projection residual* of (5.7). With probability at least $1 - (3(k-1)+2)\delta_0$, the residual will be zero for the correct subspace and strictly positive for all other subspaces. Using the span of the chosen subspace, U, we can then complete the column by using $\widehat{x} = U \left(U_\Omega^T U_\Omega\right)^{-1} U_\Omega^T x_\Omega$.

We reiterate that Lemma 5.8 allows us to complete a single column x with probability $1-(3(k-1)+2)\delta_0$. If we wish to complete the entire matrix, we will need another union bound over all N columns, leading to a $\log N$ factor in our requirement on p_0. Since N may be quite large in applications, we prefer to state our result in terms of per-column completion bound.

The confidence level stated in Theorem 5.1 is the result of applying the union bound to all the steps required in the Sections 3, 4, and 6. All hold simultaneously with probability at least

$$1 - (6 + 3(k-1) + 12s_0)\,\delta_0 \quad < \quad 1 - (6 + 15s_0)\delta_0\,,$$

which proves the theorem.

5.4 Algorithm: k-GROUSE for Subspace Clustering with Missing Data

Though the algorithm in Section 5.2 has provable guarantees, it is clear that the algorithm is not computationally efficient. The GROUSE algorithm [6], or Grassmannian Rank-One Update Subspace Estimation, was developed in Section 4.1 as an efficient way to do single subspace estimation with highly incomplete data vectors. Recall Theorem 3.2.3, which says that from a collection of subspaces we can find the closest subspace to an arbitrary incomplete vector as long as we have enough measurements of that vector. With this in mind we can combine k-subspaces and GROUSE to multiple subspace estimation. The standard k-subspaces algorithm is described in [18, 1], and we also review it in Algorithm 4. Our version, k-GROUSE, is incremental and based on GROUSE to allow for flexibility when observations are incomplete. This work is joint with Arthur Szlam [7].

In this section we return to the notation where d is the inherent dimension of our subspace, and we use r to denote the residual vectors.

Algorithm 4 K-Subspaces for Subspace Clustering

Require: An $n \times T$ data matrix V made up of sequence of vectors v_t, $t = 1, \ldots, T$. An integer number of subspaces k and dimensions d_i, $i = 1, \ldots, k$. A maximum number of iterations, maxIter.

1: **Initialize Subspaces:** Initialize k subspace estimates using data. Find orthonormal matrices U_i, $i = 1, \ldots, k$ whose columns span the k subspaces.
2: **for** $iter = 1, \ldots,$ maxIter **do**
3: **for** $t = 1, \ldots, T$ **do**
4: **Calculate Projection Residuals to k subspaces:** $r_t(k) = \|v_t - U_i U_i^T v_t\|_2^2$, $i = 1, \ldots, k$
5: **Select min residual:** clusterid$_t = \mathrm{argmin}_k r_t(k)$
6: **end for**
7: **for** $i = 1, \ldots, k$ **do**
8: **Collect vectors assigned to i^{th} subspace.** Call this new matrix V_i.
9: **Compute SVD:** $V_i = Y_i \Sigma_i Z_i^T$
10: **Update subspace**: $U_i = Y_i$
11: **end for**
12: **end for**

To initialize the subspaces we use a version of probabilistic farthest insertion, as in [79], modified for missing data by simply zero-filling the unobserved entries in each vector and collecting them in a matrix V. Specifically, we pick a random point as the first cluster "center," $v_0 \in V$. We then calculate

Algorithm 5 k-Subspaces with the GROUSE: batch

Require: An $n \times T$ data matrix V made up of sequence of vectors $v_t, t = 1, \ldots, T$. An integer number of subspaces k and dimensions $d_i, i = 1, \ldots, k$. A maximum number of iterations, maxIter. A step size η.

1: **Initialize Subspaces:** Initialize k subspace estimates using data. Find orthonormal matrices U_j, $j = 1, \ldots, k$ whose columns span the k subspaces.
2: **for** $i = 1, \ldots,$ maxIter **do**
3: **for** $t = 1, \ldots, T$ **do**
4: **Calculate projection residuals to k subspaces:** $r_t(k) = \|v_t - U_j U_j^T v_t\|_2^2$, $j = 1, \ldots, k$
5: **Select min residual:** clusterid$_t = \operatorname{argmin}_j r_t(j)$
6: **end for**
7: **for** $j = 1, \ldots, k$ **do**
8: **Collect vectors assigned to j^{th} subspace** into V_j.
9: **Update subspace using the grouse [6]:**
 $U_j = \text{grouse}(U_j, V_j, \eta)$
10: **end for**
11: **end for**

Algorithm 6 k-subspaces with the GROUSE: incremental

Require: A collection of vectors $v_\Omega(t), t = 1, \ldots, T$, and the observed indices $\Omega(t)$. An integer number of subspaces k and dimensions $d_i, i = 1, \ldots, k$. A maximum number of iterations, maxIter. A fixed step size η.

1: **Initialize Subspaces:** Zero-fill the vectors and collect them in a matrix V. Initialize k subspace estimates using probabilistic farthest insertion.
2: **Calculate Orthonormal Bases** $U_j, j = 1, \ldots, k$.
 Let $Q_{j\Omega} = (U_{j\Omega}^T U_{j\Omega})^{-1} U_{j\Omega}^T$
3: **for** $i = 1, \ldots,$ maxIter **do**
4: **Select a vector at random,** v_Ω.
5: **for** $j = 1, \ldots, k$ **do**
6: **Calculate projection weights:** $w(j) = Q_{j\Omega} v_\Omega$.
7: **Calculate residual:** $r(j) = \|v_\Omega - U_{j\Omega} w(j)\|_2^2$.
8: **end for**
9: **Select min residual:** $\widehat{j} = \operatorname{argmin}_j r(j)$. Set $r = r(\widehat{j})$ and $w = w(\widehat{j})$. Define $p = v_0 - r$, where v_0 is zero-filled v_Ω.
10: **Update subspace**:
 $U_{\widehat{j}} = U_{\widehat{j}} + \left((\cos(\sigma\eta) - 1) \frac{p}{\|p\|} + \sin(\sigma\eta) \frac{r}{\|r\|} \right) \frac{w^T}{\|w\|}$
 where $\sigma = \|r\| \|p\|$
11: **end for**

the $d + q$ nearest neighbors to v_0, where $d := \max_i d_i$ and q is a nonnegative parameter [112], and calculate the best fit subspace S^0 to the neighborhood of v_0. For the next center we choose another random point with probability proportional to the distance $dist(v, S^0)^2$, and find the best fit subspace S^1 of its $d + q$ neighborhood. For j^{th} neighborhood, we pick the center with probability proportional to $\min(dist(v, S^0)^2, ..., dist(v, S^{j-1})^2)$.

To refine the initial subspaces, the incremental algorithm k-GROUSE is presented in Algorithm 6. It is a form of sequential k-means adapted to k subspaces. In each iteration, a single incomplete vector v_Ω is chosen, and the closest subspace is found by using Corollary 3.3.2. Then this closest subspace is updated via GROUSE with the data vector v_Ω. This is repeated several times until some criteria are met. We note that this algorithm works as written for the case when the data vector is complete.

With matrix completion in mind, one may also consider a batch version of k-subspaces. The batch version would simply use GROUSE subspace estimation or any other matrix completion algorithm, such as the one found in [99], in place of the SVD step for subspace estimation used in the standard k-subspaces algorithm [18, 1, 101]. Given a cluster of vectors, matrix completion would be performed to get a subspace estimate. Then with these estimates, vectors would be reassigned, and the process repeated.

As written, k-GROUSE and this suggested batch version both require knowledge of the number of subspaces k and their dimensions, whereas the algorithm in [3] only requires an upper bound on both values. Our simulations show scenarios where the subspaces are of the same dimension, but the algorithms do not require this.

5.5 Empirical Analysis

We begin by examining the success of the first algorithm presented in Section 5.2 as compared to using low-rank matrix completion. For Figure 21, the key parameters were chosen as follows: $n = 100$, $N = 5000$, $k = 10$, and $d = 5$. The k subspaces were d-dimensional, and each was generated by

d vectors drawn from the $\mathcal{N}(0, I_n)$ distribution and taking their span. The resulting subspaces are highly incoherent with the canonical basis for $\mathbb{R}^n$. For each subspace, we generate 500 points drawn from a $\mathcal{N}(0, UU^T)$ distribution, where U is a $n \times d$ matrix whose orthonormal columns span the subspace. Our procedure was implemented using $\lceil 3k \log k \rceil$ seeds. The matrix completion software called GROUSE (available here [9]) was used in our procedure and to implement the standard low-rank matrix completions. We ran 50 independent trials of our procedure and compared it to standard low-rank matrix completion. The results are summarized in the figures below. The key message is that our new procedure can provide accurate completions from far fewer observations compared to standard low-rank completion, which is precisely what our main result predicts.

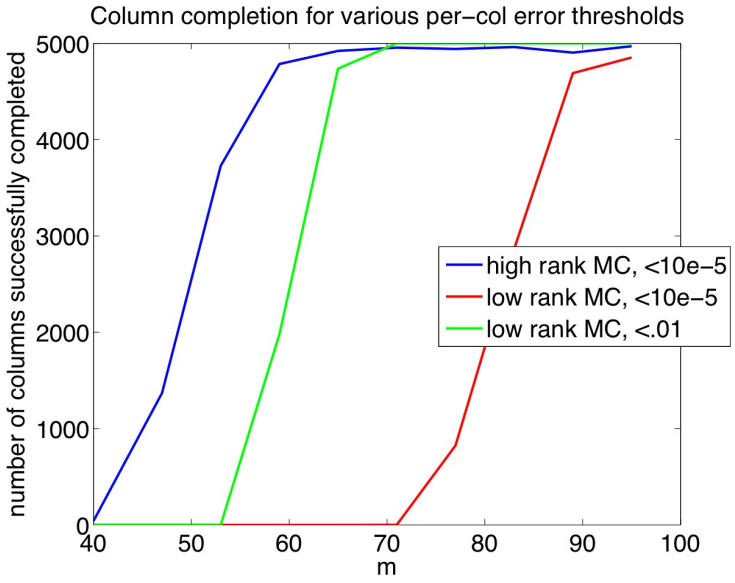


Figure 21: The number of correctly completed columns (with tolerances shown above, 10e-5 or 0.01), versus the average number of observations per column. As expected, our procedure (termed high rank MC in the plot) provides accurate completion with only about 50 samples per column. Note that $d \log n \approx 23$ in this simulation, so this is quite close to our bound. On the other hand, since the rank of the full matrix is $dk = 50$, the standard low-rank matrix completion bound requires $m > 50 \log n \approx 230$. Therefore, it is not surprising that the standard method (termed low rank MC above) requires almost all samples in each column.

