Algorithmic advances in learning from large dimensional matrices and scientific data

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Shashanka Ubaru

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Yousef Saad

May, 2018

Acknowledgements

This thesis was accomplished with the help and support of many amazing people, who I would like to wholeheartedly thank.

Firstly, I have to thank my PhD advisor, *Prof. Yousef Saad*. Without his guidance, help and support, this thesis work would not have been possible. I have learned a lot from him, and his knowledge, dedication, and strive for excellence have inspired me greatly. I am deeply indebted to him for his teachings, support and directions, and for giving me the freedom to pursue my varied interests with him and other collaborators. I am very fortunate to have worked with and learned from such an extraordinary academic.

Secondly, I wish to thank *Prof. Arya Mazumdar*, who co-advised my Master's thesis and guided me in many of the projects which are part of this thesis. He helped me explore a new direction of research in applications of coding theory. His hard-work and dedication to research have been great motivations for me.

Next, I want to thank *Kristofer Bouchard*, who has mentored me for the past two years on many projects related to ML for science, which are also part of this thesis. I also should thank *Kesheng Wu*, who along with Kris, hosted and mentored me for two summers at Berkeley Lab. I have to also thank *Abd-Krim Seghouane*, who has guided me in many projects and hosted me at Melbourne last summer. All three of them have helped me and supported me immensely in accomplishing this thesis.

I am grateful to the members of my dissertation committee, *Prof. Daniel Boley, Prof. George Karypis*, and *Prof. Georgios Giannakis*, who were very kind to agree to be on the advisory board. I would like to especially acknowledge Prof. Boley here, who served as a member of my Masters' thesis, PhD qualifier, thesis proposal, and defense committees. He has always been very gracious with his invaluable advises to me.

I have had the great fortune of collaborating with a variety of other brilliant professors and researchers over the years. I want to thank my collaborators; senior professors: Prof. Alexander Barg, Prof. James Chellikowsky, Prof. Michael Mahoney, and inspiring new professors: David P. Woodruff, Aaron Sidford, Agnieszka Meidlar, David Shuman, and Sharmodeep Bhattacharyya. I also want to thank top research scientists: Jie Chen, Praneeth Netrapalli, Anastasios Kryllidis, Prabhat, Edoardo Di Napoli, and inspiring fellow student, Cameron Musco. I am extremely grateful to all of them for their time, help, and the wisdom they have shared with me. They all have been very inspiring and motivating to work with.

At the same time, I have been lucky to share my lab space with some really smart and kind friends and colleagues. I have to thank Mr. (soon to be Dr. and then Prof.)Vassilis Kalantzis, I am fortunate to have overlapped with him for five years, Yuanzhi Xi, Thanh Ngo, Ruipeng Li, Agnieszka Meidlar, Geoffery Dillon, Amokrane Mahdi, Zahra, Naufal Nifa, Tianshi, Zechen, and Qing-qing. I have learned a lot through the wonderful talks and discussions we had over the years.

I wish to thank all my friends and my roommates over the years at Minnesota. My close friend and roommate for four years - Vahan Huroyan, former roommates and friends - Shireesh, Chidanand and Manoj, Nicole (who helped me a lot during the initial phase of my PhD), and Adil Ali. Special thanks to my friends from India, who have made me feel at home in Minnesota, especially, Prathima, Bhaskar, Manas, Durgesh, Inchara, Ashish, Abhishek, Suraj, Vikram, Shaman, Kshiteesh, Sanjay, Amoghavarsha, Karthik, Bharti aunty, Anil uncle, many friends I met at Sai Baba mandir, and the Minnesota Havyaka community. All of you made my life at Minnesota extremely fun and memorable! I also want to thank my friends at Berkeley, Anirvan, Rohit, Doris, Jonathan, and Mathew. Huge thanks to Aaditya Ramdas and Nihar Shah for being such great inspirations to me.

Last, but certainly not the least, I would like to thank my family - my parents and my brother who have greatly supported me through this long arduous journey, all my cousins, aunts and uncles from the awesome Ubaru and Odiyoor families. Special thanks to *Abbe-ajji* for her love and her mango pickle, reason for my survival away from home!

Dedication

To my parents, *Savitha* and *Ravishankar*.

Abstract

This thesis is devoted to answering a range of questions in *machine learning* and *data analysis* related to large dimensional matrices and scientific data. Two key research objectives connect the different parts of the thesis: (a) development of fast, efficient, and scalable algorithms for machine learning which handle large matrices and high dimensional data; and (b) design of learning algorithms for scientific data applications. The work combines ideas from multiple, often non-traditional, fields leading to new algorithms, new theory, and new insights in different applications.

The first of the three parts of this thesis explores *numerical linear algebra* tools to develop efficient algorithms for machine learning with reduced computation cost and improved scalability. Here, we first develop inexpensive algorithms combining various ideas from linear algebra and approximation theory for matrix spectrum related problems such as numerical rank estimation, matrix function trace estimation including log-determinants, Schatten norms, and other spectral sums. We also propose a new method which simultaneously estimates the dimension of the dominant subspace of covariance matrices and obtains an approximation to the subspace. Next, we consider matrix approximation problems such as low rank approximation, column subset selection, and graph sparsification. We present a new approach based on multilevel coarsening to compute these approximations for large sparse matrices and graphs. Lastly, on the linear algebra front, we devise a novel algorithm based on rank shrinkage for the dictionary learning problem, learning a small set of dictionary columns which best represent the given data.

The second part of this thesis focuses on exploring novel non-traditional applications of *information theory* and *codes*, particularly in solving problems related to machine learning and high dimensional data analysis. Here, we first propose new matrix sketching methods using codes for obtaining low rank approximations of matrices and solving least squares regression problems. Next, we demonstrate that codewords from certain coding scheme perform exceptionally well for the group testing problem. Lastly, we present a novel machine learning application for coding theory, that of solving large scale multilabel classification problems. We propose a new algorithm for multilabel classification which is based on group testing and codes. The algorithm has a simple inexpensive prediction method, and the error correction capabilities of codes are exploited for the first time to correct prediction errors.

The third part of the thesis focuses on devising robust and stable learning algorithms, which yield results that are interpretable from specific scientific application viewpoint. We present *Union of Intersections* (UoI), a flexible, modular, and scalable framework for statistical-machine learning problems. We then adapt this framework to develop new algorithms for matrix decomposition problems such as nonnegative matrix factorization (NMF) and CUR decomposition. We apply these new methods to data from Neuroscience applications in order to obtain insights into the functionality of the brain. Finally, we consider the application of material informatics, learning from materials data. Here, we deploy regression techniques on materials data to predict physical properties of materials.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Many modern machine learning, data analysis and scientific computation applications involve operating on large dimensional matrices and datasets (sizes ranging from $10^4 - 10^6$). With ever increasing sizes come the necessity to develop scalable fast algorithms for handling such large dimensional data. The primary focus of this thesis is in developing algorithms for high dimensional data problems with practical applications and theoretical guarantees. For this purpose, we explore ideas from different, often non-traditional, combinations of fields leading to new algorithms, new theory, and new insights in different applications. In the first part of the thesis, we explore *numerical linear algebra* tools to develop methods for matrix related problems. The spectrum (eigenvalue/singular value distribution) of data matrices, its functions and sums reveal various properties of the data, and are often required to be computed in numerous applications. However, as the size of the data matrices increases, it becomes impractical to explicitly compute the complete spectrum of these large matrices. Hence, the development of inexpensive methods which approximately estimate these spectral quantities has become a primary focus of research in many fields.

Similarly, in many modern applications involving very large datasets (matrices), the data is typically handled via some form of matrix approximation or dimensionality reduction. Matrix approximations and factorizations such as principal component analysis (PCA), partial singular value decompositions (SVD), CUR decompositions, graph sparsifications, nonnegative matrix factorization (NMF), and dictionary learning are some of the popular tools used in many applications. Designing inexpensive methods for the computation of the matrix spectral quantities and various matrix approximations for large matrices is the focus of the first part of the thesis.

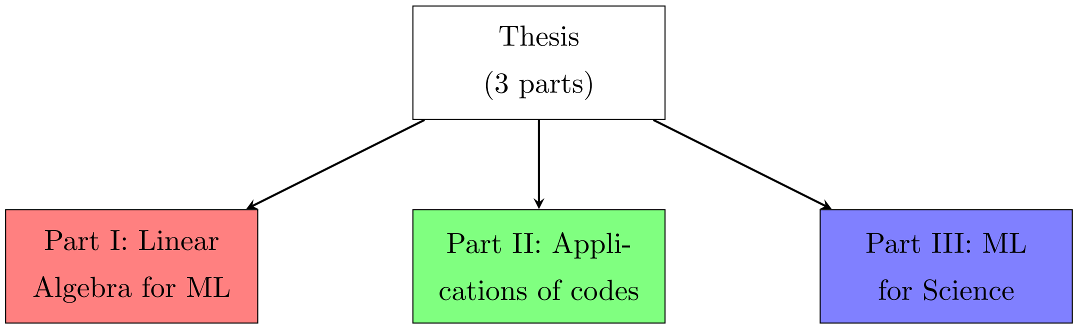
Error correcting codes (ECC) have been used successfully in communication systems for several decades. In recent years, a new line of research is emerging where information theory and codes are being explored for nontraditional applications such as distributed storage systems, compressed sensing, and scientific computing. The second part of this thesis explores novel applications of ECC, particularly in solving problems related to machine learning and high dimensional data analysis. We show how unique properties of codes such as k-wise independence of codewords, orthogonality of code matrices, low coherence, error correction and others, can be exploited to develop new methods to solve various problems including low rank approximation, least squares regression, group testing, and multilabel classification.

In many scientific fields, the development of new sensing and imaging technologies has resulted in the generation of large volumes of data. These large datasets bring with them opportunities of new discoveries and insights into the fundamentals of nature. In order to realize such opportunities, the development of novel machine learning and data analysis methods is necessary. Statistical-machine learning algorithms for scientific data should satisfy the bi-criteria of returning results that are simultaneously predictive and interpretable. However, these bi-criteria are often at odds, and methods that robustly (few assumptions on the data/noise) achieve both are lacking. In the third part of this thesis, we focus on designing new learning methods that robustly extract meaningful and interpretable information from the data which arrive from different areas of science.

1.1 Thesis Overview

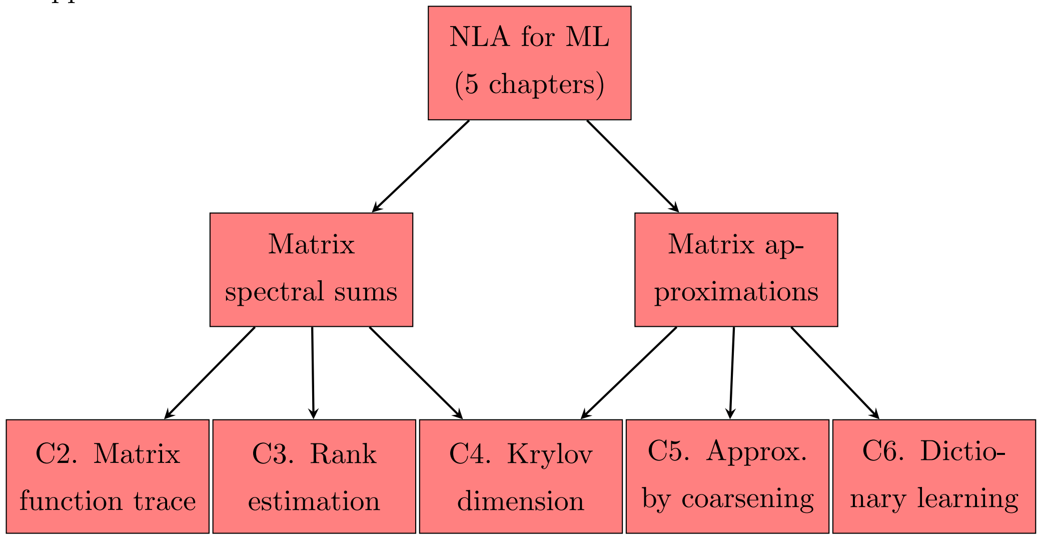
The underlining theme that connects the different parts of this thesis is: *development of algorithms for learning from large dimensional data.* The thesis comprises of three main parts.

1. Linear Algebra for Machine Learning (ML).
2. Applications of Coding Theory.
3. Machine Learning for Science.



The thesis work primarily has two key objectives. The first one is developing algorithms for handling large dimensional data matrices in machine learning that are computationally inexpensive and scalable. In order to develop these algorithms, we explore ideas and tools from two diverse, but well-studied fields, namely, numerical linear algebra (Part I) and coding theory (Part II). The second objective is designing learning algorithms that yield interpretable results in scientific applications (Part III).

Linear Algebra for machine learning - Powerful, efficient and inexpensive (in terms of computation time and memory requirements) methods can be developed using *numerical linear algebra* tools to solve various problems related to large dimensional data applications. In the first part of the thesis, we explore these tools to develop methods for matrix related problems that commonly arise in machine learning and other applications.



In most applications, the matrix related problems we encounter can be classified into two broad classes. The first class is related to the matrix spectrum, in particular, estimating spectral functions and their sums. These quantities reveal different properties of the given data matrix. The second class of matrix problems involves approximating a given matrix by smaller and/or sparser matrices in order to reduce computational and storage costs. In this thesis, we discuss the following matrix problems belonging to these two categories.

- **Matrix function trace estimation:** The problem of estimating the trace (sum of diagonal entries) of matrix functions such as log-determinant, Schatten norms, and spectral densities, appears often in different applications. However, explicit computation of such traces will be impractical for large matrices. This is the first problem considered in the thesis and a set of inexpensive algorithms are discussed for these trace estimations. In chapter 2, we demonstrate how a method called Stochastic Lanczos Quadrature can be employed for the fast computation of these traces. We establish theoretical results, which give multiplicative and additive error guarantees for the method. We also present few applications where the method can be useful. The results in this chapter were published in [1].

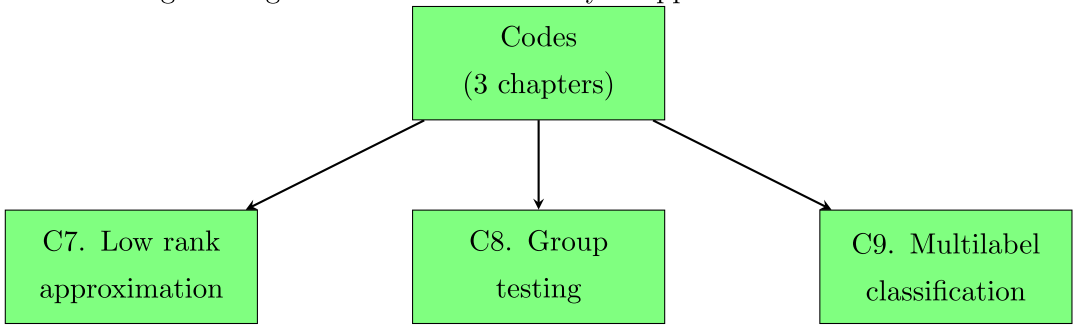
- **Numerical rank estimation:** In many of the data involving applications, it is often required to know the approximate rank of the data matrix at hand (approximate smaller dimension in which the information actually lies). In chapter 3, we consider the problem of numerical (approximate) rank estimation, and present a set of inexpensive methods, which require no matrix decomposition for the estimation of approximate matrix ranks. These methods were presented in [2, 3].

- **Krylov dimension estimation:** In many applications, it is often desired to estimate the numerical rank of the data, and then obtain an approximation for the principal subspace associated with the numerical rank, for example in principal component analysis (PCA), subspace tracking, etc. The Krylov subspace based methods are the most popular and effective methods used for computing these approximations. In chapter 4, we present a method which combines a novel criterion based rank estimator with the Krylov subspace methods to simultaneously estimating the numerical rank of covariance matrices and approximate the associated principal subspace. The work is discussed in [4]. This work addresses two problems (rank estimation and subspace approximation) simultaneously, belonging to the two classes (matrix spectrum and matrix approximation), respectively.

- **Matrix approximation via coarsening:** In modern applications involving very large datasets (matrices), the data is typically handled via some form of matrix approximation such as the partial SVD, column subset selection, graph sparsification and others. We propose a novel approach to compute these approximations for large sparse matrices and graphs in chapter 5. The approach is based on multilevel coarsening, a popular tool used in graph partitioning and parallel computing, and is presented in [5]. This method exploits the structures available in the data and yields superior approximations.

- **Dictionary learning:** In signal and image processing research, the dictionary learning problem (learning a set of dictionary columns from which the matrices are derived) has received a lot of attention in recent years. Chapter 6 explores new ideas for dictionary learning using linear algebra tools. One such algorithm we develop is to learn incoherent dictionaries (distinctive dictionary columns in terms of correlation) based on rank shrinkage, published in [6]. The proposed algorithm yields improved dictionary incoherence compared to other popular methods.

Applications of coding theory - Recently, error correcting codes (ECC) have been used in nontraditional applications such as distributed storage systems, compressed sensing, and scientific computing. In the second part of this thesis, we explores novel machine learning and high dimensional data analysis applications of codes.

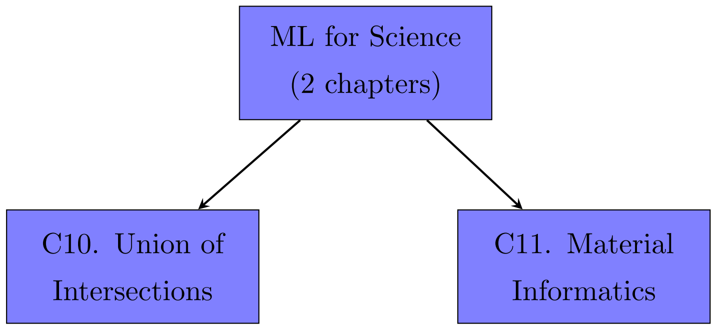


- **Low rank approximation:** In recent years, much attention has been devoted to randomized sampling and sketching methods for effectively approximating large matrices. In chapter 7, we demonstrate that certain error correcting codes (which are almost deterministic) mimic the randomness properties desired for sampling data matrices. Using this, we illustrate how codes can be used to obtain low rank approximations of matrices and also solve least squares regression problems. We first presented the idea in [7], and improved the theoretical results and expanded the scope of the work in [8].

- **Group testing:** The problem of group testing, detecting defective items from a large set of items by testing for defects in groups, has many applications in imaging, sensing, genetics, and recently in multilabel classification. We demonstrate that codewords from a popular coding scheme performs exceptionally well in group testing, and show that the proposed method outperforms traditional methods in chapter 8, first presented in [9].

- **Multilabel classification:** The coding theory application part of the thesis is concluded by presenting a new machine learning application for codes, that of solving large scale multilabel classification problems. Such classification problems are ubiquitous in various applications such as image, video and tweet annotation, web page categorization and others, and modern applications have a large number of classes. In chapter 9, we propose a new algorithm for multilabel classification which is based on group testing and codes. The algorithm has many promising advantages, including a simple prediction method that is very inexpensive, and it exploits the error correction capabilities of codes for the first time to correct prediction errors. We presented this multilabel classification approach in [10].

Machine learning for science - In order to explore the vast amount of complex data collected in different scientific fields, the development of novel machine learning and data analysis methods is necessary. The primary objective here is to obtain results that are interpretable from the scientific viewpoint (traditional methods fail to do so), and that are stable and robust to noise, since scientific data are typically noisy, where standard noise models fail. The last of the three parts of this thesis focuses on developing such learning methods to extract meaningful information from the data which arrive from different areas of science.



- **Union of Intersections:** In chapter 10, we discuss a new statistical framework named *Union of Intersections* (UoI) for interpretable machine learning proposed in [11]. UoI is a flexible, modular, and scalable framework for statistical-machine learning problems. We demonstrate the applicability of the framework by developing new algorithms for Nonnegative matrix factorization (NMF) ($UoI\text{-}NMF_{cluster}$) and CUR decomposition ($UoI\text{-}CUR$) based on UoI. Modern technologies such as Electrocorticography (ECoG) have resulted in the collection of large volumes of neurophysiological data. Extracting key features from such data will provide better insight into functioning of the brain and may result in new discoveries. We employ $UoI\text{-}NMF_{cluster}$ to summarize Neuroscience data with a set of few interpretable bases. We presented these results in [12]. We also use the $UoI\text{-}CUR$ algorithm to summarize genetics data.