Next we look at the algorithm of Section 5.2 as compared to Algorithm 6, or k-GROUSE. At this point we note that if the sum of the dimensions of the subspaces $D := \sum_{j=1}^{k} d_j$ is significantly less than n, a two-stage approach to subspace clustering is to first perform matrix completion on the

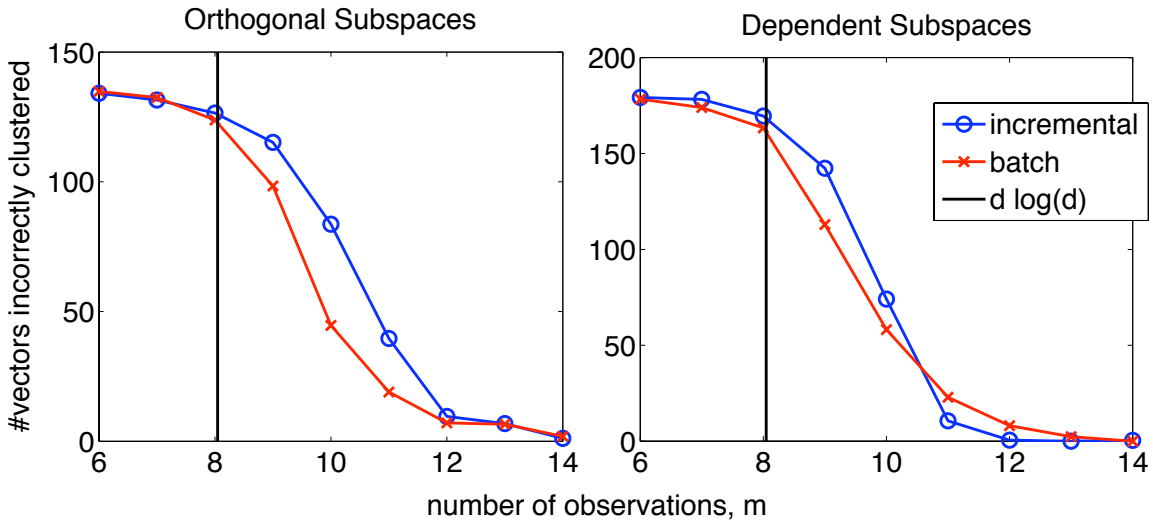


Figure 22: Simulation results: On the left we have $n = 20$, $d = 5$, $k = 4$ ($D = n$) and orthogonal subspaces. On the right we have $k = 5$ ($D > n$) and thus linearly dependent subspaces. The error measure is defined in Equation 5.12. The curves shown are averaged over 100 random observation sets.

data matrix V to recover a rank D matrix and then apply any full-data subspace clustering algorithm. However, there are two situations when this is not possible. First, it may be that D is actually greater than or equal to n; we explore this scenario in Figure 22. Second, we may have collected the $d\log(d)$ observations per vector which are sufficient for subspace assignment, but not $D\log(n)$ observations which are sufficient for matrix completion. This is the setup of the simulation of Table 2.

We show the results of three simulation scenarios. In the first, data vectors come from subspaces which are orthogonal, and the sum of the dimensions of the subspaces is the ambient dimension: $D = n$. The left plot of Figure 22 shows the results. The data matrix consists of $N = 200$ points for the left plot and $N = 300$ points for the right point, i.e. 50 vectors per subspace. The parameter for nearest neighbor subspace estimation is $q = 5$.

The error is calculated as compared to ground truth. Let A_j, $j = 1, \ldots, k$ be sets of indices corresponding to ground-truth cluster assignments. Let B_j be the cluster assignments chosen by the algorithm. For $l = 1, \ldots, k$, we find

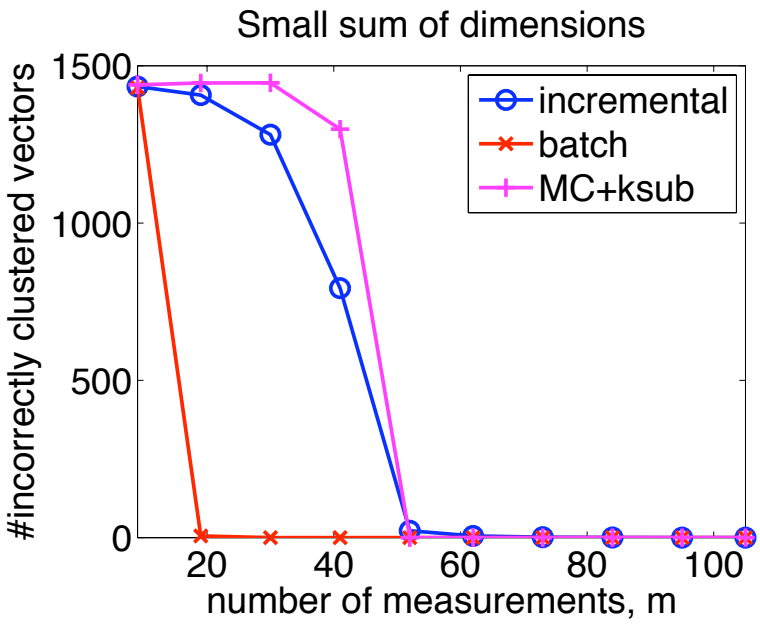


Figure 23: For this simulation, $n = 150$, $d = 5$, $k = 6$, $N = 300$, and the dimension of the union of subspaces is $D = 30$. Results are averaged over 10 random observation sets.

$$\widehat{j}_l = \text{argmax}_j |B_j \cap A_l| \ , \tag{5.11}$$

where $|\cdot|$ denotes the cardinality of a set. Then the error is

$$\sum_{l=1}^{k} |A_l \backslash \{A_l \cap B_{\widehat{j}_l}\}| \ . \tag{5.12}$$

We note that this error can be minimized trivially by an algorithm which assigns all the vectors to one cluster; however these algorithms also minimize distance to low dimensional subspaces, and we have verified that the clusters are about the correct size. Results for $D > n$ can be seen in the right hand plot of Figure 22. The third scenario is one where sum of the dimensions of the subspaces is less than the ambient dimension, $D < n$. Here we compare to the two-stage approach of matrix completion plus full-data k-planes clustering. Once there are enough measurements to estimate the rank-D matrix, all the algorithms perform with zero error. However in the low-observation regime, there are still enough measurements to estimate each of the k subspaces, and both the batch and incremental algorithms

outperform the two-stage approach of matrix completion followed by k-planes clustering.

The main benefit of k-GROUSE is its speed. The algorithm of Section 5.2 requires that the number of matrix columns $N = n^p$ for some $p \geq 2$ in order to guarantee that local neighborhoods can be found despite missing data. Then distances must be computed between $O(k \log k)$ seed columns and all these N columns using a mask, unique for every pair, that identifies the shared observations between those two columns. Finally, the refinement step can be combinatorial. By contrast, k-GROUSE uses the rough initialization using zero-filled distances, and then each incremental update only takes $O(kmd^2 + nd)$ time. In Table 2 we show results of simulations run in Matlab on a Dell Precision T5500n with a Dual Quad Core Intel Xeon 2.53GHz processor and 12 GB of RAM. Clearly k-GROUSE far outpaces the algorithm in Section 5.2. More importantly, it even performs an order of magnitude faster than the batch heuristic algorithm.

	Computation Time (sec)		% successful
Algorithm	average	std. dev.	trials
High Rank Matrix Completion with $3k \log k$ seeds	10395.0	655.8	56
High Rank Matrix Completion with $10k \log k$ seeds	34162.3	2086.5	100 (of 11 trials)
Algorithm 5, batch k-GROUSE	1079.5	17.8	97
Algorithm 6, k-GROUSE	127.6	0.24	93

Table 2: The problem size is $n = 50$, $k = 10$, $d = 4$, ($D < n$) and $N = 40,000$. 60% of the entries were sampled. Successful trials are those in which the clustering was exactly correct. This percentage is from 100 trials unless otherwise noted. High Rank Matrix Completion is the algorithm from Section 5.2, which with $3k \log k$ seeds took more than 2.5 hours on average whereas Algorithm 6 took 2 minutes.

Chapter 6

Subspace Tracking with Missing Data

As discussed in Chapter 4, the GROUSE algorithm implements an incremental gradient procedure with computational complexity linear in dimensions of the problem, and thus it can be immediately adapted to 'online' subspace estimation. In this chapter we explore the tracking capabilities of GROUSE in numerical simulation.

There are other subspace tracking algorithms which operate on batches of data; these are difficult to to adapt to the missing-data scenario because of the lack of overlap of the observations from one column vector to the next. Especially in a highly undersampled regime, there will be very little overlap, and that is additional motivation for doing subspace updates one vector at a time.

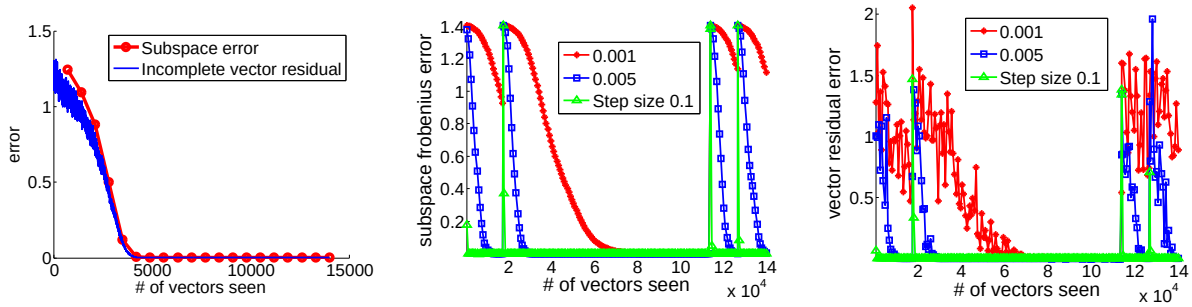
We then discuss the robust extension GRASTA (Grassmannian Robust Adaptive Subspace Tracking Algorithm), an algorithm which parallels GROUSE but allows robust subspace identification when the signal has a sparse outlier component. GRASTA was developed by Jun He [52], and in this thesis we focus on studying the relationship of step-size and iterations with the tracking capabilities of GRASTA.

6.1 Subspace Tracking with GROUSE

6.1.1 Empirical Analysis

In these simulations, we focus on GROUSE's tracking capabilities.

Subspace Change Detection and Tracking As a first example of GROUSE's ability to adapt to changes in the underlying subspace, we simulated a scenario where the underlying subspace abruptly



(a) (b) (c)

Figure 24: (a) Comparison of the distance to the true subspace and the norm of the residual in Step 4 of the GROUSE algorithm. The residual norm closely tracks the distance to the actual subspace. (b) Using constant stepsize to track sudden changes in subspace. We plot the transient behavior three constant stepsize policies. In (c), we again verify that the norm of the residual gives an accurate signature for anomaly detection and for tracking success.

changes at three points over the course of an experiment with 14000 observations. At each break, we selected a new subspace S uniformly at random and GROUSE was implemented with a constant stepsize. As is to be expected, the algorithm is able to re-estimate the new subspace in a time depending on the magnitude of the constant stepsize.

Rotating Subspace

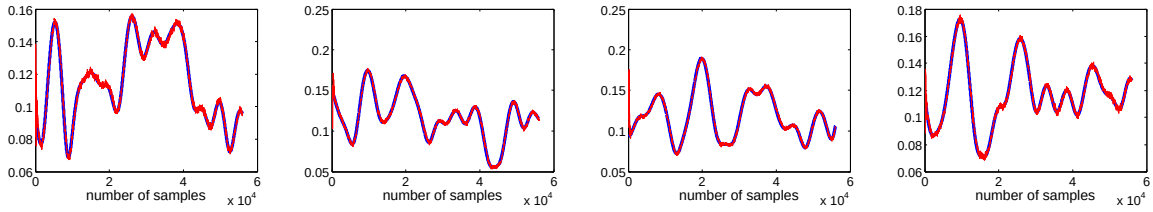
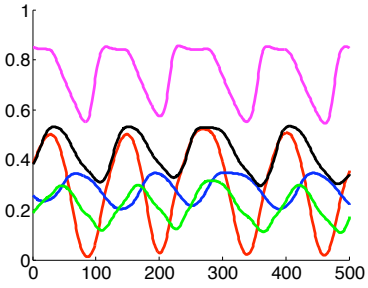


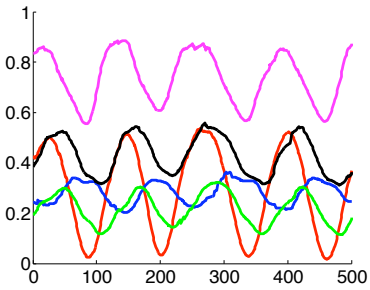
Figure 25: Tracking a rotating subspace. Here we plot the norm of the projection of four random vectors over time. The blue curves denote the true values of these norms, and the red curves plot the GROUSE predictions. Note that except for very early transients, GROUSE fully tracks the subspace.

In this second synthetic experiment, the subspace evolves according to a random ordinary differential equation. Specifically, we sample a skew-symmetric matrix B with independent, normally distributed entries and set

$$\dot{U} = BU\,, \quad U[0] = U_0.$$



(a)



(b)

$\|\Omega_t\|/n$	η_t	reconstruction error
0.2	5e-3	0.2530
	3e-2	0.1244
0.4	5e-3	0.1797
	1e-2	0.1233
	3e-2	0.1432
0.7	5e-3	0.1289
	7e-3	0.1221
	3e-2	0.1684
1	5e-3	0.1253
	3e-2	0.2217

Figure 26: (a) Actual sensor readings. (b) Predicted sensor readings from tracked subspace. In these figures we are displaying the values at sensors 4, 17, 148,158, and159. The table lists errors in tracking the chlorine data set for varying sampling densities and stepsizes. The error between the data and the best SVD approximation is 0.0704.

The resulting subspace at each iteration is thus $U[t] = \exp(\delta t B)$ where δ is a positive constant. The resulting subspace at each iteration is thus $U[t] = \exp(\delta t B)$ where δ is a positive constant. In Figure 25, we show the results of tracking the rotating subspace with $\delta = 10^{-5}$. To demonstrate the effectiveness of the tracking, we display the projection of four random vectors using both the true subspace (in blue) and our subspace estimate at that time instant (in red).

Tracking Chlorine Levels

We also analyzed the performance of the GROUSE algorithm on simulated chlorine level monitoring in a pressurized water delivery system. The data were generated using EPANET [1] software and were previously analyzed [80]. The input to EPANET is a network layout of water pipes and the output has many variables including the chemical levels, one of which is the chlorine level. The data we used is available from [80] [2]. This dataset has ambient dimension $n = 166$, and $T = 4610$ data vectors were collected, once every 5 minutes over 15 days. We tracked an $d = 6$ dimensional subspace and compare this with the best 6-dimensional SVD approximation of the entire complete dataset. The results are displayed in Figure 26. The table gives the results for various constant stepsizes and various

[1] http://www.epa.gov/nrmrl/wswrd/dw/epanet.html
[2] http://www.cs.cmu.edu/afs/cs/project/spirit-1/www/

fractions of sampled data. The smallest sampling fraction we used was 20%, and for that the best stepsize was 3e-2; we also ran GROUSE on the full data, whose best stepsize was 5e-3. As we can see, the performance error improves for the smaller stepsize of 5e-3 as the sampling fraction increases; Also the performance error improves for the larger stepsize of 3e-2 as the sampling fraction decreases. However for all intermediate sampling fractions there are intermediate stepsizes that perform near the best reconstruction error of about 0.12. The normalized error of the full data to the best 3-dimensional SVD approximation is 0.0704. Note that we only allow for one pass over the data set and yet attain, even with very sparse sampling, comparable accuracy to a full batch SVD which has access to all of the data.

Figure 26(a) and (b) show the original and the GROUSE reconstructions of five of the chlorine sensor outputs. We plot the last 500 of the 4310 samples, each reconstructed by the estimated subspace at that time instant.

6.2 Robust Tracking

Subspace tracking in the presence of outliers is an important extension for modeling of complex systems with massive and distributed data collection. In this section we provide two approaches to solve such a problem. The work in this section is joint with Jun He and Arthur Szlam [52, 53].

6.2.1 Robust Tracking by Outlier Detection

In Section 4.1 and Section 6.1 we showed that one can perform subspace tracking of a d-dimensional subspace of $\mathbb{R}^n$ with many fewer than all n measurements. It is natural then to observe that if the outlier locations are known, we can do subspace tracking in the presence of outliers as well by simply removing the outliers from the update. For this section we will assume that the initial subspace $\mathcal{S}_0$ is known and used as our initialization point. This can be interpreted as an assumption that there is a period of data collection at the start when the measurements do not have corruptions.

We denote the evolving d-dimensional subspace of $\mathbb{R}^n$ as $\mathcal{S}_t$ at time t. As before, let the columns of an $n \times d$ matrix U_t be orthonormal and span $\mathcal{S}_t$.

At each time step t, we assume that v_t is generated by the following model:

$$v_t = U_t w_t + s_t + \zeta_t \tag{6.1}$$

where w_t is the $d \times 1$ weight vector, s_t is the $n \times 1$ sparse outlier vector with support on a set $\mathcal{O}_t \subset \{1, \ldots, n\}$ whose nonzero entries may be arbitrarily large, and ζ_t is the $n \times 1$ zero-mean Gaussian white noise vector with small variance.

To do robust tracking with outlier detection means we wish to find some set $\widehat{\mathcal{O}}_t$ on which to do the subspace update for time t. We suggest several ways to estimate the sparse support of the outliers.

1. Estimate $\mathcal{O}_t$ as the largest residuals of $v_t - U_t w$, where $w = \arg\min_a \|v_t - U_t a\|_2$.

2. Solve Least Absolute Regression to estimate $w = \arg\min_a \|v_t - U_t a\|_1$ and again use the largest residuals of $v_t - U_t w$ to estimate $\mathcal{O}_t$.

3. Randomize the subset selection of #1: instead of selecting indices corresponding to the largest residuals, select an index with probability proportional to its residual. This soft decision avoids biasing our updated subspace because of one bad choice.

4. If we have the computational resources to do the l_1-norm minimization, but are concerned about sensitivity to change or noise, we can also randomize the subset selection out of the result of #3.

Empirical Analysis

Here we show the various suggested approaches, first in a noise-free simulation with a stationary subspace, and then with added noise and with a non-stationary subspace. In Figure 27 we can see that with a stationary subspace and no noise, the approach #1, 3 do not work because of the sensitivity of l^2 to outliers; both our approaches #2 and #4 achieve oracle tracking performance (as if the outlier set were

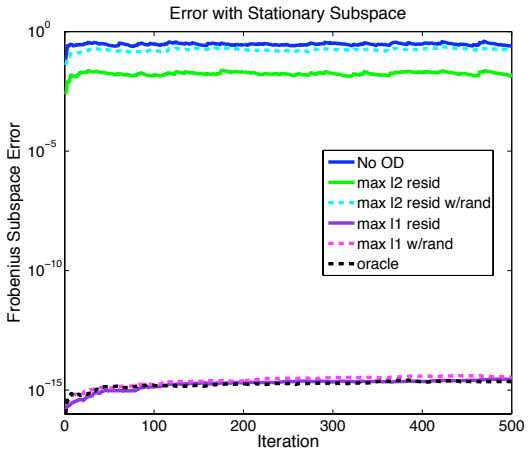


Figure 27: Tracking a stationary subspace. For this simulation, the ambient dimension $n = 500$, the inherent dimension $d = 5$, and the number of corrupted entries is 50.

known). This is also the case with added noise. However when the subspace is rotating as in Figure 29, randomization is crucial, and #4 is the only successful approach.

6.2.2 Robust Tracking by GRASTA

GRASTA [52] provides a robust extension of GROUSE by replacing the l^2-norm cost function with an l^1-norm cost function. For each subspace update, we use the gradient of the augmented Lagrangian function associated to this cost. GRASTA operates only one data vector at a time, making it faster than other state-of-the-art algorithms and amenable to streaming and real-time applications.

GRASTA alternates between estimating a low-dimensional subspace $\mathcal{S}$ and a triple (s, w, y) which represent the sparse corruptions in the signal, the weights for the fit of the signal to the subspace, and the dual vector in the optimization problem. For estimating the subspace $\mathcal{S}$, GRASTA uses gradient descent on the Grassmannian with (s, w, y) fixed; for estimating the triple (s, w, y), GRASTA uses ADMM [16].

As an example, we consider using subspaces to detect anomalies in computer networks [61]. A

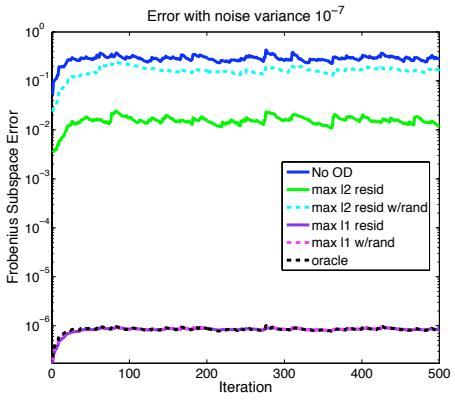


Figure 28: Tracking a stationary subspace with additive noise on the measurements.

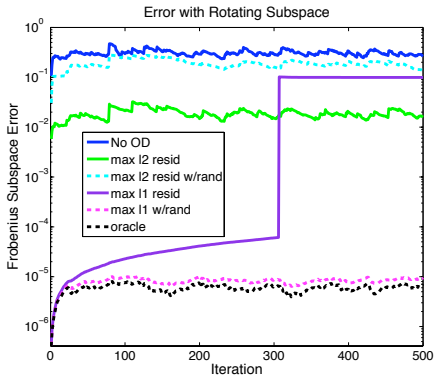


Figure 29: Tracking a rotating subspace. Here the subset choice randomization is critical.

non-robust subspace estimation algorithm like GROUSE would need a special anomaly detection component in order to differentiate anomalies and outliers from the underlying subspace of the traffic data. Often these types of anomaly detection components rely on a lot of parameter tuning and heuristic rules for detection. This motivates a more principled approach that is robust by design: GRASTA.

We stress here that the GRASTA algorithm was derived by Dr. Jun He and is included here for completeness. The contribution of the current thesis is the exposition relating this robust extension of GROUSE.