- **Material Informatics:** In recent years, the use of machine learning techniques to explore materials data is gaining popularity. In the last chapter 11 of this thesis, we examine supervised learning techniques for the study of material behavior and for material discovery. We develop a machine learning (regression) technique for the prediction of formation enthalpies of new metal alloys using easily available material data, which we presented in [13]. The goal of this work is to help bypass time intensive calculations (some take days of computations), e.g., ab-initio calculations, used in material science.

1.2 Background

In this section, we briefly review the key background topics related to the work presented in the thesis.

1.2.1 Matrix function traces

The problem of estimating the trace of matrix functions appears frequently in applications of machine learning, signal processing, scientific computing, statistics, computational biology and computational physics [14, 15, 16, 17]. For a symmetric matrix $A \in \mathbb{R}^{n\times n}$ with an eigen-decomposition $A = U\Lambda U^T$, with $\Lambda = diag(\lambda_1, \dots, \lambda_n)$, where λ_i, $i = 1, \dots, n$ are the eigenvalues of A, the matrix function $f(A)$ is defined as $f(A) = Uf(\Lambda)U^T$, with $f(\Lambda) = diag(f(\lambda_1), \dots, f(\lambda_n))$ [18]. Then the trace estimation problems mentioned above can be formulated as follows: given a symmetric matrix $A \in \mathbb{R}^{n\times n}$, compute an approximation of the trace of the matrix function $f(A)$, i.e.,

$$\texttt{tr}(f(A)) = \sum_{i=1}^{n} f(\lambda_i), \tag{1.1}$$

where λ_i, $i = 1, \dots, n$ are the eigenvalues of A, and f is the desired function. A naive approach for estimating the trace of matrix functions is to compute this trace from the eigenvalues of the matrix, which will be expensive for large matrices. Popular trace estimation problems include computing the log-determinant ($f(t) = \log(t)$), the Schatten p-norms ($f(t) = t^{p/2}$), the Estrada index ($f(t) = e^t$) and the trace of matrix inverse ($f(t) = t^{-1}$). In chapter 2, we present a set of inexpensive methods to approximately estimate these traces.

1.2.2 Stochastic trace estimator

Hutchinson's unbiased estimator [19] uses only matrix-vector products to approximate the trace of a generic matrix D. The method estimates the trace $\texttt{tr}(D)$ by first generating random vectors $v_l, l = 1, \dots, \mathrm{n_v}$ with equally probable entries ± 1, and then computing the average over the samples of $v_l^\top D v_l$,

$$\mathtt{tr}(D) \approx \frac{1}{\mathrm{n_v}} \sum_{l=1}^{\mathrm{n_v}} v_l^\top D v_l. \tag{1.2}$$

It is known that any random vectors v_l with mean of entries equal to zero and unit 2-norm can also be used in the above computation [20]. This stochastic trace estimator can be employed in methods which compute the matrix function traces $\mathtt{tr}(f(A))$ discussed above.

The convergence analysis for the stochastic trace estimator was developed in [20], and improved in [21] for sample vectors with different probability distributions. We present the convergence rate results in chapter 2. We will employ the stochastic trace estimator to compute various matrix spectrum related quantities in chapters 2 and 3.

1.2.3 Matrix factorizations and approximations

Modern applications typically handle large datasets (matrices) via some form of matrix approximation and dimensionality reduction. Here, we review some of the popular matrix factorization and approximation methods that will be considered in this thesis.

SVD and low rank approximation: Low rank approximation and principal component analysis (PCA) are the most popular dimensionality reduction methods used in numerous data related applications [22, 23]. Consider a data matrix $A \in \mathbb{R}^{m\times n}$, with its singular Value decomposition (SVD) $A = U\Sigma V^T$, where U and V are orthogonal matrices contain the left and the right singular vectors as columns, respectively, and Σ is a diagonal matrix with nonzero singular values σ_i, $i = 1,\dots,n$ as diagonal entries. For a rank $k \ll n$, a rank-k approximation can be obtained as, $A_k = U_k\Sigma_k V_k^T$, where U_k and V_k are matrices containing the top k left and right singular vectors as columns, respectively.

It is well-known that $A_k = U_k\Sigma_k V_k^T$ is the best rank-k approximation of A with respect to both Frobenius and spectral norms. That is, we have from the Eckart-Young theorem [24], for $\xi \in \{2, F\}$,

$$\min_{\mathrm{rank}(X)\le k} \|A - X\|_\xi = \|A - A_k\|_\xi. \tag{1.3}$$

We also know that, PCA and SVD are closely related. This is because the left singular vectors U of A are the eigenvectors of AA^T and the right singular vectors V of A are the eigenvectors of A^TA. Hence, in many data related applications, PCA and partial SVD are used interchangeably.

CUR decomposition: Another popular dimensionality reduction method used in many applications is column subset selection (CSSP) [25]. If even a subset of the rows are selected, then the method is called CUR decomposition [26], where C and R are matrices with the selected columns and rows, respectively, and U is the mixing matrix, such that $A \approx CUR$. Given a large data matrix $A \in \mathbb{R}^{m \times n}$ whose columns we wish to select, suppose V_k is the matrix associated with the top k right singular vectors of A. Then, the leverage score of the ith column of A is given by

$$\ell_i = \frac{1}{k}\|V_k(i,:)\|_2^2,$$

the norm of the ith row of V_k. In leverage scores sampling, the columns of A are sampled using the probability distribution $p_i = \min\{1, \ell_i\}$. Most popular methods for CSSP involve the use of this leverage scores as the probability distribution for columns selection [27, 25, 26]. However, this method is expensive since one needs to compute the top k singular vectors, which is prohibitive for large matrices. Many other closely related methods have been proposed for randomized sampling and column subset selection of matrices [28].

Graph sparsification: Sparsification of large graphs has several computational (cost and space) advantages and has hence found many applications [29, 30, 31]. Given a large graph $G = (V, E)$ with n vertices, we wish to find a sparse approximation to this graph that preserves certain information of the original graph such as the spectral information [31, 32], structures like clusters within in the graph [29], etc. Let $B \in \mathbb{R}^{\binom{n}{2} \times n}$ be the vertex edge incidence matrix of the graph G, where eth row b_e of B for edge $e = (u, v)$ of the graph has a value $\sqrt{w_e}$ in columns u and v, where w_e is the weight of the edge, and zero elsewhere. The corresponding Laplacian of the graph is then given by $K = B^TB$.

The spectral sparsification problem involves computing a weighted subgraph $\tilde{G}$ of G

such that if $\tilde{K}$ is the Laplacian of $\tilde{G}$, then $x^T\tilde{K}x$ is close to x^TKx for any $x \in \mathbb{R}^n$. Many methods have been proposed for the spectral sparsification of graphs, see e.g., [31, 32, 28]. A popular approach is to perform row sampling of the matrix B using the leverage score sampling [32]. Considering the SVD of $B = U\Sigma V^T$, the leverage scores ℓ_i for a row b_i of B can be computed as $\ell_i = \|u_i\|_2^2 \leq 1$ using the rows of U. This leverage score is related to the effective resistance of edge i [31]. By sampling the rows of B according to their leverage scores it is possible to obtain a matrix $\tilde{B}$, such that $\tilde{K} = \tilde{B}^T\tilde{B}$ and $x^T\tilde{K}x$ is close to x^TKx for any $x \in \mathbb{R}^n$.

Nonnegative matrix factorization (NMF): Since its popularization, NMF [33] has been used in many applications for obtaining interpretable decompositions of data. Given a matrix $A \in \mathbb{R}_+^{m\times n}$ ($\mathbb{R}_+$ represents the positive orthant), where each row of A corresponds to a data point in $\mathbb{R}_+^n$, and a rank k, the problem of NMF is to compute the matrices $W \in \mathbb{R}_+^{m\times k}$ and $H \in \mathbb{R}_+^{k\times n}$, such that $A \approx WH$. This problem is generally posed as a non-convex optimization problem,

$$\min_{W\geq 0, H\geq 0} \|A - WH\|_F. \tag{1.4}$$

Here, the rows of the matrix H form the basis of the objects (say images), and the rows of W are the encoding of the basis in the matrix A. Since both W and H are nonnegative, NMF sometimes gives more interpretable parts based decompositions, with the intuitive notion of “combining parts to form a whole” [33].

Several algorithms to solve the NMF problem have been developed to achieve various objectives, such as more interpretable results, sparser solutions, unique solutions, etc. Sparse NMF [34, 35] and convolutional NMF [36] are two popular variants of NMF. It has been claimed that NMF implicitly yields a sparse representation of the data. However, in order to obtain explicit sparse NMF solutions, the following objective function is popularly used:

$$\min_{W\geq 0, H\geq 0} \frac{1}{2}\left[\|A - WH\|_F^2 + \lambda_1\|W\|_F^2 + \lambda_2\sum_{j=1}^{k}\|H(j,:)\|_p^2\right]$$

with $p = \{0, 1\}$ (promotes sparsity), λ_1 and λ_2 are regularization parameters.

Dictionary learning: In recent years, sparse signal approximations by means of redundant or overcomplete dictionaries have received much attention across various research areas, particularly in signal and image processing [37, 38]. Considering a set of signals $Y = (y_1, \ldots, y_N)$ with $y_i \in \mathbb{R}^n$ and a redundant (overcomplete) dictionary $D \in \mathbb{R}^{n\times k}$, $k > n$, the sparse signal approximation model assumes that the signals y_i can be represented as a sparse linear combination of the columns of D, which are also called atoms. So, we can express this model as

$$y_i \simeq Dx_i, \quad i = 1, \ldots, N, \tag{1.5}$$

where x_i's $\in \mathbb{R}^k$ are sparse vectors with a very small number of non-zero approximation coefficients $s = \|x_i\|_0 \ll n$. The particularity of a dictionary learning model is that we learn the dictionary D, as well as find the sparse linear multivariate model that best describes the set of signals Y, simultaneously. The parameters of this model are determined by solving the sparse approximation problem

$$\underset{D\in\mathcal{D}, X\in\mathcal{X}}{\arg\min} \|Y - DX\|_F^2, \tag{1.6}$$

where $\|.\|_F$ is the Frobenius norm, and $\mathcal{D}$ and $\mathcal{X}$ are the admissible sets for the dictionary and the approximation coefficient matrix $X = (x_1, \ldots, x_N)$, respectively. The set $\mathcal{D}$ is usually defined as the set of all dictionaries with unit column norm, i.e., $\mathcal{D} = \{D \in \mathbb{R}^{n\times k} : \|d_j\|_2 = 1, \forall j\}$, whereas $\mathcal{X}$ constrains the coefficient matrix to be sparse, i.e., the number of nonzero entries s in X is much smaller compared to the total number of entries n or $\mathcal{X} = \{X \in \mathbb{R}^{k\times N} : \|x_i\|_0 \leq s \ll n, \forall i\}$. Dictionary learning starts with the set of signals Y, and aims to find both the dictionary D and the approximation coefficients matrix X. The optimization problem (1.6) is not convex and has been shown to be an NP hard problem [39]. So, the methods for solving it can only hope to achieve an approximate solution.

1.2.4 Error Correcting Codes

In communication systems, data are transmitted from a source (transmitter) to a destination (receiver) through physical channels. These channels are usually noisy, causing errors in the data received. In order to facilitate detection and correction of these errors in the receiver, error correcting codes are used [40]. A block of information (data) symbols are encoded into a binary vector[1], also called a codeword. Error correcting coding methods check the correctness of the codeword received. The set of codewords corresponding to a set of data vectors (or symbols) that can possibly be transmitted is called the *code*. As per our definition a code $\mathcal{C}$ is a subset of the binary vector space of dimension ℓ, $\mathbb{F}_2^\ell$, where ℓ is an integer.

A code is said to be linear when adding two codewords of the code coordinate-wise using modulo-2 arithmetic results in a third codeword of the code. Usually a linear code $\mathcal{C}$ is represented by the tuple $[\ell, r]$, where ℓ represents the codeword length and $r = \log_2 |\mathcal{C}|$ is the number of information bits that can be encoded by the code. There are $\ell - r$ redundant bits in the codeword, which are sometimes called parity check bits, generated from messages using an appropriate rule. It is not necessary for a codeword to have the corresponding information bits as r of its coordinates, but the information must be uniquely recoverable from the codeword.

It is perhaps obvious that a linear code $\mathcal{C}$ is a linear subspace of dimension r in the vector space $\mathbb{F}_2^\ell$. The basis of $\mathcal{C}$ can be written as the rows of a matrix, which is known as the generator matrix of the code. The size of the generator matrix G is $r \times \ell$, and for any information vector $\boldsymbol{m} \in \mathbb{F}_2^r$, the corresponding codeword is found by the following linear map:

$$\boldsymbol{c} = \boldsymbol{m}G.$$

Note that all the arithmetic operations above are over the binary field $\mathbb{F}_2$.

To encode r bits, we must have 2^r unique codewords. Then, we may form a matrix of size $2^r \times \ell$ by stacking up all codewords that are formed by the generator matrix of a given linear coding scheme,

$$\underbrace{C}_{2^r \times \ell} = \underbrace{M}_{2^r \times r} \underbrace{G}_{r \times \ell}. \tag{1.7}$$

For a given tuple $[\ell, r]$, different error correcting coding schemes have different generator matrices and the resulting codes have different properties. For example, for any two integers t and q, a BCH code [41] has length $\ell = 2^q - 1$ and dimension $r = 2^q - 1 - tq$.

[1]Here, and in the rest of the text, we are considering only binary codes. In practice, codes over other alphabets are also quite common.

Any two codewords in this BCH code maintain a minimum (Hamming) distance of at least $2t + 1$ between them.

Code properties: The minimum pairwise distance between codewords is an important parameter of a code and is called just the *distance* of the code. As a linear code $\mathcal{C}$ is a subspace of a vector space, the null space $\mathcal{C}^{\perp}$ of the code is another well defined subspace. This is called the *dual* of the code. For example, the dual of the $[2^q - 1, 2^q - 1 - tq]$-BCH code is a code with length $2^q - 1$, dimension tq and minimum distance at least $2^{q-1} - (t-1)2^{q/2}$. The minimum distance of the dual code is called the *dual distance* of the code.

Depending on the coding schemes used, the codeword matrix C will have a variety of favorable properties, e.g., low coherence which is useful in compressed sensing [42]. Since the codewords need to be far apart, they show some properties of random vectors. We can define probability measures for codes generated from a given coding scheme. If $\mathcal{C} \subset \{0,1\}^{\ell}$ is an $\mathbb{F}_2$-linear code whose dual $\mathcal{C}^{\perp}$ has a minimum distance above k (dual distance $> k$), then the code matrix is an *orthogonal array* of strength k [43]. This means that, in such a code $\mathcal{C}$, for any k entries of a randomly and uniformly chosen codeword $\boldsymbol{c}$ say $c' = \{c_{i_1}, c_{i_2}, \ldots, c_{i_k}\}$ and for any k bit binary string α, we have

$$Pr[c' = \alpha] = 2^{-k}.$$

This is called the k-wise independence property of codes.

The codeword matrix C has 2^r codewords each of length ℓ (a $2^r \times \ell$ matrix), i.e., a set of 2^r vectors in $\{0,1\}^{\ell}$. Given a codeword $c \in \mathcal{C}$, let us map it to a vector $\phi \in \mathbb{R}^{\ell}$ by setting $1 \longrightarrow \frac{-1}{\sqrt{2^r}}$ and $0 \longrightarrow \frac{1}{\sqrt{2^r}}$. In this way, a binary code $\mathcal{C}$ gives rise to a code matrix $\Phi = (\phi_1^{\top}, \ldots, \phi_{2^r}^{\top})^{\top}$. Such a mapping is called binary phase-shift keying (BPSK) and appeared in the context of sparse recovery (e.g., p. 66 [42]).

Part I

Linear Algebra for Machine Learning

Chapter 2

Matrix function trace estimation

2.1 Introduction

A number of problems which appear in applications of machine learning, signal processing, scientific computing, statistics, computational biology and computational physics [14, 44, 45, 15, 16, 17] can all be posed as the matrix function trace estimation problem discussed in the previous chapter. The development of fast and scalable algorithms to perform this task has long been a primary focus of research in these fields. An important instance of the trace estimation problem is that of approximating $\log(\det(A))$, the log-determinant of a positive definite matrix A. Log-determinants of covariance and precision matrices play an important role in Gaussian processes and Gaussian graphical models [45]. Log-determinant computations also appear in applications such as kernel learning [46], Bayesian Learning [47], spatial statistics [48] and Markov field models [17, 49].

Another instance of the trace estimation problem in applications is that of estimating Schatten p-norms, particularly the nuclear norm, since this norm is used as the convex surrogate of the matrix rank. The Schatten p-norms appear in convex optimization problems, e.g., in the context of matrix completion [50], in differential privacy problems [51], and in sketching and streaming models [52]. On the other hand, in uncertainty quantification and in lattice quantum chromodynamics [16, 53], it is necessary to estimate the trace of the inverse of covariance matrices. Estimating the Estrada index (trace of exponential function) is another illustration of the problem with applications in protein indexing [44], statistical thermodynamics [54] and information theory [55].

As mentioned in the previous chapter, the naive way to compute these traces is to obtain the complete eigendecomposition of the given matrix. A popular approach to computing the log-determinant is to exploit the Cholesky decomposition [56]. Given the decomposition $A = LL^\top$, the log-determinant of A is $\log\det(A) = 2\sum_i \log(L_{ii})$. Computing the Schatten norms in a standard way would typically require the singular value decomposition (SVD) of the matrix. These methods have cubic computational complexity (in terms of the matrix dimension, i.e., $O(n^3)$ cost) in general, and are not viable for large scale applications. In this chapter, we study inexpensive methods for accurately estimating these traces for large matrices.

We study the method called "Stochastic Lanczos Quadrature" (SLQ) for approximating the trace of functions of large matrices [14, 15]. The method combines three key ingredients. First, the stochastic trace estimator [19] discussed in the previous chapter is considered for approximating the trace. Next, the bilinear form that appears in the trace estimator is expressed as a Riemann-Stieltjes integral, and the Gauss quadrature rule is used to approximate this integral. Finally, the Lanczos algorithm is used to obtain the weights and the nodes of the quadrature rule. We establish multiplicative and additive approximation error bounds for the trace obtained by using the method. We show that the Lanczos Quadrature approximation has faster convergence rate compared to popular methods such as those based on Chebyshev or Taylor series expansions. The analysis can be extended to any matrix functions that are analytic inside a closed interval and are analytically continuable to an open Bernstein ellipse [57]. We consider several important trace estimation problems and their applications.

Related Work

A plethora of methods have been developed in the literature to deal with trace estimation problems. In the following, we discuss some of the works that are closely related to SLQ, particularly those that invoke the stochastic trace estimator. The stochastic trace estimator has been employed for a number of applications in the literature, for example, for estimating the diagonal of a matrix, for counting eigenvalues inside an interval [58], and for estimating the numerical rank [2, 3]. For the log-determinant computation, a few methods have been proposed, which also invoke the stochastic trace estimator.

These methods differ in the approach used to approximate the log function. Article [17] used the Chebyshev polynomial approximations for the log function. The log function was approximated using the Taylor series expansions in [59]. Article [49] provided an improved analysis for the log-determinant computations using these Taylor series expansions. Aune et. al [48] adopted the method proposed in [60] to estimate the log function. Here, the Cauchy integral formula of the log function is considered and the Trapezoidal rule is invoked to approximate the integral. This method is equivalent to using a rational approximation for the function. The method requires solving a series of linear systems and is generally expensive.

Not many fast algorithms are available in the literature to approximate the nuclear norm and Schatten-p norms; see [52, 61] for discussions. Article [62] extends the idea of using Chebyshev expansions developed in [58, 17] to approximate the trace of various matrix functions including Schatten norms, the Estrada index and the trace of matrix inverse. Related articles on estimating the trace of matrix inverse and other matrix functions are [16, 53].

2.2 Trace estimation problems

We begin by first discussing a few trace estimation problems that arise in certain application areas.

Log-determinant: As previously mentioned, the log-determinants have numerous applications in machine learning and related fields. The logarithm of the determinant of a given positive definite matrix $A \in \mathbb{R}^{n\times n}$, is equal to the trace of the logarithm of the matrix, i.e.,

$$\log\det(A) = \mathtt{tr}(\log(A)) = \sum_{i=1}^{n} \log(\lambda_i).$$

So, estimating the log-determinant of a matrix is equivalent to estimating the trace of the matrix function $f(A) = \log(A)$.