Problem Formulation

We start with the same measurement model as Equation 6.1, and additionally suppose we observe only a small subset of entries of v_t, denoted by $\Omega_t \subset \{1, \ldots, n\}$. Conforming to the notation of Section 4.1, we let U_{Ω_t} denote the submatrix of U_t consisting of the rows indexed by Ω_t; also for a vector $v_t \in \mathbb{R}^n$, let v_{Ω_t}denote a vector in $\mathbb{R}^{|\Omega_t|}$ whose entries are those of v_t indexed by Ω_t. Recall that in Section 4.1 we used the natural Euclidean distance, the l^2-norm, to measure the subspace error from the subspace spanned by the columns of U_t to the observed vector v_{Ω_t}:

$$F_{grouse}(\mathcal{S};t) = \min_{w} \|U_{\Omega_t} w - v_{\Omega_t}\|_2^2 \,. \tag{6.2}$$

It was shown in Section3.2.1 that this cost function gives an accurate estimate of the same cost function with full data ($\Omega = \{1,\ldots,n\}$), as long as $|\Omega_t| > \frac{8}{3}\mu(\mathcal{S})d\log(2d/\delta)$, where $\mu(\mathcal{S})$ is a measure of incoherence on the subspace and δ controls the probability of the result. However, if the observed data vector is corrupted by outliers as in Equation (6.1), an l^2-based best-fit to the subspace can be influenced arbitrarily with just one large outlier; this in turn will lead to an incorrect subspace update in the GROUSE algorithm, as is demonstrated in [52].

In order to quantify the subspace error robustly, we use the l^1-norm as follows:

$$F_{grasta}(\mathcal{S};t) = \min_{w} \|U_{\Omega_t} w - v_{\Omega_t}\|_1 \,. \tag{6.3}$$

With U_{Ω_t} known (or estimated, but fixed), this l^1 minimization problem is the classic least absolute deviations problem; Boyd [16] has a nice survey of algorithms to solve this problem and describes in detail a fast solver based on the technique of ADMM (Alternating Direction Method of Multipliers)[3]. More references can be found therein.

The augmented Lagrangian of this constrained minimization problem is then

$$\mathcal{L}(s,w,y) = \|s\|_1 + y^T(U_{\Omega_t} w + s - v_{\Omega_t}) + \frac{\rho}{2}\|U_{\Omega_t} w + s - v_{\Omega_t}\|_2^2 \tag{6.4}$$

where y is the dual vector. Our unknowns are s, y, U, and w. Note that since U is constrained to a non-convex manifold ($U^TU = I$), this function is not convex (neither is Equation (6.2)). However, note that if U were estimated, we could solve for the triple (s,w,y) using ADMM; also if (s,w,y) were estimated, we could refine our estimate of U. This is the alternating approach we take with GRASTA. We describe the two parts in detail in Section 6.2.2.

[3] `http://www.stanford.edu/~boyd/papers/admm/`

Relation to Robust PCA and Robust Matrix Completion

If the subspace $\mathcal{S}$ does not evolve over time, this problem reduces to subspace estimation, which can be related to Robust PCA. For a set of time samples $t = 1, \ldots, T$, we observe a sequence of incomplete corrupted data vectors $v_{\Omega_1}, \ldots, v_{\Omega_T}$. Let the matrix $V = [v_1, \ldots, v_T]$. Let $\mathcal{P}_\Omega(\cdot)$ denote operator which selects from each column the corresponding indices in $\Omega_1, \ldots, \Omega_T$; thus $\mathcal{P}_\Omega(V)$ denotes our partial observation of the corrupted matrix V. Note that from our model in Equation (6.1), we can write V as a sum of a sparse matrix S and a low-rank matrix $L = UW$, where the orthonormal columns of $U \in \mathbb{R}^{n \times d}$ span $\mathcal{S}$ (which is stationary), and $W \in \mathbb{R}^{d \times T}$ holds the weight vectors w_t as columns.

The global version of the l^1 cost function in Equation (6.3) follows:

$$\begin{aligned} \bar{F}(\mathcal{S}) = \sum_{t=1}^{T} \min_{w} \|U_{\Omega_t} w - v_{\Omega_t}\|_1 &= \min_{W \in \mathbb{R}^{d \times T}} \sum_{(i,j) \in \Omega} |(UW - V)_{ij}| \\ &= \min_{W \in \mathbb{R}^{d \times T}} \|\mathcal{P}_\Omega(UW - V)\|_1 \ . \end{aligned} \tag{6.5}$$

The right hand of Equation (6.5) can be rewritten as the equivalent constrained problem:

$$\begin{aligned} \min \quad & \|\mathcal{P}_\Omega(S)\|_1 \\ s.t. \quad & \mathcal{P}_\Omega(UW + S) = \mathcal{P}_\Omega(V) \\ & U \in \mathcal{G}(d, n) \end{aligned} \tag{6.6}$$

which is the same problem studied in [95], and the authors propose an efficient ADMM solver for this problem. Unlike the set-up of [27, 24], this problem is not convex; however it offers much more computationally efficient solutions. GRASTA differs from the algorithm of [95] in two major ways: it uses incremental gradient to minimize this cost function one column at a time for even greater efficiency, and it uses geodesics on the Grassmannian to compute the update of U.

Algorithm: Grassmannian Robust Adaptive Subspace Tracking

As we have said, GRASTA alternates between estimating the triple (s, w, y) and the subspace U. Here we discuss those two pieces of our algorithm. First we describe the update of (s, w, y) based on an estimate $\widehat{U}_t$ for the subspace variable. Then we describe the update of our subspace variable to $\widehat{U}_{t+1}$ based on the estimate of (s^*, w^*, y^*) resulting from the first step.

Update of the sparse vector, weight vector, and dual vector Given the current estimated subspace $\widehat{U}_t$, the partial observation v_{Ω_t}, and the observed entries' indices Ω_t, the optimal (s^*, w^*, y^*) of Equation (6.4) are the minimizers: $(s^*, w^*, y^*) = \arg\min_{s,w,y} \mathcal{L}(\widehat{U}_{\Omega_t}, s, w, y)$. These can be efficiently solved for by ADMM [16]. That is, s, w, and the dual vector y are updated in an alternating fashion:

$$\begin{cases} w^{k+1} = \arg\min_w \mathcal{L}(\widehat{U}_{\Omega_t}, s^k, w, y^k) \\ s^{k+1} = \arg\min_s \mathcal{L}(\widehat{U}_{\Omega_t}, s, w^{k+1}, y^k) \\ y^{k+1} = y^k + \rho(\widehat{U}_{\Omega_t} w^{k+1} + s^{k+1} - v_{\Omega_t}) \end{cases} \tag{6.7}$$

Specifically, these quantities are computed as follows. In this section we always assume that $U_{\Omega_t}^T U_{\Omega_t}$ is invertible, which is guaranteed if $|\Omega_t|$ is large enough [8]. We have:

$$w^{k+1} = \frac{1}{\rho}(\widehat{U}_{\Omega_t}^T \widehat{U}_{\Omega_t})^{-1} \widehat{U}_{\Omega_t}^T (\rho(v_{\Omega_t} - s^k) - y^k) \tag{6.8}$$

$$s^{k+1} = \mathsf{S}_{\frac{1}{1+\rho}}(v_{\Omega_t} - \widehat{U}_{\Omega_t} w^{k+1} - y^k) \tag{6.9}$$

$$y^{k+1} = y^k + \rho(\widehat{U}_{\Omega_t} w^{k+1} + s^{k+1} - v_{\Omega_t}) \tag{6.10}$$

where $\mathsf{S}_{\frac{1}{1+\rho}}$ is the elementwise soft thresholding operator [17]. We discuss this ADMM solver in detail as Algorithm 8 in Section 6.2.2.

Update of the Subspace Estimate Now to update our subspace estimate, we must take the gradient of 6.4 with respect to our subspace variable U. As in GROUSE, the gradient of this cost function is

rank one. We include the gradient from [52] for reference, first introducing three variables Γ, Γ_1, and Γ_2 to simplify the gradient expression:

$$\begin{aligned} \Gamma_1 &= y^* + \rho(U_{\Omega_t} w^* + s^* - v_{\Omega_t}) && (6.11) \\ \Gamma_2 &= U_{\Omega_t}^T \Gamma_1 && (6.12) \\ \Gamma &= \chi_{\Omega_t} \Gamma_1 - U\Gamma_2 && (6.13) \end{aligned}$$

and the gradient $\triangledown\mathcal{L}$ turns out to be [52]:

$$\triangledown\mathcal{L} = \Gamma {w^*}^T \tag{6.14}$$

Again as derived in [52], given (s^*, w^*, y^*), a gradient step of length η in the direction $-\triangledown\mathcal{L}$ is given by

$$U(\eta) = U + \left((cos(\eta\sigma) - 1) \frac{U w_t^*}{\|w_t^*\|} - sin(\eta\sigma) \frac{\Gamma}{\|\Gamma\|} \right) \frac{{w_t^*}^T}{\|w_t^*\|} \ . \tag{6.15}$$

Remarks Here we point out that at each subspace update step, our approach does not remove outliers explicitly. In fact, we use the gradient of the augmented Lagrangian $\mathcal{L}(U)$ Equation (6.4) which exploits the dual vector y^* to leverage the outlier effect. That is the key to success. Even when the ADMM solver 6.7 can not identify the outliers due to our current estimated subspace being far away from the true subspace, with the help of the dual vector y^* the gradient of the augmented Lagrangian gives us the right direction at each step which leads us to the right subspace.

We also must point out that since we estimate (s^*, w^*, y^*) at each step using the ADMM solver, we can not recover the exact subspace with sufficient accuracy if we do not allocate enough iterations for the ADMM solver [16]. Fortunately, as it also emphasized in [16], only a few tens of iterations per subspace update step are sufficient to achieve a modest accuracy, which is often acceptable for practical

use. Extensive experiments in [52] show that our algorithm is fast and always produces acceptable results, even when the vectors are noisy and heavily corrupted by outliers.

Algorithm Précis The discussion of Section 6.2.2 can be summarized into the GRASTA algorithm as follows. For each time step t, when we observe an incomplete and corrupted data vector v_{Ω_t}, our algorithm will first estimate the optimal value (s^*, w^*, y^*) from our current estimated subspace U_t via the l^1 minimization ADMM solver 6.7; then compute the gradient of the augmented Lagrangian loss function $\mathcal{L}$ by Equation (6.14); then choose a proper step-size; and finally do the rank one subspace update via Equation (6.15).

We state our main algorithm GRASTA (Grassmannian Robust Adaptive Subspace Tracking Algorithm) in Algorithm 7. GRASTA consists of two important sub-procedures: the ADMM solver of the least absolute derivations problem, and the computation of the adaptive step-size.

Algorithm 7 Grassmannian Robust Adaptive Subspace Tracking

[52] **Require**: An $n \times d$ orthogonal matrix U_0. A sequence of corrupted vectors v_t, each vector observed in entries $\Omega_t \subset \{1, \ldots, n\}$. A structure OPTS1 that holds parameters for ADMM. A structure OPTS2 that holds parameters for the adaptive step size computation.
Return: The estimated subspace U_t at time t.