Suppose the positive definite matrix A has its eigenvalues inside the interval $[\lambda_{\min}, \lambda_{\max}]$, then the logarithm function $f(t) = \log(t)$ is analytic over this interval. When computing the log-determinant of a matrix, the case $\lambda_{\min} = 0$ is obviously excluded, where the function has its singularity. The Lanczos algorithm requires the input matrix to be symmetric. If A is non-symmetric, we can either consider the matrix[1] $A^\top A$, since $\log\det(A^T A) = 2\log|\det(A)|$.

Log-likelihood: The problem of computing the likelihood function occurs in applications related to Gaussian processes [45]. Maximum Likelihood Estimation (MLE) is a popular approach used for parameter estimation when high dimensional Gaussian models are used, especially in statistical machine learning. The objective in parameter estimation is to maximize the log-likelihood function with respect to a hyperparameter vector ξ:

$$\log p(z \mid \xi) = -\frac{1}{2} z^\top S(\xi)^{-1} z - \frac{1}{2}\log\det S(\xi) - \frac{n}{2}\log(2\pi), \tag{2.1}$$

where z is the data vector and $S(\xi)$ is the covariance matrix parameterized by ξ. The second term (log-determinant) in (2.1) can be computed by using the SLQ method. We observe that the first term in (2.1) resembles the quadratic form that appears in the trace estimator, and it can be also computed by using the Lanczos Quadrature method. That is, we can estimate the term $z^\top S(\xi)^{-1} z$ using m steps of the Lanczos algorithm applied to $z/\|z\|$ as the starting vector, then compute the quadrature rule for the inverse function $f(t) = t^{-1}$, and rescale the result by $\|z\|^2$.

Schatten p-norms: Given an input matrix $X \in \mathbb{R}^{d\times n}$, the Schatten p-norm of X is defined as $\|X\|_p = \left(\sum_{i=1}^r \sigma_i^p \right)^{1/p} i$, where the σ_i's are the singular values of X and r its rank. The nuclear norm is the Schatten 1-norm so it is just the sum of the singular values. Estimating the nuclear norm and the Schatten p-norms of large matrices has many applications as seen earlier. Suppose we define a positive semidefinite matrix A as[1] $A = X^\top X$ or $A = XX^\top$. Then, the Schatten p-norm of X is defined as

$$\|X\|_p = \left(\sum_{i=1}^r \lambda_i^{p/2} \right)^{1/p} = \mathtt{tr}(A^{p/2}).$$

Hence, Schatten p-norms (the nuclear norm being a special case with $p = 1$) are the traces of matrix functions of A with $f(t) = t^{p/2}$.

[1]The matrix product need not be formed explicitly since the Lanczos algorithm requires only matrix vector products.

Spectral Density: Next, we consider computing the spectral density of a matrix [63], a very common problem in solid state physics. The spectral density of matrix, also known as Density of States (DOS) in physics, is a probability density distribution that measures the likelihood of finding eigenvalues of the matrix at a given point on the real line. Formally, the spectral density of a matrix is expressed as a sum of delta functions of the eigenvalues of the matrix. That is,

$$\phi(t) = \frac{1}{n}\sum_{i=1}^{n}\delta(t-\lambda_i),$$

where δ is the Dirac distribution or Dirac δ-function. This is not a proper function but a distribution and it is clearly not practically computable as it is defined. What is important is to compute a smoothed or approximate version of it that does not require computing eigenvalues, and several inexpensive methods have been proposed for this purpose, [64, 63]. Recently, the DOS has been used in applications such as eigenvalue problem, e.g., for spectral slicing, for counting eigenvalues in intervals ('eigencounts') [58], and for estimating ranks [2, 3].

Article [63] reviews a set of inexpensive methods for computing the DOS. We will further discuss the concept of DOS and its applications in Chapter 3.

Eigencount and Numerical Rank: Next, we consider counting eigenvalues located in a given interval (Eigencount) and the related problem of estimating the numerical rank of a matrix. Estimating the number of eigenvalues $\eta_{[a,b]}$ located in a given interval $[a, b]$ of a large sparse symmetric matrix is a key ingredient of effective eigensolvers [58], because these eigensolvers require an estimate of the dimension of the eigenspace to compute to allocate resources and tune the method under consideration. Estimating the numerical rank $r_\epsilon = \eta_{[\epsilon,\lambda_1]}$ is another closely related problem that occurs in machine learning and data analysis applications such as Principal Component Analysis (PCA), low rank approximations, and reduced rank regression [2, 3]. Both of these problems can be viewed from the angle of estimating the trace of a certain eigen-projector, i.e., the number of eigenvalues $\eta_{[a,\ b]}$ in $[a,\ b]$ satisfies:

$$\eta_{[a,b]} = \mathtt{tr}(P), \text{ where } P = \sum_{\lambda_i\ \in\ [a,\ b]} u_i u_i^\top .$$

We can interpret P as a step function of A given by

$$P = h(A), \text{ where } h(t) = \begin{cases} 1 & \text{if } t \in [a,\ b] \\ 0 & \text{otherwise} \end{cases}. \tag{2.2}$$

The problem then is to find an estimate of the trace of $h(A)$. A few inexpensive methods are proposed in [58, 2, 3] to approximately compute this trace. We can also compute the eigencount from the spectral density since

$$\eta_{[a,\ b]} = \int_a^b \sum_j \delta(t - \lambda_j) dt \equiv \int_a^b n\phi(t) dt. \tag{2.3}$$

The problem of numerical rank estimation is studied in chapters 3 and 4, where various methods are presented to compute the rank efficiently.

Other Applications: Other frequent matrix function trace estimation problems include estimating the trace of a matrix inverse and the Estrada index.
Trace of a matrix inverse: The matrix inverse trace estimation problem amounts to computing the trace of the inverse function $f(t) = t^{-1}$ of a positive definite matrix $A \in \mathbb{R}^{n\times n}$, whose eigenvalues lie in the interval $[\lambda_{\min}, \lambda_{\max}]$ with $\lambda_{\min} > 0$. This problem appears in uncertainty quantification and in lattice quantum chromodynamics [16, 53], where it is necessary to estimate the trace of the inverse of covariance matrices.
Estrada index: Estimation of the Estrada index of graphs is popular in computational biology. This problem accounts to estimating the trace of the exponential function, i.e., $f(t) = \exp(t)$. Note that, here the matrix A is the adjacency matrix of a graph, which need not be positive definite in general. However, the matrix $\exp(A)$ is always positive definite and our method and theory are applicable in this case.

2.3 Stochastic Lanczos Quadrature

The Lanczos Quadrature method was developed by Gene Golub and his collaborators in a series of articles [14, 15]. The idea of combining the stochastic trace estimator with the Lanczos Quadrature method appeared in [14] for estimating the trace of the inverse and the determinant of matrices. Given a symmetric positive definite matrix $A \in \mathbb{R}^{n\times n}$, we wish to compute the trace of the matrix function $f(A)$, i.e., the expression given by (1.1), where we assume that the function f is analytic inside a closed interval containing the spectrum of A. To estimate the trace, we invoke the stochastic trace estimator [19], which we discussed in the previous chapter. Recall that the method estimates the trace $\texttt{tr}(f(A))$ by generating random vectors $u_l, l = 1, \dots, \mathrm{n_v}$, with Rademacher distribution (vectors with ± 1 entries of equal probability), forming unit vectors $v_l = u_l / \|u_l\|_2$, and then computing the average over the samples $v_l^\top f(A) v_l$:

$$\texttt{tr}(f(A)) \approx \frac{n}{\mathrm{n_v}} \sum_{l=1}^{\mathrm{n_v}} v_l^\top f(A) v_l. \tag{2.4}$$

Hence, for computing the trace we only need to estimate the scalars of the form $v^\top f(A) v$, and the explicit computation of $f(A)$ is never needed.

The scalar (quadratic form) quantities $v^\top f(A) v$ are computed by transforming them to a Riemann-Stieltjes integral, and then employing the Gauss quadrature rule to approximate this integral. Consider the eigen-decomposition of A as $A = Q \Lambda Q^\top$. Then, we can write the scalar product as,

$$v^\top f(A) v = v^\top Q f(\Lambda) Q^\top v = \sum_{i=1}^{n} f(\lambda_i) \mu_i^2, \tag{2.5}$$

where μ_i are the components of the vector $Q^\top v$. The above sum can be considered as a Riemann-Stieltjes integral given by,

$$I = v^\top f(A) v = \sum_{i=1}^{n} f(\lambda_i) \mu_i^2 = \int_a^b f(t) d\mu(t), \tag{2.6}$$

where the measure $\mu(t)$ is a piecewise constant function defined as

$$\mu(t) = \begin{cases} 0, & \text{if } t < a = \lambda_1, \\ \sum_{j=1}^{i-1} \mu_j^2, & \text{if } \lambda_{i-1} \le t < \lambda_i, \ i = 2, \dots, n, \\ \sum_{j=1}^{n} \mu_j^2, & \text{if } b = \lambda_n \le t, \end{cases} \tag{2.7}$$

assuming that the eigenvalues λ_i are ordered nondecreasingly. Next, the integral can be estimated using the Gauss quadrature rule [15]

$$\int_a^b f(t)d\mu(t) \approx \sum_{k=0}^{m} \omega_k f(\theta_k), \tag{2.8}$$

where $\{\omega_k\}$ are the weights and $\{\theta_k\}$ are the nodes of the $(m+1)$-point Gauss quadrature, which are unknowns and need to be determined.

An elegant way to compute the nodes and the weights of the quadrature rule is to use the Lanczos algorithm [15]. For a given real symmetric matrix $A \in \mathbb{R}^{n\times n}$ and a starting vector w_0 of unit 2-norm, the Lanczos algorithm generates an orthonormal basis W_{m+1} for the *Krylov subspace* $\text{Span}\{w_0, Aw_0, \dots, A^m w_0\}$ such that $W_{m+1}^\top A W_{m+1} = T_{m+1}$, where T_{m+1} is an $(m+1)\times(m+1)$ tridiagonal matrix. For details see [56]. The columns w_k of W_{m+1} are related as

$$w_k = p_{k-1}(A)w_0,\ k = 1, \dots, m,$$

where p_k are the Lanczos polynomials. The vectors w_k are orthonormal, and we can show that the Lanczos polynomials are orthogonal with respect to the measure $\mu(t)$ in (2.7); see Theorem 4.2 in [15]. Therefore, the nodes and the weights of the quadrature rule in (2.8) can be computed as the eigenvalues and the squares of the first entries of the eigenvectors of T_{m+1}. Then, we can approximate the quadratic form (2.5) as,

$$v^\top f(A)v \approx \sum_{k=0}^{m} \tau_k^2 f(\theta_k) \quad \text{with} \quad \tau_k^2 = \left[e_1^\top y_k\right]^2, \tag{2.9}$$

where $(\theta_k, y_k), k = 0, 1, ..., m$ are eigenpairs of T_{m+1} by using v as the starting vector w_0. Thus, the trace of matrix function $f(A)$ can be computed as,

$$\texttt{tr}(f(A)) \approx \frac{n}{\mathrm{n_v}} \sum_{l=1}^{\mathrm{n_v}} \left(\sum_{k=0}^{m} (\tau_k^{(l)})^2 f(\theta_k^{(l)}) \right), \tag{2.10}$$

where $(\theta_k^{(l)}, \tau_k^{(l)}), k = 0, 1, ..., m$ are eigenvalues and the first entries of the eigenvectors of the tridiagonal matrix $T_{m+1}^{(l)}$ corresponding to the starting vectors $v_l, l = 1, \dots, \mathrm{n_v}$. This method is far less costly than computing the eigenvalues of the matrix A for the purpose of computing the trace via (1.1). The Stochastic Lanczos Quadrature algorithm corresponding to this procedure is summarized in Algorithm 1.

Algorithm 1 Trace of a matrix function by SLQ using the Lanczos algorithm

```
**Input:** SPD matrix $A \in \mathbb{R}^{n\times n}$, function $f$, degree $m$ and $\mathrm{n_v}$.
**Output:** Approximate trace $\Gamma$ of $f(A)$.
**for** $l = 1$ to $\mathrm{n_v}$ **do**
  **1.** Generate a Rademacher random vector $u_l$ and form unit vector $v_l = u_l/\|u_l\|_2$
  **2.** $T = \text{Lanczos}(A, v_l, m+1)$; that is, apply $m+1$ steps of Lanczos to $A$ with $v_l$ as the starting vector.
  **3.** $[Y, \Theta] = \texttt{eig}(T)$ and compute $\tau_k = [e_1^\top y_k]$ for $k = 0, \ldots, m$
  **4.** $\Gamma \leftarrow \Gamma + \sum_{k=0}^{m} \tau_k^2 f(\theta_k)$.
**end for**
Output $\Gamma = \frac{n}{\mathrm{n_v}}\Gamma$.
```

2.4 Analysis

One of the main contributions of this chapter is the derivation of multiplicative error bounds for approximating the trace of a matrix function using SLQ. Additive error bounds are also established for the log-determinant approximation of a positive definite matrix and the log-likelihood function estimation. The nuclear norm and Schatten-p norms estimation of a general matrix is discussed in the latter part of the section. First, we give the following definition: A Bernstein ellipse E_ρ is an ellipse on the complex plane with focii at $-1, 1$ and major semi-axis $(\rho + \rho^{-1})/2$, with $\rho > 1$ [57]. It can be viewed as a mapping of the circle $C(0, \rho)$ (center at zero and radius ρ) using the Joukowsky transform $(z + z^{-1})/2$. Hence we can have two values of ρ that are inverses of each other, which give the same ellipse. Following is our main result:

Theorem 1 *Consider a symmetric positive definite matrix $A \in \mathbb{R}^{n\times n}$ with eigenvalues in $[\lambda_{\min}, \lambda_{\max}]$ and condition number $\kappa = \lambda_{\max}/\lambda_{\min}$. Let f be a function analytic in $[\lambda_{\min}, \lambda_{\max}]$ and be either positive or negative (i.e., does not cross zero) inside this interval. Denote by m_f the absolute minimum value of f in the interval. Assume that f is analytically continuable in an open Bernstein ellipse E_ρ encompassing the interval,* *with foci $\lambda_{\min}, \lambda_{\max}$ and sum of the two semi-axes ρ, such that $|f(z)| \leq M_\rho$ for all $z \in E_\rho$. Let ϵ, η be constants in $(0,1)$. Then for SLQ parameters satisfying:*

- $m \geq \frac{1}{2}\log\left(\frac{4M_\rho(\lambda_{\max}-\lambda_{\min})}{\epsilon m_\rho(\rho^2-1)}\right)/\log(\rho)$ *number of Lanczos steps, and*
- $\mathrm{n_v} \geq (24/\epsilon^2)\log(2/\eta)$ *number of starting Rademacher vectors,*

the output Γ of the Stochastic Lanczos Quadrature method is such that:

$$\Pr\Big[|\mathtt{tr}(f(A)) - \Gamma| \leq \epsilon\,|\mathtt{tr}(f(A))|\Big] \geq 1-\eta. \tag{2.11}$$

In particular for $\rho = (\sqrt{\kappa}+1)/(\sqrt{\kappa}-1)$, for which the function of interest is analytic inside E_ρ, we have $m \geq (\sqrt{\kappa}/4)\log(K/\epsilon)$, with $K = (\lambda_{\max}-\lambda_{\min})(\sqrt{\kappa}-1)^2 M_\rho/(\sqrt{\kappa}m_f)$.

To prove the theorem, we first derive error bounds for the Lanczos Quadrature approximation (which gives the convergence rate), using the facts that an $(m+1)$-point Gauss Quadrature rule is exact for any $2m+1$ degree polynomial and that the function is analytic inside an interval and is analytically continuable in a Bernstein ellipse. We then combine this bound with the error bounds for the stochastic trace estimator to obtain the above result.

Convergence rate for the Lanczos Quadrature

In order to prove Theorem 1, we first establish the convergence rate for the Lanczos Quadrature approximation of the quadratic form. Recall that the quadratic form $v^\top f(A)v$ can be written as a Riemann Stieltjes integral I, as given in (2.6). Let I_m denote the $(m+1)$-point Gauss Quadrature rule that approximates the integral I, given by

$$I_m = \sum_{k=0}^{m} \omega_k f(\theta_k),$$

where $\{\omega_k\}$ are the weights and $\{\theta_k\}$ are the nodes, computed by using $m+1$ steps of the Lanczos algorithm. The well known error analysis for the Gauss Quadrature rule is given by [15],

$$|I - I_m| = \frac{f^{(2m+2)}(\zeta)}{(2m+2)!}\int_a^b \left[\prod_{k=0}^{m}(t-\theta_k)\right]^2 d\mu(t), \tag{2.12}$$

for some $a < \zeta < b$. However, this analysis might not be useful for our purpose, since the higher derivatives of both the logarithm and the square root function become excessively large in the interval of interest. Hence, in this work, we establish improved error analysis for the Lanczos Quadrature approximations, using some classical results developed in the literature, with the fact that functions of interest are analytic over a certain interval. We begin with the following result.

Theorem 2 *Let a function g be analytic in $[-1, 1]$ and analytically continuable in the open Bernstein ellipse E_ρ with foci ± 1 and sum of major and minor axis equal to $\rho > 1$, where it satisfies $|g(z)| \le M_\rho$. Then the $(m+1)$-step Lanczos Quadrature approximation satisfies*

$$|I - I_m| \le \frac{4M_\rho}{(\rho^2 - 1)\rho^{2m}}. \tag{2.13}$$

Proof. We follow a similar argument developed in [65] that estimates the error of Gaussian quadratures for a Riemann integral. The result and the proof strategy are usually covered in standard textbooks, e.g., [57, Thm. 19.3]. In our case, the integral is a Riemann-Stieltjes integral with respect to a specific measure given in (2.7). As a result, the bound admits the same rate but with a different constant.

For the given function g that is analytic over the interval $[-1, 1]$, consider the $2m+1$ degree Chebyshev polynomial approximation of $g(t)$, i.e.,

$$P_{2m} = \sum_{j=0}^{2m+1} a_j T_j(t) \approx g(t).$$

We know that the $(m + 1)$-point Gauss Quadrature rule is exact for any polynomial of degree upto $2m + 1$, see [15, Thm. 6.3] or [57, Thm. 19.1]. This can also be deduced from the error term in (2.12). Hence, the error in integrating g is the same as the error in integrating $g - P_{2m}$. Thus, we have

$$\begin{aligned} |I - I_m| &= |I(g - P_{2m}) - I_m(g - P_{2m})| \le |I(g - P_{2m})| + |I_m(g - P_{2m})| \\ &= \left| I\Big(\sum_{j=2m+2}^{\infty} a_j T_j(t) \Big) \right| + \left| I_m\Big(\sum_{j=2m+2}^{\infty} a_j T_j(t) \Big) \right| \\ &\le \sum_{j=2m+2}^{\infty} |a_j| \Big[|I(T_j)| + |I_m(T_j)| \Big] \end{aligned}$$

Next, we obtain bounds for the three terms inside the summation above.

If the function g is analytic in $[-1,1]$ and analytically continuable in the Bernstein ellipse E_ρ, then for the Chebyshev coefficients we have from Theorem 8.1 in [57] and eq. (14) in [65],

$$|a_j| \leq \frac{2M_\rho}{\rho^j}.$$

Next, for the Quadrature rule $I_m(T_j)$, we have

$$I_m(T_j) = \sum_{k=0}^{m} \tau_k^2 T_j(\theta_k) \leq \sum_{k=0}^{m} |\tau_k^2||T_j(\theta_k)| \leq 1.$$

The last inequality results from the fact that, for $f(t) = 1$, the quadrature rule is exact, and the thus integral is equal to 1 ($v_l^\top f(A) v_l = v_l^\top v_l = 1$). Therefore, the weights τ_k^2 must sum to 1. The maximum value of T_j inside the interval is 1. Finally, in order to bound the Riemann-Stieltjes integral $I(T_j)$, we use the following:

$$I(T_j) = v^\top T_j(A) v \leq \lambda_{\max}(T_j(A)) = 1,$$

by the min-max theorem and $\|v\| = 1$. Therefore,

$$|I - I_m| \leq \sum_{j=2m+2}^{\infty} \frac{2M_\rho}{\rho^j}[1+1].$$

Since the Gauss quadrature rule is a symmetric rule [65], the error in integration of $T_j(t)$ for any odd j will be equal to zero. Thus, we get the result in the theorem

$$|I - I_m| \leq \frac{4M_\rho}{(\rho^2 - 1)\rho^{2m}}.$$

□

Remark 1 *The convergence rate for the Chebyshev polynomial approximation of an analytic function is $O(1/\rho^m)$; see Theorem 8.2 in [57]. Hence, the Lanczos Quadrature approximation is twice as fast as the Chebyshev approximation. Moreover, it is known that the Gauss quadrature has the maximal polynomial order of accuracy [57].*

Theorem 2 holds for functions that are analytic over $[-1, 1]$. The functions considered in this paper such as logarithm, exponential and square root functions are analytic over $[\lambda_{\min}, \lambda_{\max}]$ for $\lambda_{\min} > 0$. Hence, we need to use the following transform to get the right interval.