1: **for** $t = 0, \ldots, T$ **do**
2: Extract U_{Ω_t} from U_t: $U_{\Omega_t} = \chi_{\Omega_t}^T U_t$
3: Estimate the sparse residual s_t^*, weight vector w_t^*, and dual vector y_t^* from the observed entries Ω_t via Algorithm 8 using OPTS1:
$$(s_t^*, w_t^*, y_t^*) = \arg\min_{w,s,y} \mathcal{L}(U_{\Omega_t}, w, s, y)$$
4: Compute the gradient of the augmented Lagrangian $\mathcal{L}$, $\nabla\mathcal{L}$ as follows:
$$\Gamma_1 = y_t^* + \rho(U_{\Omega_t} w_t^* + s_t^* - v_{\Omega_t}), \quad \Gamma_2 = U_{\Omega_t}^T \Gamma_1, \quad \Gamma = \chi_{\Omega_t}\Gamma_1 - U\Gamma_2$$
$$\nabla\mathcal{L} = \Gamma {w_t^*}^T$$
5: Compute step-size η_t according to desired update rule.
6: Update subspace: $U_{t+1} = U_t + ((cos(\eta_t\sigma) - 1)U_t \frac{w_t^*}{\|w_t^*\|} - sin(\eta_t\sigma)\frac{\Gamma}{\|\Gamma\|})\frac{{w_t^*}^T}{\|w_t^*\|}$
where $\sigma = \|\Gamma\|\|w_t^*\|$
7: **end for**

Unlike GROUSE, which has a closed form solution for computing the gradient, GRASTA estimates (s_t^*, w_t^*, y_t^*) by the ADMM iterated Algorithm 8. Certainly we would have a potential performance bottleneck if Algorithm 8 takes too much time at each subspace update step. However, we see empirically

that only a few tens of iterations in Algorithm 8 at each step allows GRASTA to track the subspace to an acceptable accuracy. In our video experiments with Algorithm 8, we always set the maximum iteration K around 20 to balance the trade-off between the subspace tracking accuracy and computational performance. We make a slight modification to the original ADMM sovler presented in [16]: in addition to returning w^* we also return the sparse vector s^* and the dual vector y^* for the further computation of the gradient $\nabla\mathcal{L}$. It is easy to verify that in the worst case the ADMM solver needs at most $O(|\Omega|d^3 + Kd|\Omega|)$ flops.

Algorithm 8 ADMM Solver for Least Absolute Deviations [16, 52]

Require: An $|\Omega_t| \times d$ orthogonal matrix U_{Ω_t}, a corrupted observed vector $v_{\Omega_t} \in \mathbb{R}^{|\Omega_t|}$, and a structure OPTS which holds four parameters for ADMM: ADMM step-size constant ρ, the absolute tolerance ϵ^{abs}, the relative tolerance ϵ^{rel}, and ADMM maximum iteration K.

Return: sparse residual $s^* \in \mathbb{R}^{|\Omega_t|}$; weight vector $w^* \in \mathbb{R}^d$; dual vector $y^* \in \mathbb{R}^{|\Omega_t|}$.

Initialize s,w,y: $s^1 = s^0$, $w^1 = w^0$, $y^1 = y^0$
(either to zero or to the final value from the last subspace update of the same data vector for a warm start.)
Cache $P = (U_{\Omega_t}^T U_{\Omega_t})^{-1} U_{\Omega_t}^T$
for $k = 1 \to K$ **do**
  Update weight vector: $w^{k+1} = \frac{1}{\rho} P(\rho(v_{\Omega_t} - s^k) - y^k)$
  Update sparse residual: $s^{k+1} = \mathsf{S}_{\frac{1}{\rho+1}}(v_{\Omega_t} - U_{\Omega_t} w^{k+1} - y^k)$
  Update dual vector: $y^{k+1} = y^k + \rho(U_{\Omega_t} w^{k+1} + s^{k+1} - v_{\Omega_t})$
  Calculate primal and dual residuals: $r^{pri} = \|U_{\Omega_t} w^{k+1} + s^{k+1} - v_{\Omega_t}\|$, $r^{dual} = \|\rho U_{\Omega_t}^T (s^{k+1} - s^k)\|$
  Update stopping criteria: $\epsilon^{pri} = \sqrt{|\Omega_t|}\epsilon^{abs} + \epsilon^{rel} \max\{\|U_{\Omega_t} w^{k+1}\|, \|s^{k+1}\|, \|v_{\Omega_t}\|\}$,
  $\epsilon^{dual} = \sqrt{d}\epsilon^{abs} + \epsilon^{rel} \|\rho U_{\Omega_t}^T y^{k+1}\|$
  if $r^{pri} \leq \epsilon^{pri}$ & $r^{dual} \leq \epsilon^{dual}$ **then**
    Converge and break the loop.
  end if
end for
$s^* = s^{k+1}$, $w^* = w^{k+1}$, $y^* = y^{k+1}$

Chapter 7

Column Selection with Missing Data

The work in this chapter is joint with Waheed Bajwa [5].

The problem of selecting k representative columns from a low-rank $m \times n$ matrix $\boldsymbol{Y}$, or *column subset selection* (CSS), arises frequently in data applications involving large data sets. Consider, for example, the case of an internet traffic measurement system in which a data matrix is constructed by collecting router traffic loads for various traffic types at different times into columns. In this application, CSS is akin to identifying representative routers that can lead to quick tracking down of network anomalies. Similarly, consider the case of a movie recommendation system in which a data matrix is constructed by collecting movie ratings of users into columns. In this application, CSS leads to identification of users who can reliably predict the ratings of newly-released movies. There are numerous other uses of the CSS problem in data applications and we refer the reader to some of the references within [15] for further motivation.

The CSS problem has been well-studied in the literature. The main algorithm that is put forth to solve this problem is *rank-revealing QR* (RRQR) decomposition [26, 49]. In particular, it has been shown that RRQR and its variants solve the CSS problem in a near-optimal fashion; see, e.g., [15, 49] and Table 1 within [15]. Unfortunately, it is nearly impossible in many data applications involving massive data sets to have access to the complete data. In the case of the internet traffic measurement system, for example, it is quite common for the routers to lose traffic data at various points in time. Similarly, in the case of the movie recommendation system, no user can be expected to rate every movie in the database. Existing formulations of the CSS problem in the literature seem incapable of

handling the case of missing data in an intelligible fashion.

In this chapter, we take a unified approach that naturally leads to algorithms for CSS with or without having access to full data. The problem of CSS is in some sense a detection problem, rather than an estimation problem, and therefore we expect it to remain well-posed even when a larger fraction of data is missing. Our main contribution in this regard is that we adapt the original optimization-based formulation of the CSS problem to accommodate situations when data are missing. The key distinguishing characteristic of our formulation is that it operates only on the matrix entries that have been observed, which is justified by the insights of Chapter 3. In addition, the work presented in here also opens the door to low-complexity CSS algorithms in terms of both computation and memory requirements for the case when a large fraction of data are missing.

7.1 Problem Formulation

Consider an $m \times n$ matrix $\boldsymbol{Y}$ whose n columns represent n sensors or measurement points and whose m rows represent measurement times. In this paper, we consider that $\boldsymbol{Y}$ is exactly low-rank and we are interested in choosing a subset of the columns of $\boldsymbol{Y}$ to represent the entire matrix. If we know the number of columns we would like to select, k, we can state CSS formally by saying that we want to find a matrix $\boldsymbol{X}$ such that $\boldsymbol{Y} = \boldsymbol{Y}\boldsymbol{X}$ while only k or fewer rows of $\boldsymbol{X}$ are non-zero. This problem is combinatorial; we may need to check all size-k subsets to find a suitable set. However, there are greedy algorithms that aim to solve this problem, and the one we use in this paper is *Block Orthogonal Matching Pursuit*, or BOMP. Block OMP can refer to an algorithm which solves a problem set up where dictionary elements are blocked [39]; in our case we have individual dictionary elements (the columns of our matrix) but require that the support on the dictionary is the same for every other column [100].

Rank-Revealing QR (RRQR) [26, 49] decompositions have been studied extensively in the literature for CSS [15]. These algorithms aim to find $\boldsymbol{Q}, \boldsymbol{R}$ and a permutation matrix $\boldsymbol{\Pi}$ such that $\boldsymbol{Y}\boldsymbol{\Pi} = \boldsymbol{Q}\boldsymbol{R}$. The permutation matrix essentially permutes the most well-conditioned columns to

the front of the matrix; given k, the first k are the selected columns when RRQR is used for column subset selection.

An adaptation of RRQR for matrices with missing data has not been developed; it is not clear to the authors whether a non-heuristic algorithm could be developed. A naïve approach would be to simply fill the matrix with zeros where data are missing. However, zero-filling a column vector changes the subspace of that vector 3.4.1 and can cause problems in selection, as we show in Section 7.3.

7.1.1 Group Lasso Formulation of Missing Data CSS

Often in applications data are missing and the matrix $\boldsymbol{Y}$ is incomplete. We can manipulate the problem given above for the situation when not all the data are observed. First we note that $\exists \boldsymbol{X}$ such that $\boldsymbol{Y} = \boldsymbol{Y}\boldsymbol{X}$ iff $\|\boldsymbol{Y} - \boldsymbol{Y}\boldsymbol{X}\|_F^2 = 0$. We now will manipulate this norm for the case where there are missing entries in $\boldsymbol{Y}$. This derivation first appeared in [5].