If $f(x)$ is analytic on $[\lambda_{\min}, \lambda_{\max}]$, then

$$g(t) = f\left[\left(\frac{\lambda_{\max} - \lambda_{\min}}{2}\right)t + \left(\frac{\lambda_{\max} + \lambda_{\min}}{2}\right)\right]$$

is analytic on $[-1, 1]$. If we denote the error in the Quadrature rule for approximating the integral of function f as $E(f)$, then we have

$$E(f) = \left(\frac{\lambda_{\max} - \lambda_{\min}}{2}\right)E(g).$$

The function g will have its singularity at $t_0 = \alpha = -\frac{\kappa+1}{\kappa-1}$. Hence, we choose the ellipse E_ρ with the semimajor axis length of $|\alpha|$ where g is analytic inside. Then, the convergence rate ρ will be

$$\rho = \alpha \pm \sqrt{\alpha^2 - 1} = \frac{\sqrt{\kappa} + 1}{\sqrt{\kappa} - 1} > 1.$$

The sign is chosen such that $\rho > 1$. From theorem 2, the error $E(g) \leq 4M_\rho/[(\rho^2-1)\rho^{2m}]$, where $|g(z)| < M_\rho$ inside E_ρ. Hence, the error $E(f)$ will be

$$E(f) = \left(\frac{\lambda_{\max} - \lambda_{\min}}{2}\right)\frac{4M_\rho}{(\rho^2 - 1)\rho^{2m}} = \frac{(\lambda_{\max} - \lambda_{\min})(\sqrt{\kappa} - 1)^2 M_\rho}{2\sqrt{\kappa}\rho^{2m}},$$

with ρ defined as above. Thus, for a function f that is analytic on $[\lambda_{\min}, \lambda_{\max}]$ and $C_\rho = 2M_\rho(\lambda_{\max} - \lambda_{\min})/(\rho^2 - 1) = (\lambda_{\max} - \lambda_{\min})(\sqrt{\kappa} - 1)^2 M_\rho/(2\sqrt{\kappa})$, we have

$$\left|v^\top f(A)v - \sum_{k=0}^{m} \tau_k^2 f(\theta_k)\right| \leq \frac{C_\rho}{\rho^{2m}}. \tag{2.14}$$

Approximation error of the trace estimator

The quadratic form $v^\top f(A)v$ for which we derived the error bounds in the previous section comes from the Hutchinson trace estimator. Let us denote this estimator as

$\mathtt{tr}_{\mathrm{n_v}}(A) = \frac{n}{\mathrm{n_v}} \sum_{l=1}^{\mathrm{n_v}} v_l^\top A v_l$. The convergence analysis for the stochastic trace estimator was developed in [20], and improved in [21] for sample vectors with different probability distributions. We state the following theorem which is proved in [21].

Theorem 3 *Let A be an $n \times n$ symmetric positive semidefinite matrix and $v_l, l = 1, \ldots, \mathrm{n_v}$ be random starting vectors sampled from the Rademacher distribution and scaled to a unit 2-norm. Then, with $\mathrm{n_v} \geq (6/\epsilon^2)\log(2/\eta)$, we have*

$$\Pr\left[|\mathtt{tr}_{\mathrm{n_v}}(A) - \mathtt{tr}(A)| \leq \epsilon |\mathtt{tr}(A)|\right] \geq 1 - \eta.$$

The above theorem can be used to bound the trace of any matrix function $f(A)$, if the function is either positive or negative inside the spectrum interval. Therefore, the theorem holds for the square root function, its powers, and the exponential. However, for the logarithm function, different scenarios occur depending on the spectrum, which will be discussed later. Let Γ be the output of the Stochastic Lanczos Quadrature method to estimate the trace of such functions, given by

$$\Gamma = \frac{n}{\mathrm{n_v}} \sum_{l=1}^{\mathrm{n_v}} \left(\sum_{k=0}^{m} (\tau_k^{(l)})^2 f(\theta_k^{(l)}) \right). \tag{2.15}$$

We need the following lemma.

Lemma 1 *Let $A \in \mathbb{R}^{n \times n}$ be a symmetric positive definite matrix with eigenvalues in $[\lambda_{\min}, \lambda_{\max}]$ and condition number $\kappa = \lambda_{\max}/\lambda_{\min}$, and f be an analytic function in this interval with $|f(z)| \leq M_\rho$, for all z inside a Bernstein ellipse E_ρ that encompasses the interval. Then, the following inequality holds:*

$$|\mathtt{tr}_{\mathrm{n_v}}(f(A)) - \Gamma| \leq \frac{nC_\rho}{\rho^{2m}},$$

where $\rho = (\sqrt{\kappa}+1)/(\sqrt{\kappa}-1)$ and $C_\rho = 2M_\rho(\lambda_{\max} - \lambda_{\min})/(\rho^2 - 1) = (\lambda_{\max} - \lambda_{\min})(\sqrt{\kappa} - 1)^2 M_\rho/(2\sqrt{\kappa})$.

Proof. The lemma follows from the equation (2.14). We have

$$\begin{aligned}|\mathtt{tr}_{\mathrm{n_v}}(f(A)) - \Gamma| &= \frac{n}{\mathrm{n_v}}\left|\sum_{l=1}^{\mathrm{n_v}} v_l^\top f(A) v_l - \sum_{l=1}^{\mathrm{n_v}} I_m^{(l)}\right| \\ &\leq \frac{n}{\mathrm{n_v}}\sum_{l=1}^{\mathrm{n_v}} |v_l^\top f(A) v_l - I_m^{(l)}| \\ &\leq \frac{n}{\mathrm{n_v}}\sum_{l=1}^{\mathrm{n_v}} \frac{C_\rho}{\rho^{2m}} = \frac{nC_\rho}{\rho^{2m}}.\end{aligned}$$

□

Now, we are ready to prove Theorem 1. Based on the condition of m,

$$\log\frac{K}{\epsilon} \leq \frac{4m}{\sqrt{\kappa}} \leq 2m\log\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right).$$

Therefore,

$$\frac{K}{\epsilon} \leq \rho^{2m} \quad \text{and hence} \quad \frac{C_\rho}{\rho^{2m}} \leq \frac{\epsilon}{2} f_{\min}(\lambda),$$

where $f_{\min}(\lambda) \equiv m_f$ is the absolute minimum of the function in the interval $[\lambda_{\min}, \lambda_{\max}]$. This gives us the lower bound on the degree m in the Theorem. Then, from Lemma 1 we have

$$|\mathtt{tr}_{\mathrm{n_v}}(f(A)) - \Gamma| \leq \frac{\epsilon n}{2} f_{\min}(\lambda) \leq \frac{\epsilon}{2}|\mathtt{tr}(f(A))|. \tag{2.16}$$

From Theorem 3, we have

$$\Pr\left[|\mathtt{tr}(f(A)) - \mathtt{tr}_{\mathrm{n_v}}(f(A))| \leq \frac{\epsilon}{2}|\mathtt{tr}(f(A))|\right] \geq 1-\eta. \tag{2.17}$$

Combining the above two inequalities (2.16) and (2.17) leads to the result in Theorem 1:

$$\begin{aligned}1-\eta &\leq \Pr\left[|\mathtt{tr}(f(A)) - \mathtt{tr}_{\mathrm{n_v}}(f(A))| \leq \frac{\epsilon}{2}|\mathtt{tr}(f(A))|\right] \\ &\leq \Pr\left[|\mathtt{tr}(f(A)) - \mathtt{tr}_{\mathrm{n_v}}(f(A))| + |\mathtt{tr}_{\mathrm{n_v}}(f(A)) - \Gamma| \leq \frac{\epsilon}{2}|\mathtt{tr}(f(A))| + \frac{\epsilon}{2}|\mathtt{tr}(f(A))|\right] \\ &\leq \Pr\left[|\mathtt{tr}(f(A)) - \Gamma| \leq \epsilon\,|\mathtt{tr}(f(A))|\right].\end{aligned}$$

For comparison, note that for Chebyshev approximations [17], the required degree of the polynomial is $m = \Theta(\sqrt{\kappa}\log\frac{\kappa}{\epsilon})$ and for Taylor approximations [49], $m = O(\kappa\log\frac{\kappa}{\epsilon})$. Recall from Remark 1, the Lanczos algorithm is superior to the Chebyshev expansions because the former approximation converges twice as fast as does the latter. Clearly, the Lanczos approximation also converges faster than the Taylor approximation. Theorem 1 can be used to establish the error bounds for approximating the log-determinants and the Schatten p-norms. The quality and the complexity of the algorithms depend on the condition number κ, since matrix function approximations become harder when matrices become more ill-conditioned, which requires higher degree approximations.

Bounds for Log-determinant

For the logarithm function, we encounter three different scenarios depending on the spectrum of the matrix. The first case is when $\lambda_{\max} < 1$, $\log(A)$ is negative definite and the log-determinant will always be negative. Thus, the conditions of Theorem 1 are satisfied. Similarly, Theorem 1 holds in the second case when $\lambda_{\min} > 1$, since $\log(A)$ is positive definite. In the third case when $\lambda_{\min} < 1$ and $\lambda_{\max} > 1$, however, we cannot obtain multiplicative error bounds of the form given in Theorem 1, since the log function will cross zero inside the interval. In the worst case, the log-determinant can be zero. One simple workaround to avoid this case is to scale the matrix such that its eigenvalues are either all smaller than 1 or all greater than 1; however, such an approach requires the computation of the extreme eigenvalues of A. The following corollary gives additive error bounds without scaling; it holds for any SPD matrix.

Corollary 1 *Given $\epsilon, \eta \in (0,1)$, a SPD matrix $A \in \mathbb{R}^{n\times n}$ with its eigenvalues in $[\lambda_{\min}, \lambda_{\max}]$, and condition number $\kappa = \lambda_{\max}/\lambda_{\min}$, for SLQ parameters:*

- $m \geq (\sqrt{3\kappa}/4)\log(K_1/\epsilon)$ *number of Lanczos steps, and*
- $\mathrm{n_v} \geq (24/\epsilon^2)\log(1+\kappa))^2\log(2/\eta)$ *number of starting vectors,*

where $K_1 = 5\kappa\log(2(\kappa+1))/\sqrt{2\kappa+1}$, we have

$$\Pr\bigg[|\log\det(A) - \Gamma| \leq \epsilon n\bigg] \geq 1-\eta, \tag{2.18}$$

where Γ is the output of the Stochastic Lanczos Quadrature method for log-determinant computation.

Proof. The proof of the Corollary is on the similar lines as the proof of Theorem 1. In the logarithm case, Theorem 2 still holds, however, we need to choose a smaller ellipse (smaller α) since the log function goes to infinity near the singularity. We choose $\alpha = (\kappa+1)/\kappa$, then $\rho = (\sqrt{2\kappa+1}+1)/(\sqrt{2\kappa+1}-1)$. For theorem 3, we consider the fact that, if $B = \frac{A}{\lambda_{\max}+\lambda_{\min}}$, then

$$\log\det A = \log\det B + n\log(\lambda_{\max}+\lambda_{\min}).$$

Since the matrix B has its eigenvalues inside $(0,1)$, the logarithm function is negative and we hence can apply Theorem 3 with $f(A) = \log\left(\frac{A}{\lambda_{\max}+\lambda_{\min}}\right)$, and then add and subtract $n\log(\lambda_{\max}+\lambda_{\min})$ to get an inequality of the form (2.17). To compute the parameters in Theorem 2, we consider this function $f(t) = \log\left(\frac{t}{\lambda_{\max}+\lambda_{\min}}\right)$, and the ellipse E_ρ where the function is analytic with ρ as defined above. Then, we have $\rho^2-1 = (4\sqrt{2\kappa+1})/(\sqrt{2\kappa+1}-1)^2$ and M_ρ is computed as,

$$\begin{aligned}\max_{z\in E_\rho}|\log(z)| &\leq \max_{z\in E_\rho}\sqrt{(\log|z|)^2+\pi^2}\\ &= \sqrt{(\log|1/2\kappa|)^2+\pi^2} \leq 5\log(2(\kappa+1)) = M_\rho.\end{aligned}$$

The first inequality comes from the fact $|\log(z)| = |\log|z| + i\arg(z)| \leq \sqrt{(\log|z|)^2+\pi^2}$. The ellipse E_ρ is defined with foci at $1/(\kappa+1)$ and $\kappa/(\kappa+1)$. The maximum occurs at end point $z_0 = 1/(2\kappa)$. As in the proof of Theorem 1, we have

$$E(f) = \left(\frac{\kappa-1}{\kappa+1}\right)\frac{2M_\rho}{(\rho^2-1)\rho^{2m}} \leq \frac{5\kappa\log(2(\kappa+1))}{2\sqrt{2\kappa+1}\rho^{2m}}.$$

The K_1 value is obtained by setting

$$\frac{n5\kappa\log(2(\kappa+1))}{2\sqrt{2\kappa+1}\rho^{2m}} \leq \frac{\epsilon n}{2}.$$

The lower bound for m_f is simplified using the fact $\sqrt{2\kappa+1} \leq \sqrt{3\kappa}$. We can then conclude the corollary using $|\log\det B| \leq n\log(1+\kappa)$ and choosing $\epsilon = \epsilon/\log(1+\kappa)$ in Theorem 3. □

Bounds for Log-likelihood function

Recall the log-likelihood function defined in (2.1). The log-determinant term in it can be bounded as above. The first term $z^\top S(\xi)^{-1}z$ is computed using the Lanczos quadrature method with $z/\|z\|$ as the starting vector for the Lanczos algorithm. The following corollary gives the error bound for the log-likelihood function estimation by SLQ, which follows from Theorem 1 and Corollary 1.

Corollary 2 *Given a data vector $z \in \mathbb{R}^n$, a covariance matrix $S(\xi) \in \mathbb{R}^{n\times n}$ with hyperparameter ξ and its eigenvalues in $[\lambda_{\min}, \lambda_{\max}]$, and constants $\epsilon, \eta \in (0,1)$, for SLQ parameters:*

- $m_1 \geq (\sqrt{3\kappa}/4)\log(K_1/\epsilon)$, $m_2 \geq (\sqrt{3\kappa}/4)\log(K_2/\epsilon)$ *and*
- $\mathrm{n_v} \geq (24/\epsilon^2)(\log(1+\kappa))^2\log(2/\eta)$,

where K_1 is defined in Corollary 1 and $K_2 = \|z\|^2(\kappa-1)(\sqrt{2\kappa-1}-1)^2/\sqrt{2\kappa-1}$, we have

$$\Pr\left[|\log p(z \mid \xi) - \Gamma| \leq \epsilon(n+1)\right] \geq 1-\eta, \tag{2.19}$$

where $\Gamma = -\Gamma_1 - \Gamma_2 - \frac{n}{2}\log(2\pi)$, Γ_1 is the output of SLQ with parameters m_1 and $\mathrm{n_v}$, and Γ_2 is the output of the Lanczos Quadrature method for approximating $z^\top S(\xi)^{-1}z$ with m_2 steps of Lanczos and scaled by $\|z\|^2$.

Proof. To prove the Corollary, we obtain bounds for the two quantities Γ_1 and Γ_2. We bound the log-determinant term Γ_1 obtained by SLQ using Corollary 1. Bounds of Γ_2, the Lanczos quadrature approximation of $z^\top S(\xi)^{-1}z$, can be computed using Theorem 2 as follows. We again need to choose a smaller ellipse E_ρ where the function is analytic, since the function $f(t) = t^{-1}$ also goes to infinity near singularity. We set $\alpha = \kappa/(\kappa-1)$, then $\rho = (\kappa+\sqrt{2\kappa-1})/(\kappa-1)$ and $\rho^2-1 = (4\sqrt{2\kappa-1})/(\sqrt{2\kappa-1}-1)^2$. For the inverse function, the maximum must occur on the real line, particularly at $-\alpha$ for $g(z)$ or at $\lambda_{\min}/2$ for $f(z)$, so, $M_\rho = 2/\lambda_{\min}$. Then,

$$E(f) = \frac{(\kappa-1)(\sqrt{2\kappa-1}-1)^2}{\sqrt{2\kappa-1}\rho^{2m}}.$$

We will have a scaling $\|z\|^2$. We get the bounds by setting $\|z\|^2E(f) \leq \epsilon$. $\square$

We can also compute the error bounds for approximating the trace of matrix inverse by SLQ using the above proof.

Schatten p-norms estimation

When estimating the nuclear and Schatten p-norms, we encounter the following issue when approximating the square root function. *In order to obtain strong theoretical results (exponential convergence) for a given function $f(t)$, the function must be analytic in the spectrum interval. However, the square root function is non-differentiable at $t = 0$.* This will be a major stumbling block for rank-deficient matrices since the interval of eigenvalues now contains zero.

Shifting the Spectrum: To overcome the issue, we propose the following remedy, which is based on the key observation, proper to the computation of nuclear norm, that the small and zero singular values do not contribute much to the norm itself. In other words, the nuclear norm of a matrix depends mainly on the top singular values.

The idea is then to shift the spectrum of the matrix by a small $\delta > 0$ such that no eigenvalues of the matrix A are equal to zero. That is, we replace A by $A + \delta I$, such that the eigenvalues of the new shifted matrix are $\lambda_i + \delta$. For the square root function, the error is given by,

$$\sqrt{\lambda_i + \delta} - \sqrt{\lambda_i} = \frac{\delta}{\sqrt{\lambda_i + \delta} + \sqrt{\lambda_i}}.$$

Hence, the error in the large eigenvalues will be small. The error in the nuclear norm will be

$$\sum_{i=1}^{n} \sqrt{\lambda_i + \delta} - \sum_{i=1}^{n} \sqrt{\lambda_i} = \sum_{i=1}^{n} \frac{\delta}{\sqrt{\lambda_i + \delta} + \sqrt{\lambda_i}}. \tag{2.20}$$

For the shifted matrix, the eigenvalues will be in the interval $[\delta, \lambda_{\max} + \delta]$. Now, theorem 1 holds in this interval (square root function will be positive) and we can obtain the approximation error bounds. The error due to shifting is small, and can also be corrected using the Taylor series expansion of the square root function (details omitted). Algorithm 2 will be better suited for nuclear norm estimation.

Bounds for Schatten p-norms To summarize, if the input matrix has full rank ($\lambda_{\min} > 0$), then Theorem 1 is directly applicable, since the square root function is positive and analytic in the interval. For rank deficient matrices (has zero singular values), we will encounter the above problem, and we need to shift the spectrum by δ. From (2.20), we can upper bound the error due to shifting by $n\sqrt{\delta}$. Thus, the shift δ is chosen such that this error due to shifting is at most $\epsilon\|X\|_p^p$. Here, the value of $\|X\|_p$ can be taken to be roughly poly(n). We can then compute $\|X\|_p$ of the shifted matrix using SLQ. We have the following general result.

Corollary 3 *Given $\epsilon, \eta \in (0,1)$, a matrix $X \in \mathbb{R}^{d\times n}$ with its singular values in $[\sigma_{\min}, \sigma_{\max}]$, we consider the SPD matrix $A = X^TX$ (not formed explicitly) with its condition number $\kappa = \sigma_{\max}^2/\sigma_{\min}^2$, for SLQ parameters:*

- *$m \geq (\sqrt{\kappa}/4)\log(K_3/\epsilon)$ number of Lanczos steps, and*
- *$\mathrm{n_v} \geq (24/\epsilon^2)\log(2/\eta)$ number of starting vectors,*

where $K_3 = \frac{\sigma_{\min}^2(\kappa+1)^{p/2}(\kappa^2-1)}{\sqrt{\kappa}}$, we have

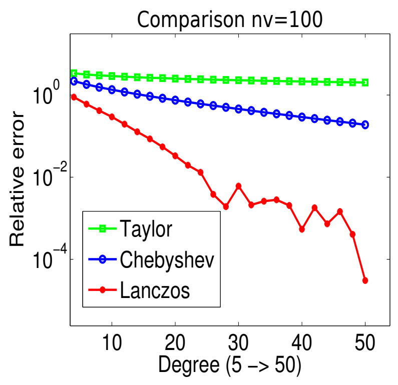
$$\Pr\left[\left|\|X\|_p^p - \Gamma^p\right| \leq \epsilon\|X\|_p^p\right] \geq 1-\eta, \tag{2.21}$$

where Γ is the output of the Stochastic Lanczos Quadrature method for Schatten p norm computation.

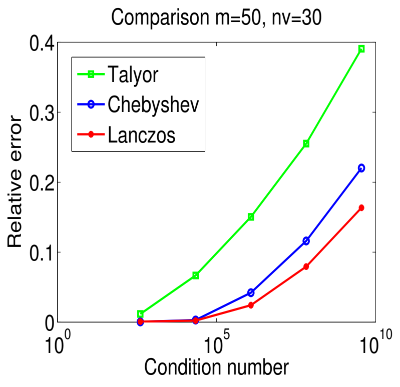
Proof. Theorem 1 gives the above error bound. We consider the function $f(t) = t^{p/2}$ applied to $A = X^TX$ (not formed explicitly). We can also consider the G-K-B algorithm. We choose $\rho = (\sqrt{\kappa}+1)/(\sqrt{\kappa}-1)$ as in the theorem since $g(t)$ is analytic inside the ellipse E_ρ. Then, $m_f = \sigma_{\min}^p$ and $M_\rho = \left(\sigma_{\max}^2 + \sigma_{\min}^2\right)^{p/2}$, since the maximum occurs at the right end of the ellipse. Substituting these values in the theorem, we get the above result. $\square$

2.5 Numerical Experiments

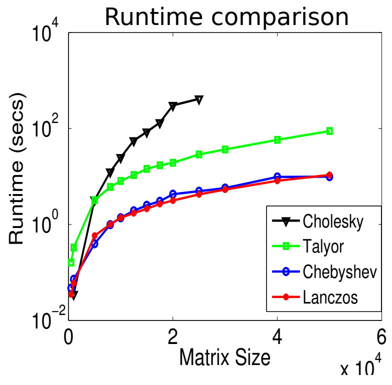
Log-determinant computation: In this section, we first present four examples to illustrate the performance of the SLQ method for logdet computation, and compare the performance against other related stochastic methods.



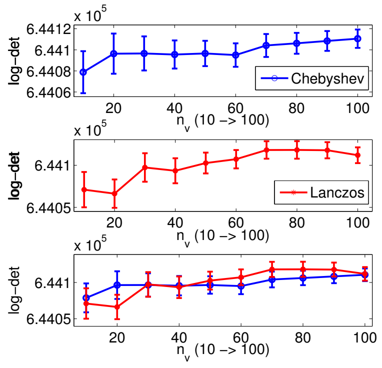
(a)



(b)



(c)



(d)

Figure 2.1: Performance comparisons between SLQ, Chebyshev and Taylor series expansions.

In Figure 2.1(a), we compare the relative errors obtained by the SLQ method for different degrees chosen, and compare it against the stochastic Chebyshev [17] (implemented by the authors) and the stochastic Taylor series expansions method [59]. We consider the sparse matrix `california` (a graph Laplacian matrix) of size 9664×9664, nnz $\approx 10^5$ and $\kappa \approx 5 \times 10^4$ from the SuiteSparse matrix collection [66]. The number of starting vectors $\mathrm{n_v} = 100$ in all three cases. The figure shows that our method is superior in accuracy compared to the other two methods. With just a degree of around 50, we get 4 digits of accuracy, while Chebyshev expansions give only 1-2 digits of accuracy and Taylor series expansions are very inaccurate for such low m.

Next, we evaluate the performance of our method with respect to the condition number of the matrix. We consider a Hadamard matrix H of size 8192 and form the test matrix as $HDH^\top$, where D is a diagonal matrix with entries such that the desired condition number is obtained. Figure 2.1(b) plots the relative errors obtained by the three stochastic methods for the log-determinant estimations of the matrices with different condition numbers. The degree and the number of starting vectors used in all three cases were $m = 50$ and $\mathrm{n_v} = 30$.

In the third experiment, we compare the runtime of the three algorithms for log-determinant estimation of large sparse matrices. The matrices have used 10% nonzeros in each row. An example Matlab code is the following: `N=20000; rho = 10/N; A = sprand(N,N,rho); A = A'*A + lmin*speye(N)`. We also include the runtime for the Cholesky decomposition. For a fair comparison, we chose $m = \sqrt{\kappa}$ for the Chebyshev method, $m = \sqrt{\kappa}/2$ for SLQ and $m = 4\sqrt{\kappa}$ for Taylor series (will be less accurate since we need $m \approx \kappa$ for similar accuracy). Figure 2.1(c) plots the runtime of the four algorithms for different matrix sizes. We observe that the runtime of the SLQ method is equal to or less than the runtime of the Chebyshev method. Note also that, both Chebyshev and Taylor methods require computation of the extreme eigenvalues. The relative errors we obtained by SLQ in practice are also lower than that obtained by the Chebyshev method. These two methods are both significantly faster than the one based on Cholesky.

For very large matrices ($\sim 10^6$ and above), it is impractical to compute the exact log-determinants. To gauge the approximation quality, we approximate the estimator variance by using sample variance and show the standard errors. Figure 2.1(d) plots the log-determinants estimated and the error bars obtained for different number of starting vectors for the matrix `webbase-1M` (Web connectivity matrix) of size $10^6 \times 10^6$ obtained from SuiteSparse database [66]. For Lanczos Quadrature, we chose degree $m = 30$, and for Chebyshev $m = 60$. The width of the error bars gives us a rough idea of how close the estimation might be to the trace of $f(A)$ approximated by the respective methods.

Nuclear norm estimation: Next, let us consider the estimation of the nuclear norm, and employ our SLQ algorithm for the nuclear norm estimation of real datasets.

Table 2.1 lists the approximate nuclear norm estimated by our method for a set of matrices from various applications. All matrices were obtained from SuiteSparse[66] and

Table 2.1: Estimation of the sum of singular values of various matrices

Matrices	size n	m	Exact Sum	Estimated Sum	Time (secs)	SVD time
Erdos992	6100	40	3292.06	3294.5	1.05	876.2 secs
deter3	7047	30	16518.08	16508.46	1.62	1.3 hrs
California	9664	100	3803.74	3803.86	8.32	4.17 mins
FA	10617	150	1306.79	1312.8	23.13	1.5 hrs
qpband	20000	60	26708.14	26710.1	0.35	2.9 hrs

are sparse. We increment the degree m until we achieve 3-4 digit accuracy. The degree used and the approximate sum obtained are listed in the table along with the exact sum and the time taken (averaged over 5 trials) by our algorithm. In all experiments, the number of starting vectors $\mathrm{n_v} = 30$. In addition, we also list the time taken to compute only the top 2000 singular values of each matrices (computed using MATLAB's `svds` function which relies on ARPACK) in order to provide a rough illustration of the potential computational gain of our algorithm over partial SVD.

Maximum Likelihood estimation for GRF: We now test our method for maximum likelihood (ML) estimation of Gaussian Random Fields (GRF). To illustrate the use of log-determinant calculation in GRFs, we simulate one such field by using the Wendland covariance function [45] with smoothness $q = 0$ on a 900×1200 grid ($n = 1.08 \times 10^6$); see fig. 2.2(a). To better demonstrate the fine details of this highly non-smooth data, we have zoomed into the middle 300×400 grid and shown only this part. Next, we randomly sample ten percent of the data and used them to estimate the length scale of the function. These training data are the non-white pixels in fig. 2.2(b). We compute a local log-likelihood curve (as in (2.1)) shown in fig. 2.2(c) using SLQ with different values for the hyperparameter, which suggests a peak at 50. That is, MLE estimates using SLQ suggests the hyperparameter value to be 50. This coincides with the true value used for simulation. The log-determinants therein were computed using 100 Lanczos steps and 100 random vectors. Because the covariance matrix is multilevel Toeplitz, the matrix-vector multiplications were carried out through circulant embedding followed by FFT, which resulted in an $O(n \log n)$ cost [56]. With the estimated length scale, we perform a prediction calculation for the rest of the data (white pixels in fig. 2.2(b)) and show the predicted values, together with the ten percent used for

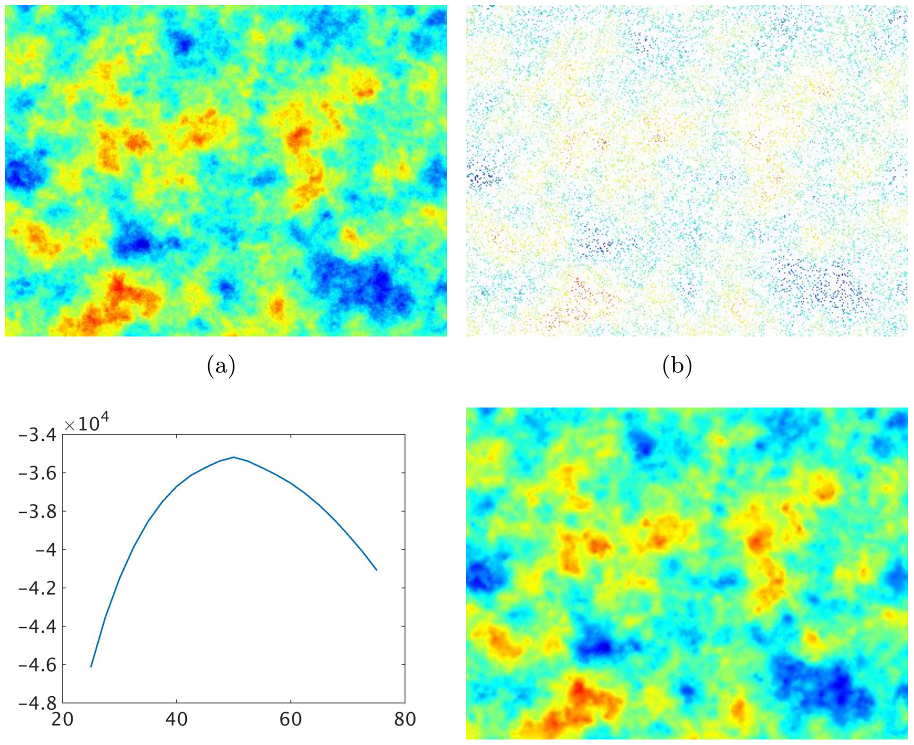


Figure 2.2: Estimation and prediction for a Gaussian random field.

training, in fig. 2.2(d). We observe that the pattern obtained from the predicted values appears quite similar to the original data pattern. The relative difference between fig. 2.2(a) and fig. 2.2(d) is 0.27.

Spatial Analysis using GMRF for CO_2 data: In the final experiment, we consider the Gaussian Markov Random field (GMRF) [45] parameter estimation problem for real spatial data with missing entries. We use a global dataset of column-integrated CO_2 obtained from `http://niasra.uow.edu.au/cei/webprojects/UOW175995.html`. The values of column integrated CO_2 are on a grid of $1.25°$ longitude by $1°$ latitude, which results in a total of $288 \times 181 = 52,128$ grid cells (matrix size) on the globe. The dataset has $26,633$ observations. We assume GRMF model for the data and use MLE to predict the remaining (missing) values. For the GMRF field, we considered the spatial autoregressive model, i.e., the precision matrix is defined as $G(\xi) = \xi^4 C + \xi^2 G_1 + G_2$, where matrices C, G_1 and G_2 define the neighborhood (four, eight and 16 neighbors,

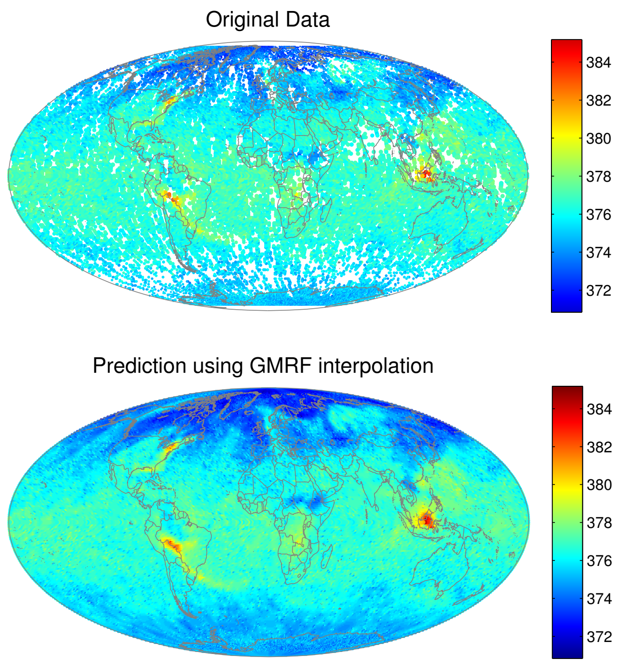


Figure 2.3: GMRF interpolation for CO_2 data.

respectively) and are sparse [45]. We obtain ML estimates using the SLQ method to choose the optimal parameter ξ. That is, we sweep through a set of values for ξ and estimate the log-likelihood for the data given by $\log p(z \mid \xi) = \log\det G(\xi) - z^\top G(\xi) z - \frac{n}{2}\log(2\pi)$, and determine the parameter ξ that maximizes the log-likelihood. Figure 2.3(top) shows the sparse observations of the CO_2 data across the globe. The GMRF interpolation with the parameter $\xi = 0.2$ is given in fig. 2.3(bottom).

Chapter 3

Numerical rank estimation

3.1 Introduction

In many machine learning, data analysis, scientific computations and signal processing applications, the high dimensional data encountered generally have intrinsically low dimensional representations. A widespread tool used in these applications to exploit this low dimensional nature of data is the Principal Component Analysis (PCA) [23]. Other well known techniques such as randomized low rank approximations [67, 7] and low rank subspace estimations [68] also exploit the ubiquitous low rank character of data. However, a difficulty with these approaches that is well-recognized in the literature is that, it is not known in advance how to select the reduced rank k. This problem is aggravated in the applications of algorithms such as online PCA [69], stochastic approximation algorithms for PCA [70] and subspace tracking [68], where the dimension of the subspace of interest changes frequently.

The rank estimation problem also arises in many useful methods employed in fields such as machine learning for example, where the data matrix $X \in \mathbb{R}^{d\times n}$ is replaced with a factorization of the form $UV^\top$, where $U \in \mathbb{R}^{d\times k}$ and $V \in \mathbb{R}^{n\times k}$. In these methods, the original problem is solved by fixing the rank of the unknown matrix to a preselected value k [71]. Similar rank estimation problems are encountered in reduced rank regression, when solving numerically rank deficient linear systems of equations [72], and in numerical methods for eigenvalue problems that are used to compute the dominant subspace of a matrix, for e.g., subspace iteration.

In the most common situation, the rank k required as input in the above applications is typically selected in an ad-hoc way. This is because standard rank estimation methods in the existing literature rely on expensive matrix factorizations such as the QR [73], LDL^T or SVD [56]. Other methods also assume certain asymptotic behavior such as normal responses, for the input matrices [74, 75]. Many of the rank estimation methods proposed in the literature focus on specific applications, e.g., in econometrics and statistics [74], statistical signal processing [76, 75], reduced-rank regression [77], estimating the dimension of linear systems [78] and others.

Powerful and inexpensive tools from computational linear algebra can be developed to estimate the approximate ranks of large matrices. In this chapter, we present examples of such methods. These methods require only matrix-vector products ('matvecs') and are inexpensive compared to traditional methods. In addition, they do not make any particular statistical, or asymptotic behavior assumptions on the input matrices. Since the data matrix can be approximated in a low dimensional subspace, the only assumption is that there is a set of relevant eigenvalues in the spectrum that correspond to the eigenvectors that span this low dimensional subspace, and that these are well separated from the noise-related eigenvalues.

3.2 Numerical Rank

The *approximate rank* or *numerical rank* of a general $d \times n$ matrix X is often defined by using the closest matrix to X in the 2-norm sense. Specifically, the *numerical ϵ-rank* r_ϵ of X, with respect to a positive tolerance ϵ is

$$r_\epsilon = \min\{rank(B) : B \in \mathbb{R}^{d\times n}, \|X - B\|_2 \leq \epsilon\}, \tag{3.1}$$

where $\|\cdot\|_2$ refers to the 2-norm or spectral norm. This standard definition has been often used in the literature, see, e.g., [79, §2], [72, §3.1]. The ϵ-rank of a matrix X is equal to the number of columns of X that are linearly independent for any perturbation of X with norm at most the tolerance ϵ. The definition is pertinent when the matrix X was originally of rank $r_\epsilon < \min\{d, n\}$, but its elements have been perturbed by some small error or when the relevant information of the matrix lies in a lower dimensional subspace. The input (perturbed) matrix is likely to have full rank but it can be well

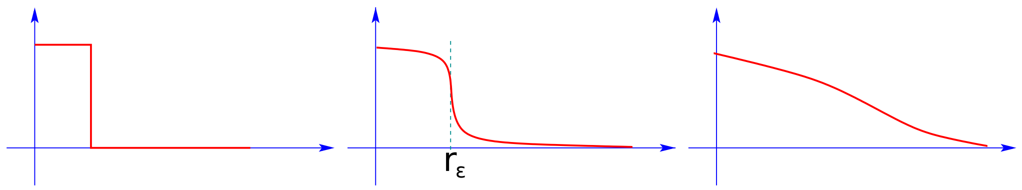


Figure 3.1: Three different scenarios for eigenvalues of PSD matrices

approximated by a rank-r_ϵ matrix.

The singular values of X, with the numerical rank r_ϵ must satisfy

$$\sigma_{r_\epsilon} > \epsilon \geq \sigma_{r_\epsilon+1}. \tag{3.2}$$

It should be stressed that the numerical rank r_ϵ is practically significant only when there is a *well-defined gap* between σ_{r_ϵ} and $\sigma_{r_\epsilon+1}$ or in the corresponding eigenvalues of the matrix $X^\top X$ (also highlighted in [79, 72]). It is important that the rank r_ϵ estimated be robust to small variations of the threshold ϵ and the eigenvalues of X, and this can be ensured only if there is a good gap between the relevant eigenvalues and those close to zero associated with noise.

For illustration, consider the three situations shown in Figure 3.1. The curves shown in the figure are continuous renderings of simple plots showing eigenvalues λ_i (y-axis) of three different kinds of PSD matrices as a function of i (x-axis). The leftmost case is an idealized situation where the matrix is exactly low rank and the rank is simply the number of positive eigenvalues. The situation depicted in the middle plot is common in situations where the same original matrix of rank say r_ϵ is perturbed by some small noise or error. In this case, the approximate rank r_ϵ of the matrix can be estimated by counting the number of eigenvalues larger than a threshold value ϵ that separates the spectrum into two distinct well separated sets. The third curve shows a situation when the perturbation or the noise added overwhelms the small relevant eigenvalues of the matrix, so there is no clear separation between relevant eigenvalues and the others. In this case there is no clear-cut way of recovering the original rank.

The matrices related to the applications discussed earlier typically belong to the second situation and we shall consider only these cases in this paper. Unless there is an advance knowledge of the noise level or the size of the perturbation, we will need a way to estimate the gap between the small eigenvalues to be neglected and the others. This issue of selecting the gap, i.e., the parameter ϵ has been addressed by a few articles in the signal processing literature, e.g., see [76]. Section 3.4 describes a method to determine this gap and to choose a value for the threshold ϵ based on the plot of spectral density of the matrix. Once the threshold ϵ is selected, the rank can be estimated by counting the number of eigenvalues of A that are larger than ϵ.

3.3 Polynomial filters

In the previous chapter, we saw how the trace of an eigen-projector can be used to compute the numerical rank of a matrix. We also saw that a projector P can be viewed as a step function applied to the matrix. We can use the SLQ method to compute the trace of a step function. In this chapter, the projector $P = h(A)$ in (2.2) is approximated by $\psi_m(A)$, where $\psi_m(t)$ is a 'filter' polynomial. In practice, we only need $\psi_m(t)$ to transform the larger relevant eigenvalues into a value close to one and the smaller eigenvalues to a value close to zero. We first consider a simple filter based on Hermite interpolation, which has a number of advantages relative to the more common Chebyshev filter.