Let $\mathcal{C}_j = \{i : \text{The } i^{th} \text{ entry of column } j \text{ of } \boldsymbol{Y} \text{ is observed}\}$ and $\mathcal{R}_j = \{i : \text{The } i^{th} \text{ entry of row } j \text{ of } \boldsymbol{Y} \text{ is observed}\}$. Also let $\boldsymbol{Y}_{ij}$ represent the element in the i^{th} row and j^{th} column of $\boldsymbol{Y}$. Then we can look at $\|\boldsymbol{Y} - \boldsymbol{Y}\boldsymbol{X}\|_F^2$ considering only elements of $\boldsymbol{Y}$ that have been observed:

$$\sum_{j=1}^{n} \sum_{k \in \mathcal{C}_j} \left(\boldsymbol{Y}_{kj} - \sum_{i \in \mathcal{R}_k} \boldsymbol{X}_{ij} \boldsymbol{Y}_{ki} \right)^2 . \tag{7.1}$$

We again note that this norm is zero iff $\exists \boldsymbol{X}$ such that $\boldsymbol{Y} = \boldsymbol{Y}\boldsymbol{X}$ only on the observed entries of $\boldsymbol{Y}$. Define the zero-filled version of $\boldsymbol{Y}$, and indicator function $\boldsymbol{w}$ which indicates observed entries in $\boldsymbol{Y}$:

$$\widetilde{\boldsymbol{Y}}_{ij} = \begin{cases} \boldsymbol{Y}_{ij} & \text{if } \boldsymbol{Y}_{ij} \text{ is observed;} \\ 0 & \text{if } \boldsymbol{Y}_{ij} \text{ is unobserved.} \end{cases} \qquad \boldsymbol{w}_{ij} = \begin{cases} 1 & \text{if } \boldsymbol{Y}_{ij} \text{ is observed;} \\ 0 & \text{if } \boldsymbol{Y}_{ij} \text{ is unobserved.} \end{cases}$$

Also denote the j^{th} column vector with subscript $*j$ and the Hadamard product with $\circ$. Using these we can rewrite (7.1) as

$$\sum_{j=1}^{n}\sum_{k=1}^{n}\boldsymbol{w}_{kj}\left(\tilde{\boldsymbol{Y}}_{kj}-\sum_{i=1}^{n}\boldsymbol{X}_{ij}\tilde{\boldsymbol{Y}}_{ki}\right)^{2}=\sum_{j=1}^{n}\|\boldsymbol{w}_{*j}\circ\tilde{\boldsymbol{Y}}_{*j}-\boldsymbol{w}_{*j}\circ\left(\tilde{\boldsymbol{Y}}\boldsymbol{X}_{*j}\right)\|_2^2.$$

We can manipulate this into matrix-vector form. Let $\boldsymbol{W}_i$ be the diagonal matrix with column $\boldsymbol{w}_{*i}$ on the diagonal.

$$\sum_{j=1}^{n}\|\boldsymbol{w}_{*j}\circ\tilde{\boldsymbol{Y}}_{*j}-\boldsymbol{w}_{*j}\circ\left(\tilde{\boldsymbol{Y}}\boldsymbol{X}_{*j}\right)\|_2^2=\left\|\tilde{\boldsymbol{Y}}-\left[\boldsymbol{W}_1\tilde{\boldsymbol{Y}}\boldsymbol{X}_{*1},\boldsymbol{W}_2\tilde{\boldsymbol{Y}}\boldsymbol{X}_{*2},\ldots,\boldsymbol{W}_n\tilde{\boldsymbol{Y}}\boldsymbol{X}_{*n}\right]\right\|_F^2. \quad (7.2)$$

Now we will use $\boldsymbol{x}=\mathsf{vec}(\boldsymbol{X})$ for the operation of stacking the columns of $\boldsymbol{X}$into a single column vector; $\mathsf{vec}^{-1}(\boldsymbol{x})$ will be defined to undo this operation. Apply vec to the term inside the norm of (7.2). We first have $\mathsf{vec}(\tilde{\boldsymbol{Y}})=\tilde{\boldsymbol{y}}$. Let $\boldsymbol{W}_i$ be the diagonal matrix with column $\boldsymbol{w}_{*i}$ on the diagonal. Define the block-diagonal matrix

$$\boldsymbol{A}=\mathsf{diag}\left(\boldsymbol{W}_1\widetilde{\boldsymbol{Y}},...,\boldsymbol{W}_n\widetilde{\boldsymbol{Y}}\right). \quad (7.3)$$

Thus (7.2) becomes $\|\tilde{\boldsymbol{y}}-\boldsymbol{A}\boldsymbol{x}\|_2^2$. This is 0 iff $\exists\boldsymbol{X}$ such that $\tilde{\boldsymbol{y}}=\boldsymbol{A}\boldsymbol{x}$. The problem becomes:

Find $\boldsymbol{X}$ such that $\mathsf{vec}(\widetilde{\boldsymbol{Y}})=\boldsymbol{A}\mathsf{vec}(\boldsymbol{X})$, requiring that $\boldsymbol{X}$ has exactly k non-zero rows. (7.4)

7.2 Algorithm: Block OMP for Missing Data CSS

As is the full-data formulation, this problem is combinatorial, but many algorithms can be applied to solve it efficiently. One possible choice for such an algorithm is Block OMP for column subset selection with missing data, which would operate as shown in Algorithm 9. In the case where noise is added to the low-rank matrix $\boldsymbol{Y}$, the Group Lasso [111] can be used to formulate the problem as $\min_{\boldsymbol{x}}\|\widetilde{\boldsymbol{y}}-\boldsymbol{A}\boldsymbol{x}\|_2^2+\lambda\sum_{i=1}^{n}\|x_i\|_2$ where x_i is the i^{th} row of $\boldsymbol{X}$. The parameter λ can then be chosen in such a way to enforce that k rows of $\boldsymbol{X}$ are non-zero.

Algorithm 9 Block OMP for Column Subset Selection with missing data

Require: Samples from an $m \times n$ matrix $\boldsymbol{Y}$ and sample locations, from which we construct $\boldsymbol{A}$ as in (7.3) and $\mathsf{vec}(\widetilde{\boldsymbol{Y}})$. An integer k, the number of columns to be selected.

1: **Initialize:** $\widetilde{\boldsymbol{y}}_0 \leftarrow \mathsf{vec}(\widetilde{\boldsymbol{Y}})$, $\mathcal{I} \leftarrow \{\}$
2: **for** $t = 1, \ldots, k$ **do**
3: **Back-project and devectorize:** $d = \boldsymbol{A}^T \mathsf{vec}(\widetilde{\boldsymbol{y}}_0)$ and $D = \mathsf{vec}^{-1}(d)$.
4: **Choose max row norm:** $i_t = \arg\min_i \|D_i\|_2$ where D_i refers to the i^{th} row of D.
5: **Add index to selected set:** $\mathcal{I} = \mathcal{I} \cup \{i_t\}$.
6: **Fit to the current index set:** $\theta = \boldsymbol{A}_{\mathcal{I}}^{\dagger} \mathsf{vec}(\widetilde{\boldsymbol{Y}})$ where † denotes the pseudo-inverse.
7: **Update**: $\widetilde{\boldsymbol{y}}_0 = \mathsf{vec}(\widetilde{\boldsymbol{Y}}) - \boldsymbol{A}_{\mathcal{I}} \theta$
8: **end for**
9: **return** $\mathcal{I}$

Our new formulation of the CSS problem in (7.4) operates only on the matrix entries that have been observed and automatically reduces to the traditional problem formulation for the case when no data are missing. In this section, we demonstrate the significance of this formulation by numerically comparing the performance of the BOMP algorithm proposed for solving (7.4) and the RRQR factorization run naïvely on a zero-filled data matrix. For the sake of this exposition, we implement the RRQR factorization by making use of the implementataion of QR factorization provided in Matlab, which returns an ordering of matrix columns such that all the diagonal entries of $\boldsymbol{R}$ are decreasing.

7.3 Empirical Analysis

The numerical experiments reported in here correspond to column selection on a low-rank matrix $\boldsymbol{Y}$ with $m = 150$, $n = 200$, and $k = 4$. In order to built the matrix $\boldsymbol{Y}$, we first generate four standard Gaussian column vectors and then orthonormalize them. Next, we generate four sets of columns from each of these generating vectors such that each column in a set is a random scaling of the corresponding generating vector. The matrix $\boldsymbol{Y}$ then corresponds to collecting these four sets of columns into a matrix followed by a random permutation of the locations of these columns. We define success as picking one column from each of the four sets. The first set of numerical experiments that we carried out corresponded to the case of complete data (not shown in here). In this case, both BOMP and RRQR

picked exactly the same columns in every run: the largest-norm column from each set. This remained true over numerous runs regardless of the cardinality of the sets of columns.

Next, we carried out numerical experiments corresponding to the case of missing data. For this purpose, we randomly erased 10% to 90% of the mn entries of $\boldsymbol{Y}$ and also varied the cardinality of the column set sizes. We varied the first set from $0.05n$ to $0.75n$, and then we split the remaining columns evenly among the three remaining sets. The results of these experiments, shown in Fig. 30, demonstrate that our problem formulation together with BOMP outperforms RRQR run on a zero-filled data matrix, especially for the case when 30% to 60% of the data are missing, as long as no more than $\sim 70\%$ of the columns are explained by a single column in the matrix.

Two remarks are now in order concerning the insights gained from the numerical experiments and our proposed approach to CSS in the case of missing data. First, note that it is reasonable to expect that for the case when a single column describes a very large fraction of other columns along with missing data, BOMP would perform worse than RRQR with zero-filled matrix. This is because of the greedy nature of Step 4 in the BOMP algorithm that picks columns that best describe the "residual" energy in the remaining columns. Missing data would always leave some residual energy in the columns, which can accumulate to overshadow the energy in other sets if one single set gets too large. Second, the fact that RRQR with a zero-filled matrix significantly underperforms our problem formulation together with BOMP for the case of 30% to 60% missing data and comparable cardinality of column sets is completely in line [8], which shows that it is important to work only with observed data for subspace detection with missing data, since zero-filling the data in that case leads to false detection of energy outside the true subspace.

7.3.1 Discussion

We conclude this discussion by pointing out that it is indeed possible to envision alternative, sophisticated approaches to the problem of CSS in the absence of complete data. One immediate choice in

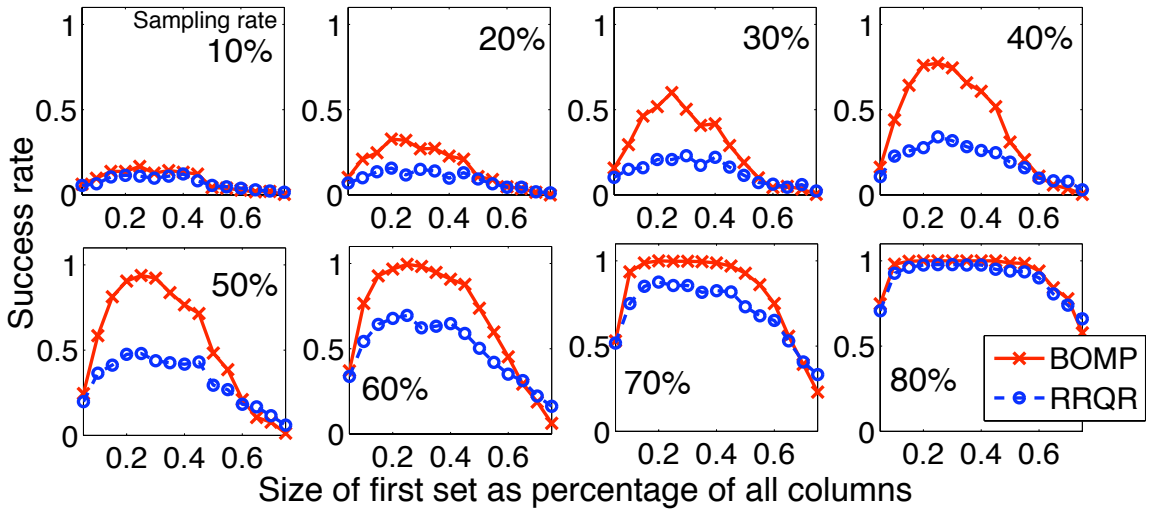


Figure 30: This simulation was done with $n = 200$, $m = 150$, rank of $\boldsymbol{Y} = 4$. Success is defined as choosing exactly one column from each of the four sets.

this regard could be a two-stage CSS procedure that involves preprocessing of the available data to impute the missing data followed by the RRQR factorization. However, this adds an extra layer of processing that increases the computational complexity of the problem. On the other hand, we have from Figure 30 that perfect CSS can be carried out in the absence of complete data *without* having to impute the missing data. In addition, it is also easy to see that one-stage CSS procedures could in fact take advantage of the missing data to reduce their computational complexity and memory requirements, whereas a two-stage CSS procedure which started with full-matrix imputation would be incapable of doing that. In this regard, the BOMP algorithm proposed in here for solving (7.4) validates the idea of CSS without missing data imputation. In the future, we plan on making use of our formulation in (7.4) to devise fast algorithms that are not only uniformly better than RRQR with a zero-filled matrix, but also take advantage of the missing data to reduce overall computational complexity and memory requirements. It is not hard to imagine that this could eventually lead us to situations where one would intentionally throw away data to reduce complexity and storage without sacrificing performance.