3.3.1 The McWeeny filter

The McWeeny transform [80] has been used in solid-state physics to develop 'linear-scaling' methods. It starts by scaling and shifting the matrix so that its eigenvalues are in the interval [0, 1]. This can be achieved by simply defining $B = A/\lambda_1$, where the largest eigenvalue λ_1 can be inexpensively computed with a few steps of the Lanczos algorithm [56].

The extended McWeeny filters have been studied in a different context [81]. A systematic way of generating them is through interpolation in the Hermite sense, using two integer parameters m_0, m_1 that define the degree of matching or smoothness at two points τ_0 and τ_1 respectively. In the following, we denote by $\Theta_{[m_0,m_1]}$ the interpolating (Hermite) polynomial that satisfies the following conditions:

$$\begin{aligned}
\Theta_{[m_0,m_1]}(\tau_0) = 0; \Theta'_{[m_0,m_1]}(\tau_0) = \cdots = \Theta^{(m_0-1)}_{[m_0,m_1]}(\tau_0) = 0 \\
\Theta_{[m_0,m_1]}(\tau_1) = 1; \Theta'_{[m_0,m_1]}(\tau_1) = \cdots = \Theta^{(m_1-1)}_{[m_0,m_1]}(\tau_1) = 0.
\end{aligned}$$

Thus, $\Theta_{[m_0,m_1]}$ has degree $m_0 + m_1 - 1$ and the two parameters m_0 and m_1 define the degree of smoothness at the points τ_0 and τ_1 respectively. The polynomials $\Theta_{[m_0,m_1]}$ can be easily determined by standard finite difference tables. The paper [81] also gives a closed form expression for $\Theta_{[m_0,m_1]}$ when $\tau_0 = -1$ and $\tau_1 = 1$:

$$\Theta_{[m_0,m_1]} = \frac{\int_{-1}^{t} (1-s)^{m_1-1}(1+s)^{m_0-1}\, ds}{\int_{-1}^{1} (1-s)^{m_1-1}(1+s)^{m_0-1}\, ds}. \tag{3.3}$$

Furthermore, when $m_0 + m_1 > 2$ (at least 3 conditions imposed), the function has an inflexion point at :

$$t = \frac{m_0 - m_1}{m_0 + m_1 - 2}.$$

When translated back to the interval $[0,\ 1]$ this point becomes $(t+1)/2 = (m_0 - 1)/(m_0 + m_1 - 2)$. See more details in [2].

3.3.2 Chebyshev filters

Chebyshev polynomials are commonly used to expand the step function h, i.e., $h(t)$ is approximately expanded as :

$$h(t) \approx \sum_{k=0}^{m} \gamma_k T_k(t), \tag{3.4}$$

where each T_k is the k-degree Chebyshev polynomial of the first kind, formally defined as $T_k(t) = \cos(k \cos^{-1}(t))$. Since Chebyshev polynomials are based on the interval $[-1,\ 1]$ we will assume first that A has eigenvalues between -1 and 1. Let a, b such that $-1 \le a < b \le 1$. The expansion coefficients γ_k for the polynomial to approximate a step function $h(t)$, which takes value 1 in $[a,\ b]$ and 0 elsewhere, are known:

$$\gamma_k = \begin{cases} \frac{1}{\pi}(\cos^{-1}(a) - \cos^{-1}(b)) & :\ k = 0, \\ \frac{2}{\pi}\left(\frac{\sin(k\cos^{-1}(a)) - \sin(k\cos^{-1}(b))}{k}\right) & :\ k > 0 \end{cases}.$$

Once the γ_k's are known, the desired Chebyshev expansion of the projector P will be given by: $P \approx \psi_m(A) = \sum_{k=0}^{m} \gamma_k T_k(A)$.

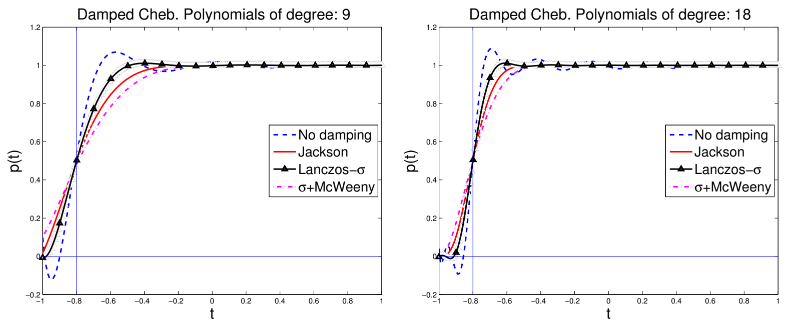


Figure 3.2: Different ways of damping Gibbs oscillations for Chebyshev approximation.

Damping and other practicalities

Expanding discontinuous functions using Chebyshev polynomials results in oscillations known as *Gibbs Oscillations* near the boundaries. To reduce or suppress these oscillations, damping multipliers are often added. That is, each γ_k in the expansion above is multiplied by a smoothing factor g_k^m which will tend to be quite small for the larger values of k that correspond to the highly oscillatory terms in the expansion. Jackson smoothing is a popular damping approach used in the whereby the coefficients g_k^m are given by the formula

$$g_k^m = \frac{\sin(k+1)\alpha_m}{(m+2)\sin(\alpha_m)} + \left(1 - \frac{k+1}{m+2}\right)\cos(k\alpha_m),$$

where $\alpha_m = \frac{\pi}{m+2}$. More details on this expression can be seen in [58]. Not as well known is another form of smoothing proposed by Lanczos [82, Chap. 4] and referred to as σ-smoothing. It uses the following simpler damping coefficients, called σ factors by the author:

$$\sigma_0^m = 1;\ \sigma_k^m = \frac{\sin(k\theta_m)}{k\theta_m}, k = 1, \ldots, m \text{ with } \theta_m = \frac{\pi}{m+1}.$$

The damping factors are small for larger values of k and this has the effect of reducing the oscillations. The Jackson polynomials have a much stronger damping effect on these last terms than the Lanczos σ factors. For example the very last factors, and their approximate values for large m's, are in each case:

$$g_m^m = \frac{2\sin^2(\alpha_m)}{m+2} \approx \frac{2\pi^2}{(m+2)^3}; \qquad \sigma_m^m = \frac{\sin(\theta_m)}{m\theta_m} \approx \frac{1}{m}.$$

Jackson coefficients tend to over-damp the oscillations at the expense of sharpness of the approximation. The Lanczos smoothing can be viewed as an intermediate form of damping between no damping and Jackson damping. A comparison of the three forms of damping is shown in Figure 3.2. To the three forms of damping (no-damping, Jackson, σ-damping), we have added a fourth one which consists of compounding the degree 3 McWeeny filter with the Chebyshev polynomials. In the numerical experiments, we have used Lanczos σ damping.

An important practical consideration is that we can economically compute vectors of the form $T_k(B)v$, where B is as defined as

$$B = \frac{A - cI}{d} \qquad \text{with} \quad c = \frac{\lambda_1 + \lambda_n}{2}, \quad d = \frac{\lambda_1 - \lambda_n}{2}. \tag{3.5}$$

since the Chebyshev polynomials obey a three term recurrence. That is, we have $T_{k+1}(t) = 2tT_k(t) - T_{k-1}(t)$ with $T_0(t) = 1, T_1(t) = t$. As a result, the following iteration can be used to compute $w_k = T_k(B)v$, for $k = 0, \cdots, \cdots, m$

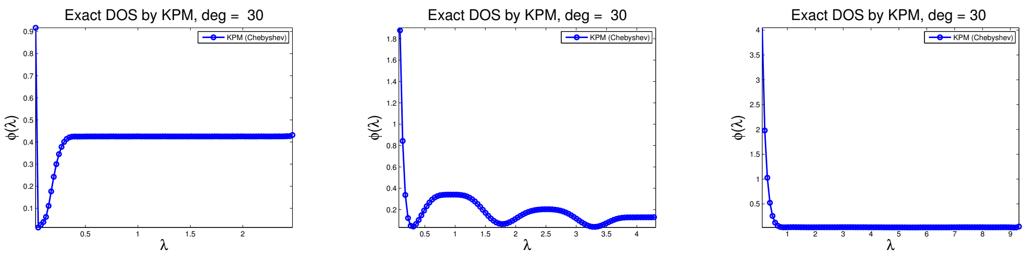
$$w_{k+1} = 2Bw_k - w_{k-1}, k = 1, 2, \ldots, m-1, \tag{3.6}$$

with $w_0 = v;\ w_1 = Bv$.

Remark 2 *Note that if the matrix B is of the form $B = Y^\top Y$, where Y is a linear transformation of the data matrix X using the mapping* (3.5)*, then we need not compute the matrix B explicitly since the only operations that are required with the matrix B are matrix-vector products.*

3.4 Threshold selection

The method we described so far requires a threshold parameter ϵ that separates the small eigenvalues, those assumed to be perturbations of the zero eigenvalue, from the



(A) (B) (C)

Figure 3.3: Exact DOS plots for three different types of matrices.

relevant larger eigenvalues that contribute to the rank. We now describe a method to select ϵ based on the spectral density, which was introduced in the previous chapter.

3.4.1 DOS plot analysis

For motivation, let us first consider a matrix that is exactly of low rank and observe the typical shape of its DOS function plot. As an example we take an $n \times n$ PSD matrix with rank $k < n$, that has k eigenvalues uniformly distributed between 0.2 and 2.5, and whose remaining $n-k$ eigenvalues are equal to zero. An approximate DOS function plot of this low rank matrix is shown in figure 3.3(A). The DOS is generated using KPM, with a degree $m = 30$ where the coefficients μ_k are estimated using the exact trace of the Chebyshev polynomial functions of the matrix. Jackson damping is used to eliminate oscillations in the plot. The plot begins with a high value at zero indicating the presence of a zero eigenvalue with high multiplicity. Following this, it quickly drops to almost a zero value, indicating a region where there are no eigenvalues. This corresponds to the region just above zero and below 0.2. The DOS increases at 0.2 indicating the presence of new eigenvalues. Because of the uniformly distributed eigenvalues between 0.2 and 2.5, the DOS plot has a constant positive value in this interval.

To estimate the rank k of this matrix, we can count the number of eigenvalues in the interval $[\epsilon,\ \lambda_1] \equiv [0.2, 2.5]$ by integrating the DOS function over the interval. The value $\lambda_1 = 2.5$ can be replaced by an estimate of the largest eigenvalue. The initial value $\epsilon = 0.2$ can be estimated as *the point immediately following the initial sharp drop observed* or *the mid point of the valley.* For low rank matrices such as the one considered here, we should expect to see this sharp drop followed by a valley. The cutoff point between zero eigenvalues and relevant ones should be at the location where the curve ceases to decrease, i.e., the point where the derivative of the spectral density function becomes zero (local minimum) for the first time. Thus, the threshold ϵ can be selected as

$$\epsilon = \min\{t : \phi'(t) = 0, \lambda_n \leq t \leq \lambda_1\}. \tag{3.7}$$

For more general numerically rank deficient matrices, the same idea based on the DOS plot can be employed to determine the approximate rank. Defining a cut-off value between the relevant singular values and insignificant ones in this way works when there is a gap in the matrix spectrum. This corresponds to matrices that have a cluster of eigenvalues close to zero, which are zero eigenvalues perturbed by noise/errors, followed by an interval with few or no eigenvalues, a gap, and then clusters of relevant eigenvalues, which contribute to the approximate rank. Two types of DOS plots are often encountered depending on the number of relevant eigenvalues and whether they are in clusters or spread out wide.

Figures 3.3(B) and (C) show two sample DOS plots which belong to these two categories, respectively. Both plots were estimated using KPM and the exact trace of the matrices, as in the previous low rank matrix case. The middle plot (figure 3.3(B)) is a typical DOS curve for a matrix which has a large number of eigenvalues related to noise which are close to zero and a number of larger relevant eigenvalues which are in a few clusters. The spectral density curve displays a fast decrease after a high value near zero eigenvalues due to the gap in the spectrum and the curve increases again due the appearance of large eigenvalue clusters. In this case, we can use equation (3.7) to estimate the threshold ϵ.

In the last DOS plot on figure 3.3(C), the matrix has again a large number of eigenvalues related to noise which are close to zero, but the number of relevant eigenvalues is smaller and these eigenvalues are spread farther and farther apart from each other as their values increase, (as for example when $\lambda_i = K(n-i)^2$.) The DOS curve has a similar high value near zero eigenvalues and displays a sharp drop, but it does not increase again and tends to hover near zero. In this case, there is no valley or local minimum, so the derivative of the DOS function may not reach the value zero. The best we can do here is detect a point at which the derivative exceeds a certain negative

Algorithm 2 Numerical rank estimation by polynomial filtering

```
**Input:** An $n \times n$ symmetric PSD matrix $A$, $\lambda_1$ and $\lambda_n$ of $A$, and number $\mathrm{n_v}$ of sample vectors to be used.
**Output:** The numerical rank $r_\epsilon$ of $A$.
1. Generate the random starting vectors $v_l : l = 1, \ldots \mathrm{n_v}$, such that $\|v_l\|_2 = 1$.
2. Transform the matrix $A$ to $B = A/\lambda_1$, choose degree $m$ for DOS and form the matvecs
       $B^k v_l : l = 1, \ldots, \mathrm{n_v}, k = 0, \ldots, m.$
3. Form the scalars $v_l^\top T_k(B) v_l$ using the above matvecs and obtain the DOS $\tilde{\phi}(t)$.
4. Estimate the threshold $\epsilon$ from $\tilde{\phi}(t)$ using eq. (3.8).
5. **McWeeny filter:** Estimate $m_1$ and $\tau_1$ from $\epsilon$. Compute $\Theta_{[m_0,m_1]} v_l$ using the above matvecs (compute additional matvecs if required). Estimate the numerical rank $r_\epsilon$.
   **Chebyshev filter:** Compute the degree $m$ and estimate the coefficients $\gamma_k$ for the interval $[\epsilon, \lambda_1]$. Compute the numerical rank $r_\epsilon$ using the above matvecs.
```

value, for the first time, indicating a significant slow-down of the initial fast decrease. In summary, the threshold ϵ for all three cases can be selected as

$$\epsilon = \min\{t : \phi'(t) \geq tol, \lambda_n \leq t \leq \lambda_1\}. \tag{3.8}$$

Our sample codes use $tol = -0.01$ which seems to work well in practice.

When the input matrix does not have a large gap between the relevant and noisy eigenvalues (when numerical rank is not well-defined), the corresponding DOS plot of that matrix will display similar behavior as the plot in figure 3.3(C), except the plot does not go to zero. That is, the DOS curve will have a similar knee as in figure 3.3(C).

3.4.2 Algorithm

Algorithm 2 describes our approach for estimating the approximate rank r_ϵ by the two polynomial filtering methods discussed earlier.

Computational cost. The core of the computation in the two rank estimation methods is the matrix vector product of the form $T_k(A)v_l$ or in general $A^k v_l$ for $l = 1, \ldots, \mathrm{n_v}, k = 0, \ldots, m$ (step 3). Note that no matrix-matrix products or factorizations are required. In addition, the matrix vector products $A^k v_l$ computed during the estimation of the threshold, for the spectral density, can be saved and reused for the rank estimation, and so the related matrix-by-vector products are computed only once. All remaining steps of the algorithm are essentially based on these 'matvec' operations.

For an $n \times n$ dense symmetric PSD matrix, the computational cost of Algorithm 2 is $O(n^2 m \mathrm{n_v})$. For a sparse matrix, the computation cost will be $O(\mathrm{nnz}(A) m \mathrm{n_v})$, where $\mathrm{nnz}(A)$ is the number of nonzero entries of A. This cost is linear in the number of nonzero entries of A for large matrices and it will be generally quite low when A is very sparse, e.g., when $\mathrm{nnz}(A) = O(n)$. These methods are very inexpensive compared to methods that require matrix factorizations such as QR or SVD.

Remark 3 *In some of the rank estimation applications, it is perhaps required to estimate the corresponding eigenpairs or the singular triplets of the matrix, after the approximate rank estimation. These can be easily computed using a Rayleigh-Ritz projection type methods, exploiting again the vectors $A^k v_l$ generated for estimating the rank.*

On the convergence. The convergence analysis of the trace estimator was discussed in the previous chapter. The best known convergence rate for (2.4) is $O(1/\sqrt{\mathrm{n_v}})$ for Hutchinson and Gaussian distributions, see Theorem 3.

Theoretical analysis for approximating a step function as in (2.2) is not straightforward since we are approximating a discontinuous function. Convergence analysis on approximating a step function is documented in [83]. A convergence rate of $O(1/m)$ can be achieved with any polynomial approximation [83]. However, this rate is obtained for point by point analysis (at the vicinity of discontinuity points), and uniform convergence cannot be achieved due to the Gibbs phenomenon.

Improved theoretical results can be obtained if we first replace the step function by a piecewise linear approximation, and then employ polynomial approximation. Article [81] shows that uniform convergence can be achieved using Hermite polynomial approximation (as in sec. 3.3.1) when the filter is constructed as a spline (piecewise linear) function. For example,

$$\psi(t) = \begin{cases} 0 & : for\ t \in [0, \epsilon_0) \\ \Theta_{[m_0.m_1]} & : for\ t \in [\epsilon_0, \epsilon_1) \\ 1 & : for\ t \in [\epsilon_1, 1] \end{cases} . \tag{3.9}$$

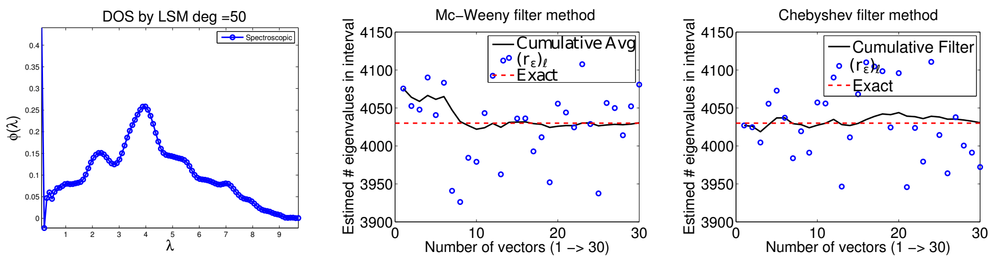


Figure 3.4: Numerical ranks estimated for the example `ukerbe1`.

It is well known that uniform convergence can be achieved with Chebyshev polynomial approximation if the function approximated is continuous and differentiable [57]. Further improvement in the convergence rate can be accomplished, if the step function is replaced by an analytic function, for example, $\psi(t)$ can be a shifted version of $\tanh(\alpha t)$ function. In this case, exponential convergence rate can be achieved with Chebyshev polynomial approximation [57]. However, such complicated implementations are unnecessary in practice. The bounds achieved for both the trace estimator and the approximation of step functions discussed above are too pessimistic, since in practice we can get accurate ranks for $m \sim 50$ and $\mathrm{n_v} \sim 30$.

3.5 Numerical experiments

We first illustrate the performance of the rank estimation techniques on a $5,981 \times 5,981$ matrix named `ukerbe1` from the AG-Monien group (the matrix is a Laplacian of an undirected graph), available in the SuiteSparse Matrix Collection [66] database. The performances of the Chebyshev Polynomial filter method and the extended McWeeny filter method for estimating the numerical rank of this matrix are shown in figure 3.4.

Figure 3.4 (Left) gives the spectral density plot obtained using Chebyshev polynomials of degree $m = 50$ and a number of samples $\mathrm{n_v} = 30$. Using this plot, the threshold ϵ estimated by the method described in subsection 3.4 was $\epsilon = 0.169$. Figure 3.4 (Middle) plots the numerical ranks estimated by the McWeeny filter method with 30 sample vectors. The degrees $[m_0, m_1]$ for the Hermite polynomials estimated were $[2, 54]$. In the plot, the circles indicate the approximate ranks estimated with the ℓth sample vectors and the dark line is the cumulative (running) average of these estimated approximate rank values. The average numerical rank estimated over 30 sample vectors was equal

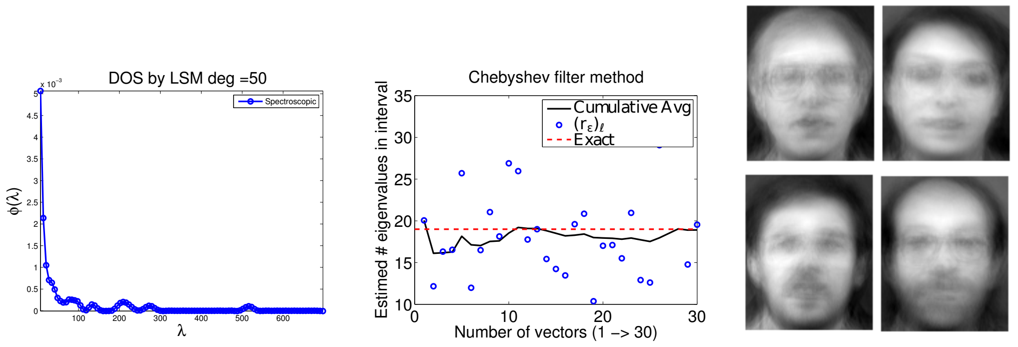


Figure 3.5: Rank estimation for the ORL dataset. Right: Eigenfaces.

to 4030.47. The exact number of eigenvalues above the threshold is 4030, indicated by the dotted line in the plot. Similarly, figure 3.4 (Right) plots the numerical ranks estimated by the Chebyshev filter method with $n_v = 30$. The degree for the Chebyshev polynomials was $m = 50$. The average numerical rank estimated over 30 sample vectors is 4030.57.