Chapter 8

Future Directions and Conclusions

A great many very interesting questions have arisen from this thesis work. The subspace model is arguably one of the most powerful fundamental models of signal processing. However, there are many other very useful models that are not addressed here and are of great interest: autoregressive models and dynamical models, to name just two. We hope that the work here may inspire future work in estimating other models with missing data; for example the unions of subspaces model can be used to approximate non-linear manifold models.

8.1 More General Modeling

In some applications, linear subspace models are not enough to capture the richness of correlations in data. A very exciting popular research direction is to explore the generalization to *manifolds*: a topological space which is locally Euclidean. A smooth manifold is one that looks linear for a small neighborhood around any given point. If data are sampled densely enough from a smooth manifold, a point and its closest neighbors will lie on a "locally linear patch of the manifold" [88]. Often smooth manifolds can be approximated to a desired approximation error by a union of linear subspaces. Very little (if any) work has been done on estimating manifolds when data are missing.

Another generalization that offers interesting problems is that of permutation spaces. If measurements come in the form of rankings as opposed to real numbers, we can consider problems of inference over these rankings as functions on permutation spaces.

8.2 Different Models for Missingness

As we mentioned in the introduction Section 1.1.1, this thesis made a very specific assumption that data are missing uniformly at random. It is of great interest to find a theoretical formulation under which one can analyze more structured missing data.

Social Sciences In the social sciences, analysis of surveys has always been an area where data go missing; some survey questions go unanswered by the survey takers. Often particular questions do not get answered, or sensitive questions may not get answered depending on the true response. Related to this, in [66], the authors define three types of missingness which provide food for thought.

Missing Completely At Random: This describes data which are missing independent of the *all* actual data values, both observed and missing. However, the missingness itself does not have to have a uniformly random or even a random pattern.

Missing At Random: This describes data which are missing independent of the missing data values themselves, but it may be dependent on the observed data.

Not Missing At Random: This describes data which are missing dependent on the missing data values themselves.

Hardware faults In sensor networks, data are missing or corrupted depending on the value of the sensor itself– for example if the true temperature is out of range of the sensor, that measurement will be clipped. When a measurement collection point fails, all the measurements are lost. An excellent taxonomy of sensor network failures can be found in [77].

8.3 Conclusion

In this thesis, we examined many classical statistical signal processing and linear algebra problems in subspace estimation from the modern perspective of big and messy datasets. Using tools from

probability theory, random matrix theory, statistical signal processing, adaptive filtering, and linear algebra, we were able to thoroughly explore the problem of subspace modeling when vector or matrix observations are missing.

Yet indeed, with this thesis we have only scratched the surface for what needs to be done to bring statistical modeling techniques up to date for the key applications of the 21st century. Big data applications inevitably have missing and corrupted data, and there are schools of modeling techniques that need to be altered to work provably well with missing data: for time-series models, autoregressive models, dynamical models (linear and non-linear), graphical models, just to name a few, little to nothing is known on how to estimate them when data are missing. That means there is a lot of work left to be done.

Bibliography

[1] Pankaj K. Agarwal and Nabil H. Mustafa. k-Means projective clustering. In *Proceedings of ACM SIGMOD-SIGACT-SIGART Symposium on Principles of database systems*, 2004.

[2] B.A. Ardekani, J. Kershaw, K. Kashikura, and I. Kanno. Activation detection in functional MRI using subspace modeling and maximum likelihood estimation. *IEEE Transactions on Medical Imaging*, 18(2), February 1999.

[3] Laura Balzano, Brian Eriksson, and Robert Nowak. High rank matrix completion and subspace clustering with missing data. In *Proceedings of the conference on Artificial Intelligence and Statistics (AIStats)*, 2012.

[4] Laura Balzano and Robert Nowak. Blind calibration of sensor networks. In *Proceedings of Information Processing in Sensor Networks*, April 2007.

[5] Laura Balzano, Robert Nowak, and Waheed Bajwa. Column subset selection with missing data. In *NIPS workshop on Low-Rank Methods for Large-Scale Machine Learning*, December 2010.

[6] Laura Balzano, Robert Nowak, and Benjamin Recht. Online identification and tracking of subspaces from highly incomplete information. In *Proceedings of the Allerton conference on Communication, Control, and Computing*, 2010.

[7] Laura Balzano, Robert Nowak, Arthur Szlam, and Benjamin Recht. k-Subspaces with missing data. In *IEEE Statistical Signal Processing Workshop (SSP)*, August 2012.

[8] Laura Balzano, Bejamin Recht, and Robert Nowak. High-dimensional matched subspace detection when data are missing. In *Proceedings of ISIT*, June 2010.

[9] Laura Balzano and Benjamin Recht, 2010. `http://sunbeam.ece.wisc.edu/grouse/`.

[10] R. Basri and D. Jacobs. Lambertian reflectance and linear subspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(2):218–233, February 2003.

[11] Dimitri P. Bertsekas. Incremental gradient, subgradient, and proximal methods for convex optimization: A survey. Technical Report LIDS-P-2848, MIT Lab for Information and Decision Systems, August 2010.

[12] Dimitri P. Bertsekas and John N. Tsitsiklis. Gradient convergence in gradient methods with errors. *SIAM Journal of Optimization*, 10(3):627–642, 2000.

[13] Christian H. Bischof. Incremental condition estimation. *SIAM Journal on Matrix Analysis and Applications*, 11(2):312–322, 1990.

[14] Christian H. Bischof and Gautam M. Shroff. On updating signal subspaces. *IEEE Transactions on Signal Processing*, 40(1), January 1992.

[15] C. Boutsidis, M. W. Mahoney, and P. Drineas. An improved approximation algorithm for the column subset selection problem. In *Proceedings of the Symposium on Discrete Algorithms*, pages 968–977, 2009.

[16] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–123, 2011.

[17] S.P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[18] Paul S. Bradley and Olvi L. Mangasarian. k-Plane clustering. *Journal of Global Optimization*, 16:23–32, 2000.

[19] M Brand. Fast low-rank modifications of the thin singular value decomposition. *Linear Algebra and its Applications*, 415(1):20–30, 2006.

[20] Samuel Burer and R. D. C. Monteiro. Local minima and convergence in low-rank semidefinite programming. *Mathematical Programming*, 103(3):427–444, 2005.

[21] Jian-Feng Cai, Emmanuel J. Candès, and Zuowei Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2008.

[22] E. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, December 2009.

[23] E. Candès and T. Tao. The power of convex relaxation: Near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56(5):2053 –2080, May 2010.

[24] E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the ACM*, 58(1):1–37, 2009.

[25] E. J. Candès, J. Romberg, and T. Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52(2):489–509, 2006.

[26] T. F. Chan. Rank revealing QR factorizations. *Linear Algebra and its Applications*, 88-89:67–82, April 1987.

[27] V. Chandrasekaran, S. Sanghavi, P.A. Parrilo, and A.S. Willsky. Rank-sparsity incoherence for matrix decomposition. *SIAM Journal on Optimization*, 21:572, 2011.

[28] Francoise Chatelin. *Eigenvalues of matrices*, pages 14–18. Wiley, Chichester, 1993.

[29] G. Chen and M. Maggioni. Multiscale Geometric and Spectral Analysis of Plane Arrangements. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Colorado Springs, CO, June 2011.

[30] Victor S. Chernyak. *Fundamentals of Multisite Radar Systems: Multistatic Radars and Multistatic Radar Systems*. Gordon and Breach Science Publishers, Amsterdam, The Netherlands, 1998.

[31] Pierre Comon and Gene Golub. Tracking a few extreme singular values and vectors in signal processing. *Proceedings of the IEEE*, 78(8), August 1990.

[32] Joao Paulo Costeira and Takeo Kanade. A multibody factorization method for independently moving objects. *International Journal of Computer Vision*, 29, 1998.

[33] Wei Dai and Olgica Milenkovic. SET: An algorithm for consistent matrix completion. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2010.

[34] J.-P. Delmas and J.-F. Cardoso. Performance analysis of an adaptive algorithm for tracking dominant subspaces. *Signal Processing, IEEE Transactions on*, 46(11):3045 –3057, nov 1998.

[35] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[36] Xinghao Ding, Lihan He, and L. Carin. Bayesian robust principal component analysis. *Image Processing, IEEE Transactions on*, 20(12):3419 –3430, dec. 2011.

[37] Alan Edelman, Tomas A. Arias, and Steven T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications*, 20(2):303–353, 1998.

[38] Alan Edelman and Steven T. Smith. On conjugate gradient-like methods for eigen-like problems. *BIT Numerical Mathematics*, 36:494–508, 1996. 10.1007/BF01731929.

[39] Y.C. Eldar, P. Kuppinger, and H. Bolcskei. Block-sparse signals: uncertainty relations and efficient recovery. *IEEE Transactions on Signal Processing*, 58:3042–3054, June 2010.

[40] Brian Eriksson, Paul Barford, Joel Sommers, and Robert Nowak. DomainImpute: Inferring Unseen Components in the Internet. In *Proceedings of IEEE INFOCOM Mini-Conference*, pages 171–175, Shanghai, China, April 2011.

[41] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[42] Neil Gershenfeld, Stephen Samouhos, and Bruce Nordman. Intelligent infrastructure for energy efficiency. *Science*, 327(5969):1086–1088, February 2010.

[43] Fulvio Gini, Alfonso Farina, and Maria Greco. Radar detection and preclassification based on multiple hypothesis. *Aerospace and Electronic Systems, IEEE Transactions on*, 40(3):1046 – 1059, july 2004.

[44] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 1996.

[45] Gene H. Golub and Hongyuan Zha. *Linear Algebra for Signal Processing*, volume 69, chapter "The canonical correlations of matrix pairs and their numerical computation", pages 29–49. Springer-Verlag, 1995.

[46] David Gross. Recovering low-rank matrices from few coefficients in any basis. *IEEE Transactions on Information Theory*, 57(3):1548 –1566, march 2011.

[47] A. Gruber and Y. Weiss. Multibody factorization with uncertainty and missing data using the em algorithm. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, 2004.

[48] M. Gu and S. Eisenstat. A stable and efficient algorithm for the rank-one modification of the symmetric eigenproblem. *SIAM Journal on Matrix Analysis and Applications*, 15(4):1266–1276, October 1994.