Timing Experiment : Here, we provide an example to illustrate how fast these methods can be. We consider a sparse matrix of size 1.25×10^5 called `Internet` from SuiteSparse database, with $\text{nnz}(A) = 1.5 \times 10^6$. The estimation of its rank by the Chebyshev filter method took only 7.18 secs on average (over 10 trials) on a standard $3.3GHz$ Intel-i5 machine. Computing the rank of this matrix by an approximate SVD, for example using the svds/eigs function matlab which relies on ARPACK, will be exceedingly expensive. It took around 2 hours to compute 4000 singular values of the matrix on the same machine. Methods based on rank-revealing QR factorizations or the standard SVD are not even possible for this problem on a standard workstation such as the one we used.

Eigenfaces: It is well known that face images lie in a low-dimensional linear subspace and the low rank approximation methods are widely used in applications such as face recognition. *Eigenfaces* is a popular method used for face recognition which is based on Principal Component Analysis (PCA). Such PCA based techniques require the knowledge of the dimension of the smaller subspace. Here, we demonstrate how our rank estimation methods can be combined with the randomized-SVD method [67] in

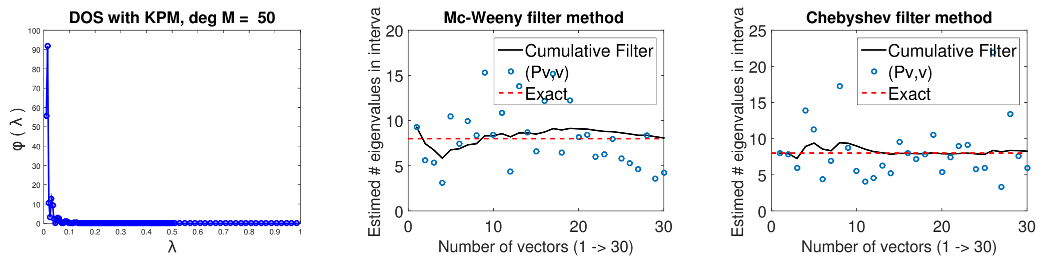


Figure 3.6: Estimation of the number of signals for the adaptive beamforming example.

the application of face recognition. As an illustration, we consider the ORL face dataset obtained from the AT&T Labs Cambridge database of faces. There are ten different images of each of 40 distinct subjects. The size of each image is 92×112 pixels, with 256 gray levels per pixel. So, the input matrix is of size 400×10304, which is formed by vectorizing the images. The matrix is mean centered (required for eigenfaces method) and scaled.

In figure 3.6 (left and middle plots) the DOS and the numerical rank are plotted for the ORL image matrix, both estimated using Chebyshev polynomials of degree $m = 50$ and $\mathrm{n_v} = 30$. The numerical rank estimated over 30 sample vectors was found to be 18.90. There are 19 eigenvalues above the threshold, estimated using (3.8) with $tol = -0.01$. The four images (on the right) in the figure are the eigenfaces of 4 individuals recovered using rank $k = 20$ (top 20 singular vectors) computed using the randomized SVD algorithm [67].

Estimation of the number of signals: The next application we will consider here comes from Signal Processing. The objective is to detect the number of signals q embedded in the noisy signals received by a collection of n sensors (equivalent to estimating the number of transmitting antennas). This can be achieved by finding the numerical rank of the corresponding sample covariance matrix of the received signals.

We consider $n = 1000$ element sensor array receiving $r = 8$ interference signals incident at angles $[-90^0, 90^0, -45^0, 45^0, 60^0, -30^0, 30^0, 0^0]$. The output signal $y(t)$ can be represented as

$$y(t) = \sum_{i=1}^{q} s_i(t)a_i + \eta(t) = As(t) + \eta(t), \tag{3.10}$$

where $A = [a_1(\theta_1), a_2(\theta_2), \ldots, a_q(\theta_q)]$ is an $n \times q$ mixing matrix, $s(t) = [s_1(t), s_2(t), \ldots, s_q(t)]$ an $q \times 1$ signal vector (signals sent from the transmitters) and $\eta(t)$ is a white noise vector, with the noise power set to $-10DB$. The covariance matrix $C = \mathbb{E}[y(t)y(t)^\top]$ is a numerically rank deficient matrix. That is, the matrix is a noisy version of a low rank q matrix. Hence, we can employ the rank estimation methods to estimate the numerical rank of this matrix, in turn estimating the number of signals q in the received signals.

Figure 3.6 (left) shows the spectral density obtained using Chebyshev polynomial of degree $m = 50$ and number of samples $n_{vec} = 30$. The threshold ϵ was estimated by the method described in the main paper using this spectral density plot. Figure 3.6 (middle) shows the estimated numerical rank by the extended Mc-Weeny filter method with 30 samples. The degree of Hermite polynomial estimated was $[m_0, m_1] = [2, 44]$. The average of approximate ranks estimated over 30 sample vectors was equal to 8.07. The actual count in the interval is 8 (we know there are 8 signals).

Similarly, figure 3.6 (right) shows the estimated numerical rank by the Chebyshev filtering method using degree $m = 50$ and $n_{vec} = 30$. The average of approximate ranks estimated over 30 sample vectors was equal to 8.25. Clearly, both the methods have accurately estimated the number of interference signals embedded in the received signals.

This problem of number of signals estimation is further discussed in the next chapter, where a new method is proposed for this estimation.

Chapter 4

Dimension estimation for Krylov approximation

4.1 Introduction

In many applications, for a given set of data observations, it is required to estimate the dimension of the principal (dominant) subspace of the covariance matrix associated with the observations [84, 74, 85, 2]. These observations can typically be modeled as high dimensional random quantities embedded in noise. In statistical signal and array processing, detection of the number of signals in the observations of array of passive sensors is a fundamental problem [84, 85], which can be posed as this dimension estimation problem. Similar estimation problems occur in many other fields such as chemo-metrics [86], econometrics and statistics [74], population genetics [87], and reduced rank regression models [77]. In modern data related applications, the observed data is typically high dimensional, but their relevant information lies in a low dimensional subspace. Low rank approximation is a popular tool used in such applications to reduce the data [23, 67, 7]. Determining the lower dimension (rank k) remains a principal problem in these applications too, as discussed in the previous chapter. Moreover, in many of these applications, once the dimension of the principal subspace (approximate rank) is estimated, it is also desired to obtain an approximation for this principal subspace, e.g., in principal component analysis (PCA) [23], subspace tracking and others. Krylov subspace based methods [88] are the most popular and effective methods used to compute approximations for the principal subspace, see [89, 90] for examples. Hence, we view this problem as dimension estimation for Krylov subspace approximation of covariance matrices.

The problem of estimating the rank or the dimension of the principal subspace has been studied in various fields, and a number of different methods have been proposed in the literature, as we saw in the previous chapter. However, most of these methods require computing the complete eigen-decomposition of the sample covariance matrix, which becomes impractical for large dimensional matrices, e.g., in modern data applications and large aperture arrays in array signal processing. Even forming the covariance matrix is inviable in many cases. A set of inexpensive methods were proposed for numerical rank estimation of data matrices in the previous chapter (in [2, 3]). However, methods that simultaneously estimate the dimension and obtain an approximation to the principal subspace are lacking.

In this chapter, we present a novel method for estimation of the dimension of the principal subspace of covariance matrices. The method can be combined with Krylov subspace methods to compute an approximation to the principal subspace, simultaneously. The method operates on a model selection framework, and the proposed selection criterion requires computing only the top k eigenvalues of the sample covariance matrix $S_n = \frac{1}{n}XX^T$, where X is the matrix containing n observed data of dimension p, for a given integer k. In order to compute these eigenvalues, we can use the popular Lanczos algorithm [88] which requires only matrix-vector products with S_n. Hence, we do not even need to form the sample covariance matrix $S_n = \frac{1}{n}XX^T$, explicitly. Our approach can be viewed as a stopping criterion for the Krylov subspace method, to simultaneously estimate the dimension and compute the subspace. We only compute the eigenpairs up to the optimal $k = q$, the exact dimension of the principal subspace.

The proposed selection criterion is derived using random matrix perturbation theory results [91]. The criterion also includes a penalty (function) term which under certain assumptions yields us a strongly consistent estimator, i.e., the method estimates the exact dimension as the number of data observations $n \to \infty$. We establish this strong consistency for the proposed method and also present performance analysis. We derive conditions on the signal strength and the noise level for avoiding incorrect dimension estimation in the finite n case, using recently developed random matrix theory results [92]. We also show that the consistency results and the performance analysis hold for the eigenvalues computed by the Krylov subspace methods using the recent results in [93].

4.2 Problem Formulation

The data observations which form the matrix X are typically modeled as high dimensional random quantities embedded in noise. We assume the standard Gaussian random model for the set of n data observations each of dimension p. We denote the p-dimensional data as $\{x_i\}_{i=1}^n$ described as

$$x_i = Ms_i + \sigma n_i,\ i = 1, \ldots, n \tag{4.1}$$

where M is a $p \times q$ mixing matrix with q independent columns, s_i are $q \times 1$ vectors containing the zero mean relevant data and n_i are p-dimensional Gaussian (white) noise vectors with parameter σ^2 as the unknown noise variance.

Typically, the covariance matrix Σ of the underlining (relevant) data is assumed to be a low rank matrix of rank q. That is,

$$\Sigma = BB^T,$$

where $B \in \mathbb{R}^{p\times q}, q \ll p$ and $span(B)$ is the principal subspace. The top q eigenvalues λ_i for $i = 1, \ldots, q$ of the matrix correspond to the q dimensional relevant data s and the remaining $p - q$ eigenvalues are zeros. Hence, the subspace associated with the top q eigenvectors (eigenvalues) forms the principal subspace, which is of interest. However, due to noisy observations, the exact covariance matrix of the underlining data will usually be unavailable (else finding q would be simple), but we can form the sample covariance matrix $S_n = \frac{1}{n}\sum_{i=1}^n x_i x_i^T$ using the n (noisy) observations of the data. Our goal is to estimate q, the dimension of (relevant) data in the observation, i.e., the dimension of the principal subspace. Most methods for dimension estimation are based on the eigenvalues of the sample covariance matrix S_n denoted by $\ell_1 \geq \ell_2 \geq \ldots \geq \ell_p$.

4.3 Proposed Method

In this section, we first present the proposed method for the principal subspace dimension estimation. We then discuss the Krylov subspace methods for computing partial eigen-decomposition of matrices, and present the proposed algorithm for simultaneous estimation of the dimension of the principal subspace and its approximation.

The novel method is based on model selection technique and the proposed criterion is the following:

$$\arg\min_{k} \left[n \sum_{i=k+1}^{p} \ell_i - C_n \frac{(p-k)(p-k+1)}{2} \right] \tag{4.2}$$

where ℓ_i, for $i = 1, \ldots, p$ are the eigenvalues of the sample covariance matrix $S_n = \frac{1}{n}XX^T$, and C_n is a parameter that depends on n. Note that the first term in the criterion depends on the sum of bottom $p-k$ eigenvalues of S_n, which can be written as

$$\sum_{i=k+1}^{p} \ell_i = \frac{1}{n}\|X\|_F^2 - \sum_{i=1}^{k} \ell_i.$$

Thus, the method requires computing only the top k eigenvalues of S_n. Also, if Krylov subspace method such as the Lanczos algorithm [88] is used for computing these eigenvalues, then we do not need to compute $S_n = \frac{1}{n}XX^T$ explicitly. Note that, the Krylov subspace methods also yield an approximation to the eigenvectors corresponding to the computed eigenvalues. Therefore, we can use the above method as a stopping criterion for the Krylov subspace methods, and hence, estimating the dimension and approximating the principal subspace of the covariance matrix, simultaneously. The above criterion is derived in [4] using concepts from random matrix perturbation theory.

Krylov subspace methods

The proposed dimension estimation criterion requires computing only the top k eigenvalues of the sample covariance matrix S_n for a given k, since $\sum_{i=k+1}^{p} \ell_i = \frac{1}{n}\|X\|_F^2 - \sum_{i=1}^{k} \ell_i$. Krylov subspace methods are a popular set of methods used to compute the top k eigenvalues and eigenvectors of matrices [88, 93]. So, the proposed method can be used as a stopping criterion for Krylov subspace approximation of the covariance matrix.

For a symmetric matrix A, the m dimensional Krylov subspace is defined as $\mathcal{K}^m(A, v) =$

$span\{v, Av, \ldots, A^{m-1}v\}$, where v is a random vector of unit norm, $\|v\| = 1$, $v \nsubseteq null(A)$ and m is a scalar. The Lanczos algorithm builds an orthonormal basis for this Krylov subspace [88]. We can also define a block Krylov subspace as: $\mathbb{K}^m(A, V) = span\{V, AV, \ldots, A^{m-1}V\}$, where $V \in \mathbb{R}^{p\times k}$ is a random matrix such that $V \nsubseteq null(A)$, see [93] for recent theoretical results for randomized block Krylov subspace methods. Approximate eigenvalues and eigenvectors of A, say $\{\theta_i, y_i\}_{i=1}^k$ for some k, can be computed using the Krylov subspace methods. We have the following result from eqn. 3 and Theorem 1 in [93]:

Lemma 2 *Consider a symmetric PSD matrix $A \in \mathbb{R}^{p\times p}$ with eigenvalues $\ell_i, i = 1, \ldots, p$. Let $\{\theta_i, y_i\}_{i=1}^k$ be the k eigenpair computed using m steps of block Krylov subspace method (using the orthonormal basis of $\mathbb{K}^m(A, V)$ for $V \in \mathbb{R}^{p\times k}$). If $m = \frac{\log(p)}{\sqrt{\epsilon}}$ for some $0 < \epsilon < 1$, then we have*

$$|\theta_i - \ell_i| \leq \epsilon\ell_{k+1},\ i = 1, \ldots, k.$$

Moreover, suppose Y_k is a matrix containing the eigenvectors $\{y_i\}_{i=1}^k$ computed by the Krylov subspace method as columns, then we have for $\xi \in \{2, F\}$

$$\|A - Y_kY_k^TA\|_\xi \leq (1+\epsilon)\|A - A_k\|_\xi,$$

where A_k is the best rank k approximation of A obtained using its exact eigen-decomposition.

Therefore, when the Krylov subspace method is used to compute the top k eigenvalues of S_n, the eigenvalues θ_i's computed are multiplicative approximations of the eigenvalues ℓ_is of the sample covariance matrix. That is, we have

$$\ell_i - \epsilon\ell_{k+1} \leq \theta_i \leq \ell_i,\ i = 1, \ldots, k.$$

The error ϵ in the above analysis is related to the gap in the spectrum, i.e., we can replace ϵ by $\frac{\ell_k}{\ell_{k+1}} - 1$ to be the error, see [93, §7]. For $k > q$, the error term $\epsilon\ell_{k+1}$ is related to the noise related eigenvalues and we have $\epsilon\ell_{k+1} = O(\frac{1}{\sqrt{n}})$ from the analysis in [94]. Asymptotically, this term goes to zero. Thus, θ_i's have the same statistical properties of ℓ_i's, and are good approximation to them. Since ℓ_i's are asymptotically equivalent to λ_i's, θ_i's are good estimates of λ_i's. From the above Lemma, we also note that the Krylov

Algorithm 3 Proposed Algorithm

```
Input: Data matrix $X \in \mathbb{R}^{p\times n}$, the parameter $C_n$, and a error tolerance $\epsilon$.
Output: Dimension $q$ and an approximation to the principal subspace $Y_q$.
Set $IC = zeros(p,1), Q = [\,], k = 1, m = \log(p)/\sqrt{\epsilon}$.
for $k = 1$ to $p$ do
  1. Generate a random vector $v_k$ with $\|v_k\|_2 = 1$.
  2. $K = \frac{1}{n}[Xv_k, (XX^T)Xv_k, \dots, (XX^T)^{m-1}Xv_k]$.
  3. $Q = orth([Q, K])$, $Q = Q(:, 1:k)$.
  4. $T = \frac{1}{n}Q^TXX^TQ$.
  5. $[V, \Theta] = \texttt{eig}(T)$.
  6. $IC(k) = n(\|X\|_F^2 - \sum_{i=1}^k \theta_i) - C_n\frac{(p-k)(p-k+1)}{2}$.
  if ($k > 1$ && $IC(k) > IC(k-1)$) then
    break;
  end if
end for
$q = k - 1$. Output $q$ and $Y = QV$.
```

subspace method yields a good approximation to the corresponding eigenvectors of the matrix, hence yielding a good approximation to the principal subspace of the covariance matrix.

Proposed Algorithm: Algorithm 3 presents the proposed algorithm for simultaneously estimating the dimension of the principal subspace of the covariance matrix and approximating it. In step 2, note that only matrix-vector products with the data X and its transpose are needed to form the Krylov matrix K. In step 3, since Q is already orthonormal from the previous iteration, the new vectors in K can be quickly orthonormalized with respect to Q using say Gramm-Schmidt algorithm [56]. We can also replace steps 2-5, by a version of the Lanczos algorithm [88], where the previous subspace Q and the tridiagonal matrix T are updated via. the Lanczos algorithm.

4.4 Analysis

In this section, we first show that the proposed method yields a strong consistent estimator for q, the exact dimension. We then analyze when the method will underestimate and overestimate the dimension q, and provide the conditions for incorrect estimation.

4.4.1 Strong consistency

Theorem 4 *The criterion defined by*

$$IC(k) = n \sum_{i=k+1}^{p} \ell_i - C_n \frac{(p-k)(p-k+1)}{2} \tag{4.3}$$

can be used to obtain a strong consistent estimator for q, the exact dimension of the principal subspace. That is, $\lim_{n\to\infty} \hat{k} = q$, where $\hat{k} = \arg\min_k IC(k)$, with value of C_n such that

$$\lim_{n\to\infty} \frac{C_n}{n} = 0 \text{ and } \lim_{n\to\infty} \frac{C_n}{\sqrt{n \log\log n}} = \infty.$$

Proof of the above theorem can be found in [4]. For the right choice of C_n, the proposed estimator is strongly consistent. Next, let us substitute the eigenvalues computed using the Krylov subspace method in our criterion. Then we have the following result:

Corollary 4 *For the choice of C_n in Theorem 6, the criterion 4.2 is strongly consistent for the eigenvalues computed using the Krylov subspace method in Algorithm 3.*

Next, we analyze the performance of the proposed method for finite sample size and obtain the conditions for wrong detection using the recent random matrix theory results.

4.4.2 Performance Analysis

The consistency analysis above considered the asymptotic case when n goes to infinity and the law of iterated logarithm is used to derive the results. Here, we analyze the performance of the proposed method for finite sample size (general n), and obtain the conditions when the method either *underestimates* or *overestimates* the dimension.

The notorious scenario for wrong detection is when the dimension is off by exactly one ($q \pm 1$), which we analyze here. The analysis trivially generalizes to other cases. First, let us consider underestimation by one, and consider the following difference:

$$\begin{aligned} \Delta_1 &= IC(q-1) - IC(q) \\ &= n\ell_q - C_n(p-q+1). \end{aligned}$$

Note that we will not have underestimation when $\Delta_1 > 0$, i.e., when

$$\ell_q > \frac{C_n}{n}(p - q + 1).$$

Thus, for the finite sample size case, we need the signal strength (hence the magnitude of ℓ_q) to be large enough in order not to underestimate the dimension. That is, we need a big gap between relevant eigenvalues and the noise related eigenvalues. For the asymptotic case ($n \to \infty$), we know that the RHS goes to zero by the property of C_n and, hence we will not have any underestimation of the dimension since the eigenvalues are always positive.