[49] Ming Gu and Stanley C. Eisenstat. Efficient algorithms for computing a strong rank-revealing qr factorization. *SIAM Journal on Scientific Computing*, 17:848–869, July 1996.

[50] John A. Gubner. *Probability and Random Processes for Electrical and Computer Engineers*. Cambridge University Press, Cambridge, UK, 2006.

[51] Jayant Gupchup, Randal Burns, Andreas Terzis, and Alex Szalay. Model-based event detection in wireless sensor networks. In *Proceedings of the Workshop on Data Sharing and Interoperability (DSI)*, 2007.

[52] Jun He, Laura Balzano, and John C.S. Lui. Online robust subspace tracking from partial information. Preprint available at `http://arxiv.org/pdf/1109.3827v2.`, 2011.

[53] Jun He, Laura Balzano, and Arthur Szlam. Incremental gradient on the grassmannian for online foreground and background separation in subsampled video. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2012.

[54] Peter J. Huber and Elvezio M. Ronchetti. *Robust Statistics*. Wiley, 2009.

[55] I. Jolliffe. *Principal Component Analysis*. Springer-Verlag, 1986.

[56] K. Kanatani. Motion segmentation by subspace separation and model selection. In *Proceedings of the Eighth IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 586–591, 2001.

[57] Raghunandan H. Keshavan, Andrea Montanari, and Sewoong Oh. Matrix completion from a few entries. *IEEE Transactions on Information Theory*, 56(6):2980–2998, June 2010.

[58] R.H. Keshavan, A. Montanari, and S. Oh. Matrix completion from noisy entries. *Journal of Machine Learning Research*, 11:2057–2078, July 2010.

[59] H. Krim and M. Viberg. Two decades of array signal processing research: the parametric approach. *Signal Processing Magazine, IEEE*, 13(4):67–94, July 1996.

[60] H. Kwon and N.M. Nasrabadi. Kernel matched subspace detectors for hyperspectral target detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(2), February 2006.

[61] Anukool Lakhina, Mark Crovella, and Christophe Diot. Diagnosing network-wide traffic anomalies. In *Proceedings of SIGCOMM*, 2004.

[62] Kirung Lee and Yoram Bresler. ADMiRA: Atomic decomposition for minimum rank approximation. *IEEE Transactions on Information Theory*, 56(9):4402 – 4416, Sep. 2010.

[63] Gilad Lerman and Teng Zhang. Robust Recovery of Multiple Subspaces by L_p Minimization. *Annals of Statistics*, 39(5):2686–2715, 2011.

[64] Yongmin Li. On incremental and robust subspace learning. *Pattern Recognition*, 37(7):1509–1518, 2004.

[65] Z. Lin, M. Chen, L. Wu, and Y. Ma. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. Technical Report UILU-ENG-09-2215, UIUC, October 2009. Arxiv preprint arXiv:1009.5055.

[66] Roderick J.A. Little and Donald B. Rubin. *Statistical Analysis with Missing Data*. Wiley-Interscience, 2002.

[67] Yue M. Lu and Minh N. Do. A theory for sampling signals from a union of subspaces. *IEEE Transactions on Signal Processing*, 56(6), June 2008.

[68] Shiqian Ma, Donald Goldfarb, and Lifeng Chen. Fixed point and Bregman iterative methods for matrix rank minimization. *Mathematical Programming*, 128:321–353, 2011. 10.1007/s10107-009-0306-5.

[69] Lester Mackey, Ameet Talwalkar, and Michael I. Jordan. Divide-and-conquer matrix factorization. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*, 2011.

[70] Jan R. Magnus and Heinz Neudecker. *Matrix Differential Calculus with Applications in Statistics and Econometrics*. John Wiley and Sons, 1999.

[71] Gonzalo Mateos and Georgios B. Giannakis. Sparsity control for robust principal component analysis. In *Proceedings of 44th Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, November 2010.

[72] G Mathew, Vellenki Reddy, and Soura Dasgupta. Adaptive estimation of eigensubspace. *IEEE Transactions on Signal Processing*, 43(2), February 1995.

[73] M.L. McCloud and L.L. Scharf. Interference estimation with applications to blind multiple-access communication over fading channels. *IEEE Transactions on Information Theory*, 46(3), May 2000.

[74] C. McDiarmid. On method of bounded differences. *Surveys in Combinatorics*, 141:148–188, 1989.

[75] Marc Moonen, Paul Van Dooren, and Joos Vandewalle. An SVD updating algorithm for subspace tracking. *SIAM Journal of Matrix Analysis and Applications*, 13:1015–1038, 1992.

[76] D. Needell and Joel Tropp. CoSaMP: Iterative signal recovery from incomplete and inaccurate samples. *Applied and Computational Harmonic Analysis*, 26(3):301–321, 2009.

[77] Kevin Ni, Nithya Ramanathan, Nabil Hajj Chehade, Laura Balzano, Sheela Nair, Sadaf Zahedi, Eddie Kohler, Greg Pottie, Mark Hansen, and Mani B. Srivastava. Sensor network data fault types. *ACM Transactions on Sensor Networks*, 5(3), 2009.

[78] N.M. Oliver, B. Rosario, and A.P. Pentland. A bayesian computer vision system for modeling human interactions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):831–843, 2000.

[79] Rafail Ostrovsky, Yuval Rabani, Leonard J. Schulman, and Chaitanya Swamy. The effectiveness of lloyd-type methods for the k-means problem. In *In 47th IEEE Symposium on the Foundations of Computer Science (FOCS)*, pages 165–176, 2006.

[80] Spiros Papadimitriou, Jimeng Sun, and Christos Faloutsos. Streaming pattern discovery in multiple time-series. In *Proceedings of VLDB Conference*, 2005.

[81] J.L. Paredes, Zhongmin Wang, G.R. Arce, and B.M. Sadler. Compressive matched subspace detection. In *Proceedings of the European Signal Processing Conference (EUSIPCO)*, Glasgow, Scotland, August 2009.

[82] Patrick O. Perry and Patrick J. Wolfe. Minimax rank estimation for subspace tracking. *IEEE Journal of Selected Topics in Signal Processing*, 4(3):504–513, June 2010.

[83] Chenlu Qiu and Namrata Vaswani. ReProCS: A missing link between recursive robust pca and recursive sparse recovery in large but correlated noise. Available at `http://arxiv.org/abs/1106.3286`, 2011.

[84] M. Rangaswamy, F.C. Lin, and K.R. Gerlach. Robust adaptive signal processing methods for heterogeneous radar clutter scenarios. *Signal Processing*, 84:1653–1665, September 2004.

[85] Benjamin Recht. A simpler approach to matrix completion. *Journal of Machine Learning Research*, 12:3413–3430, 2011.

[86] Benjamin Recht, Maryam Fazel, and Pablo Parrilo. Guaranteed minimum rank solutions of matrix equations via nuclear norm minimization. *SIAM Review*, 52(3):471–501, 2010.

[87] Benjamin Recht and Christopher Re. Parallel stochastic gradient algorithms for large-scale matrix completion. Submitted for publication. Available at `http://pages.cs.wisc.edu/~brecht/papers/11.Rec.Re.IPGM.pdf`, 2011.

[88] Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[89] R Roy and T Kailath. ESPRIT – estimation of signal parameters via rotational invariance techniques. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(7):984–995, July 1989.

[90] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Incremental svd-based algorithms for highly scalable recommender systems. In *Proceedings of the 5th International Conference on Computer and Information Technology (ICCIT)*, 2002.

[91] Ali H. Sayed. *Adaptive Filters*. John Wiley & Sons, Hoboken, New Jersey, 2008.

[92] L.L. Scharf. *Statistical Signal Processing*. Addison-Wesley, Reading, MA, 1991.

[93] L.L. Scharf and B. Friedlander. Matched subspace detectors. *IEEE Transactions on Signal Processing*, 42(8):2146–2157, August 1994.

[94] Ralph O. Schmidt. Multiple emitter location and signal parameter estimation. *IEEE Transactions on Antennas and Propagation*, AP-34(3):276–280, March 1986.

[95] Y. Shen, Z. Wen, and Y. Zhang. Augmented lagrangian alternating direction method for matrix separation based on low-rank factorization. *TR11-02, Rice University*, 2011.

[96] Steven T. Smith. *Geometric Optimization Methods for Adaptive Filtering*. PhD thesis, Harvard University, 1993.

[97] D.W.J. Stein, S.G. Beaven, L.E. Hoff, E.M. Winter, A.P. Schaum, and A.D. Stocker. Anomaly detection from hyperspectral imagery. *IEEE Signal Processing Magazine*, January 2002.

[98] G. W. Stewart. An updating algorithm for subspace tracking. *IEEE Transactions on Signal Processing*, 1992.

[99] Kim-Chuan Toh and Sangwoon Yun. An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems. *Pacific Journal of Optimization*, 6:615–640, 2010.

[100] J.A. Tropp, A.C. Gilbert, and M.J. Strauss. Algorithms for simultaneous sparse approximation, Part I: Greedy pursuit. *Signal Processing, special issue on Sparse approximations in signal and image processing*, 86:572–588, April 2006.

[101] René Vidal. A tutorial on subspace clustering. *IEEE Signal Processing Magazine*, 28(2):52–68, 2011.

[102] René Vidal, Yi Ma, and Shankar Sastry. Generalized Principal Component Analysis (GPCA). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27, December 2005.

[103] René Vidal, Stefano Soatto, Yi Ma, and Shankar Sastry. An algebraic geometric approach to the identification of a class of linear hybrid systems. In *Proceedings of the Conference on Decision and Control*, pages 167–172, 2003.

[104] René Vidal, Roberto Tron, and Richard Hartley. Multiframe motion segmentation with missing data using powerfactorization and GPCA. *International Journal of Computer Vision*, 79:85–105, 2008.

[105] Gregory S. Wagner and Thomas J. Owens. Signal detection using multi-channel seismic data. *Bulletin of the Seismological Society of America*, 86(1A):221–231, February 1996.

[106] Lu Wang, Lei Wang, Ming Wen, Qing Zhuo, and Wenyuan Wang. Background subtraction

using incremental subspace learning. In *Proceedings of the International Conference on Image Processing (ICIP)*, 2007.

[107] Rachel Ward. Compressed sensing with cross validation. *IEEE Transactions on Information Theory*, 55:5773–5782, December 2009.

[108] Huan Xu, C. Caramanis, and S. Sanghavi. Robust pca via outlier pursuit. *Information Theory, IEEE Transactions on*, 58(5):3047 –3064, may 2012.

[109] Bin Yang. Projection approximation subspace tracking. *IEEE Transactions on Signal Processing*, 43(1), January 1995.

[110] Jar-Ferr Yang and Mostafa Kaveh. Adaptive eigensubspace algorithms for direction or frequency estimation and tracking. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36(2), February 1988.

[111] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B*, 68:49–67, 2007.

[112] Teng Zhang, Arthur Szlam, Yi Wang, and Gilad Lerman. Randomized hybrid linear modeling by local best fit flats. In *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition*, San Francisco, CA, June 2010.