Next, let us consider overestimation of the dimension by one, and the following difference:

$$\begin{aligned} \Delta_2 &= IC(q+1) - IC(q) \\ &= C_n(p-q) - n\ell_{q+1}. \end{aligned}$$

Again, we will not overestimate if $\Delta_2 > 0$, i.e., when

$$\ell_{q+1} < \frac{C_n}{n}(p - q).$$

We know that ℓ_{q+1} corresponds to the largest noise related eigenvalue of the covariance matrix. The above equation indicates that the noise level must be low for the method to avoid overestimation in the finite sample size case. For finite n, when the ratio of p/n or n/p is not too large, our method will perform well for reasonable noise levels. In the asymptotic case, we again saw that the method is consistent.

We know that the noise level in the data is σ^2. Using the recent random matrix theory results [92], we can obtain a bound on this noise level for avoiding overestimation for finite but large values of p, n. We know that the largest eigenvalue of the sample covariance matrix (Wishart matrix) of pure noise vectors with Gaussian distribution follows the Tracy-Widom distribution [92]. Then, for finite p, n as long as $\min\{p, n\} \gg 1$ and the ratio of p/n or n/p is not too large, the largest eigenvalue due to noise will be approximately $\sigma^2(1 + \sqrt{p/n})^2$, see [85] for details. Hence for finite but large values of

p, n, we have

$$\ell_{q+1} \approx \sigma^2 \left(1 + \sqrt{\frac{p}{n}}\right)^2.$$

Substituting in the condition above for overestimation, we get the following bound for the noise level for exact detection for finite but large values of p, n:

$$\sigma^2 < \frac{C_n(p-q)}{(\sqrt{n}+\sqrt{p})^2}.$$

The above analysis provides us the conditions on the relevant eigenvalue ℓ_q (gap) and noise level σ^2 in order to avoid incorrect estimation of the dimension q using the proposed method. Therefore, if the noise level σ^2 is known or if it can be estimated, then the parameter C_n can be chosen such that the above condition is always true.

4.5 Numerical experiments

In this section, we present some numerical experimental results to illustrate the performance of the proposed method, and compare it to other popular methods. First, we consider examples for the number of signals detection application in signal and array processing. We then consider few large data matrices to illustrate the method's performance.

4.5.1 Number of signals detection

We saw the problem of estimating the number of signals in the previous chapter. Here, we first look at the standard signal embedded in noise model. We consider p dimensional signals x_i's that are corrupted by white noise with $\mathcal{N}(0, \sigma^2 I)$, variance σ^2. There are three parameters in this model, namely the number of samples n, the signal strength or the magnitude of λ_q eigenvalue, and the noise level σ^2. We compare the performances of the proposed method and the MDL method proposed in [84], as a function of these three parameters. In all experiments, we consider $C_n = 2\sigma^2$ when $n < p$, $C_n = \sigma^2 \log n$ in most other cases, except when when $n \gg p$ (say $> 5p$), we choose $C_n = \sigma^2\sqrt{n}$ to ensure the asymptotic property of C_n holds.

Figure 4.1 presents the three results which illustrate the performances of the two

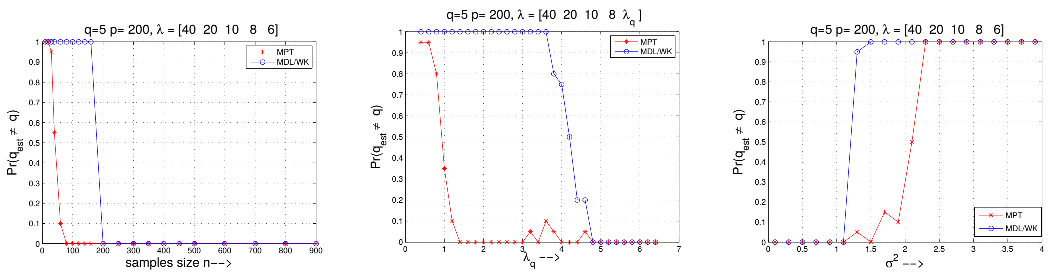


Figure 4.1: Signal detection: Comparison between proposed method MPT and MDL.

methods, the proposed matrix perturbation theory (MPT) based method and the MDL method. For a chosen signal dimension p (reported in the plot), we generate the signals and the sample covariance matrix based on the considered signal eigenvalues λ (listed in the plot). We then add noise covariance matrix corresponding to the noise level σ^2 considered. We plot the probability of the estimated rank q_{est} being not equal to the actual rank q, i.e., $Pr(q_{est} \neq q)$ over 100 trials. In the first plot of Fig. 4.1, we plot $Pr(q_{est} \neq q)$ as a function of the number of samples n. The signal dimension is $p = 200$, the actual rank $q = 5$ and the noise level $\sigma^2 = 1$. The eigenvalues corresponding to the signals are given in the plot. We note that MDL requires $n \geq p$ to yield exact rank, where as the proposed method MPT yields exact rank for much smaller sample size.

In the second (middle) plot, we compare the performances with respect to the signal strength, i.e., the magnitude of the qth eigenvalue λ_q of the covariance matrix. Again the signal dimension is $p = 200$, the actual rank $q = 5$ and the noise level $\sigma^2 = 1$. The number of samples is $n = 400$. We note that, the proposed method again outperforms MDL and yields more accurate results for much smaller signal strength. In the last plot, we compare the performances with respect to the noise level σ^2. Here too, the signal dimension is $p = 200$, the actual rank $q = 5$ and the number of samples is $n = 400$. The signal eigenvalues are given in the plot and the signal strength $\lambda_q = 6$. The proposed method MPT performs better than MDL wrt. the noise level too. This is because, in our method, an appropriate parameter C_n can be chosen based on the noise level σ^2, as seen in the performance analysis, when σ^2 is known (hence the choice $C_n = \sigma^2 \log n$).

Non-Gaussian signal model: Next, we consider the signals to be non-Gaussian, and simulate a uniform linear array of p sensors with q incoherent sinusoidal plane waves

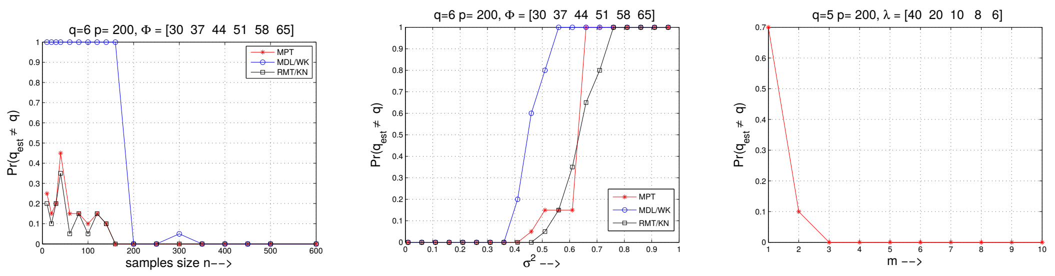


Figure 4.2: Signal detection: Comparison between MPT, MDL and RMT methods and Krylov method.

impinging from directions $[\phi_1, \ldots, \phi_q]$. Here, the received signals is modeled as,

$$y(t) = \sum_{k=1}^{q} H(\phi_k)e^{-j\eta(t)} + n(t),$$

where $H(\phi_k)^T = [1, e^{-j\tau_k}, \ldots, e^{-j(q-1)\tau_k}]$ are steering (direction) vectors with $\tau_k = \pi \sin \phi_k$, and $\eta(.)$ = random phase uniformly distributed on $(0, 2\pi)$. The noise added $n(.)$ = white noise with Gaussian distribution $\mathcal{N}(0, \sigma^2 I)$, with variance σ^2. In this case, the signals are not Gaussian distributed. We compare the performances of the proposed method MPT and MDL [84], along with a threshold based method proposed in [95] based on random matrix theory (RMT) for signal detection in this model.

The first two plots in Figure 4.2 present the results comparing the performances of the three methods. We again plot the probability of the estimated rank q_{est} being not equal to the actual rank q, i.e., $Pr(q_{est} \neq q)$ over 100 trials. In both plots, the signal dimension is $p = 200$, the rank is $q = 6$, and the angles of incidence are $\Phi = [30^o, 37^o, 44^o, 51^o, 58^o, 65^o]$. In the first plot, we compare the performances as a function of the sample size n, with the noise level $\sigma^2 = 0.2$. In the second plot, we compare the performances as a function of the noise level σ^2, with $n = 400$ (The magnitude of the smallest signal eigenvalue was between 2 to 2.5). We observe that the performance of our method is superior to MDL even in the non-Gaussian signal case, and is similar to the state of the art threshold selection method RMT. However, RMT has parameters, such as confidence level α to be selected and the noise level σ^2 used tends to yield wrong results in some cases. Moreover, both MDL and RMT require computing of all

Table 4.1: Performance of the Krylov Subspace method, Algorithm 3 with $m = 10$.

Dataset	n	Actual q	λ_q	σ^2	Estimated $\tilde{q}$	$\|A - Y_{\tilde{q}}Y_{\tilde{q}}^T A\|_F$	Runtime
sprand	500	50	5	1	50	1.1e4	0.72 secs
	500	100	2	0.5	100	1.5e4	1.42 secs
	1000	100	2	0.5	100	2.9e4	5.11secs
	1000	100	1	0.3	100	2.8e4	5.35secs
	4000	100	2	0.3	100	5.0e4	20.56secs
Harvard	500	169	0.73	0.25	165	34.2	0.44secs
CSPhD	1882	705	0.8	0.25	705	32.1	1.8secs
lpiceria3d	3576	122	5	0.25	145	120.5	0.61secs
EVA	8497	349	2	0.25	349	55.2	4.92secs
lpstocfor3	16675	981	562	0.25	981	3.0e4	548secs

the eigenvalues of the sample covariance matrix.

Krylov subspace method: In the previous two set of experiments, we used the exact eigenvalues of the covariance matrices (computed using `eig` function in Matlab) for the dimension estimation by the three compared methods (MDL and RMT require all eigenvalues). Next, we illustrate how the proposed Krylov subspace based algorithm 3 performs for the dimension estimation. The last plot in figure 4.2 give the performance of the algorithm as a function of the number of Lanczos steps m. We know the relation between the error ϵ in the eigenvalue estimation by the Lanczos algorithm and the number of Lanczos steps m from Lemma 2. Hence, increasing m is equivalent to decreasing ϵ. We see that for a very few Lanczos steps $m \geq 4$, we get accurate results. This superior performance of the Lanczos algorithm was observed in [90] as well for a similar Gaussian signal detection model. With Algorithm 3, for $m \geq 4$, we observed that the performance of the proposed method for different n, σ^2, λ_q was similar to what we have reported in Figure 4.1.

4.5.2 Data matrices

In the last set of experiments, we illustrate the performance of the proposed method for numerical rank estimation of data matrices. We consider general data matrices that have low numerical rank from publicly available database, SuiteSparse [66], and a few synthetic sparse random matrices.

Table 4.1 presents the performance of the Krylov Subspace method, i.e., Algorithm 9 for dimension estimation and approximation of the principal subspace. We consider few synthetic sparse random matrices of the form $X = A\Lambda A^T + N$, where A is a sparse signal matrix of size $n \times q$ (sparsity $\text{nnz}(A)/nq = 0.2$), Λ is a diagonal matrix with the smallest diagonal entry equal to λ_q listed in the 4th column. N is a Gaussian sparse random matrix with σ^2 listed in fifth column. The number of Lanczos steps per iteration (for each k) is $m = 10$. The exact dimension q and the estimated dimension $\tilde{q}$ are reported (dimension estimation), along with the Frobenius norm error $\|A - Y_{\tilde{q}} Y_{\tilde{q}}^T A\|_F$ evaluating the approximation to the principal subspace. The runtime of the algorithm is also reported (computed using `cputime` function on an Intel i-5 3.4GHz machine). For the synthetic examples, we vary the following parameters :size n, rank q, signal strength λ_q and noise level σ^2, and report the results. We note that the proposed algorithm performs very well and is inexpensive. We also consider a few sparse data matrices. We report only those matrices that have smaller numerical rank ($q \ll \min(n, p)$) and a reasonable gap in the spectrum. Lanczos algorithm works well only when there is a spectral gap. Otherwise, the interior eigenvalues do not converge. We observe that the algorithm performs reasonably well for the reported matrices.

Chapter 5

Matrix approximations via coarsening

5.1 Introduction

Modern applications involving data often rely on very large datasets, but in most situations the relevant information lies in a low dimensional subspace. In many of these applications, the data matrices are sparse and/or are representations of large graphs. In recent years, there has been a surge of interest in approximating large matrices in a variety of different ways, such as by low rank approximations [7, 67, 96], graph sparsification [32], and compression. Methods to obtain low rank approximations include the partial singular value decomposition (SVD) [67] and Column Subset Selection (CSS) [25]. Efficient methods have been developed to compute the partial SVD [88, 56], a problem that has been studied for a few decades. However, traditional methods for partial SVD computations cannot cope with very large data matrices. Such datasets prohibit even the use of rather ubiquitous methods such as the Lanczos or subspace iteration algorithms [88], since these algorithms require consecutive accesses to the whole matrix multiple times. Computing such matrix approximations is even harder in the scenarios where the matrix under consideration receives frequent updates in the form of new columns or rows.

Much recent attention has been devoted to a class of 'randomization' techniques [97, 96, 67] whereby an approximate partial SVD is obtained from a small randomly sampled subset of the matrix, or possibly a few subsets. Several randomized embedding and sketching methods have also been proposed [67, 28]. These randomization techniques are well-established (theoretically) and are proven to give good results in some situations. However, randomized methods by themselves can be suboptimal in many situations since they do not exploit available information or the redundancies in the matrix. For example, many sampling methods only consider column norms, and embedding and sketching methods are usually independent of the input matrix. One of the goals of this chapter is to show that multilevel graph coarsening, a technique that is often used in the differnt context of graph partitioning [98], can provide superior alternatives to randomized sampling, at a moderate cost.

Coarsening a graph (or a hypergraph) $G = (V, E)$ means finding a 'coarse' approximation $\bar{G} = (\bar{V}, \bar{E})$ to G with $|\bar{V}| < |V|$, which is a reduced representation of the original graph G, that retains as much of the structure of the original graph as possible. Multilevel coarsening refers to the technique of recursively coarsening the original graph to obtain a succession of smaller graphs that approximate the original graph G. These techniques can be more expensive than down-sampling with column norm probabilities [96] but they are also more accurate. Moreover, coarsening methods will be inexpensive compared to the popular leverage scores based sampling [27] which is more accurate than column norm based sampling. For very large matrices, a typical algorithm would first perform randomized sampling to reduce the size of the problem and then utilize a multilevel coarsening technique for computing an approximate partial SVD of the reduced matrix.

The second low rank approximation problem considered in this chapter is the *column subset selection problem* [25] (CSSP) or CUR decomposition [26, 27]. Here, the goal is to select a subset of columns that best represent the given matrix spectrally, i.e., with respect to spectral and Frobenius norms. Popular methods for the CSSP use the leverage score sampling measure for sampling/selecting the columns. Computing the leverage scores requires a partial SVD of the matrix and this may be expensive, particularly for large matrices and when the (numerical) rank is not small. We will see how graph coarsening techniques can be adapted for column subset selection. The coarsening approach is an inexpensive alternative for this problem and it performs well in many situations as the experiments will show.

The third problem we consider is that of *graph sparsification* [29, 32]. Here, given a large (possibly dense) graph G, we wish to obtain a sparsified graph $\tilde{G}$ that has significantly fewer edges than G but still maintains important properties of the original graph. Graph sparsification allows one to operate on large (dense) graphs G with a reduced space and time complexity. In particular, we are interested in spectral sparsifiers, where the Laplacian of $\tilde{G}$ spectrally approximates the Laplacian of G [31, 32, 28]. That is, the spectral norm of the Laplacian of the sparsified graph is close to the spectral norm of the Laplacian of G, within a certain additive or multiplicative factor. Such spectral sparsifiers can help approximately solve linear systems with the Laplacian of G and to approximate effective resistances, spectral clusterings, random walk properties, and a variety of other computations. We will again show how graph coarsening can be adapted for the task of graph sparsification.

5.2 Applications

We first present a few applications, where the (multilevel) coarsening methods discussed in this chapter can be employed. The matrices encountered in these applications are typically large, and sparse, often representing graphs.

i. Latent Semantic Indexing - Latent semantic indexing (LSI) is a well-established text mining technique for processing queries in a collection of documents [99, 100]. Given a user's query, the method is used to retrieve a set of documents from a given collection that are most relevant to the query. The truncated SVD [99] and related techniques [100] are common tools used in LSI. The argument exploited is that a low rank approximation preserves the important underlying structure associated with terms and documents, and removes the noise or variability in word usage [96]. Multilevel coarsening for LSI was considered in [101].

ii. Projective clustering - Several projective clustering methods such as Isomap, Local Linear Embedding (LLE), spectral clustering, subspace clustering, Laplacian eigenmaps and others involve a partial eigen-decomposition or a partial SVD of graph Laplacians. Various kernel based learning methods also require the (partial) SVD of large graph Laplacians. In most applications today, the number of data-points is large and computing singular vectors (eigenvectors) is expensive in most cases. Graph coarsening is an effective strategy to reduce the number of data-points in these applications, see [102, 103] for a few illustrations.

iii. Eigengene analysis - Analyzing gene expression DNA microarray data has become an important tool in the study of a variety of biological processes [104, 105]. In a microarray dataset, we have m genes (from m individuals possibly from different populations) and a series of n arrays probe genome-wide expression levels in n different samples, possibly under n different experimental conditions. The data is large with several individuals and gene expressions, but is known to be of low rank. Hence, it has been shown that a small number of eigengenes and eigenarrays (few singular vectors) are sufficient to capture most of the gene expression information [104]. Article [105] showed how column subset selection (CSSP) can be used for selecting a subset of gene expressions that describe the population well in terms of spectral information captured by the reduction. Later we will show how hypergraph coarsening can be adapted to choose a good (small) subset of genes in this application.

iv. Multilabel Classification - The last application we consider is that of multilabel classification in machine learning [106, 10]. In the multilabel classification problem, we are given a set of labeled training data $\{(x_i, y_i)\}_{i=1}^n$, where each $x_i \in \mathbb{R}^p$ is an input feature for a data instance which belongs to one or more classes, and $y_i \in \{0, 1\}^d$ are vectors indicating the corresponding labels (classes) to which the data instances belong. A vector y_i has a one at the j-th coordinate if the instance belongs to j-th class. We wish to learn a mapping (prediction rule) between the features and the labels, in order to be able to predict a class label vector y of a new data point x. Such multilabel classification problems occur in many domains such as computer vision, text mining, and bioinformatics [107], and modern applications involve a large number of labels.

A common approach to handle classification problems with many classes is to begin by reducing the effective number of labels by means of so-called embedding-based approaches. The label dimension is reduced by projecting label vectors onto a low dimensional space, based on the assumption that the label matrix $Y = [y_1, \ldots, y_n]$ has a low-rank. The reduction is achieved in different ways, for example, by using SVD in [107] and column subset selection in [108]. In this work, we demonstrate how hypergraph coarsening can be employed to reduce the number of classes, and yet achieve accurate learning and prediction.

5.3 Hypergraph Coarsening

Hypergraphs extend the classical notion of graphs. A hypergraph $H = (V, E)$ consists of a set of vertices V and a set of hyperedges E [109, 110]. In a standard graph an edge connects two vertices, whereas a hyperedge may connect an arbitrary subset of vertices. A hypergraph $H = (V, E)$ can be canonically represented by a sparse matrix A, where the vertices in V and hyperedges (nets) in E are represented by the columns and rows of A, respectively. This is called the *row-net model.* Each hyperedge, a row of A, connects the vertices, i.e., the columns, whose corresponding entries in that row are non-zero.

Given a (sparse) data set of n entries in $\mathbb{R}^m$ represented by a matrix $A \in \mathbb{R}^{m\times n}$, we can consider a corresponding hypergraph $H = (V, E)$ with vertex set V representing to the columns of A. Several methods exist for coarsening hypergraphs, see, e.g., [109]. Here, we consider a hypergraph coarsening based on column matching, which is a modified version of the *maximum-weight matching* method, e.g., [109]. The modified approach follows the maximum-weight matching method and computes the non-zero inner product $\langle a^{(i)}, a^{(j)}\rangle$ between two vertices i and j, i.e., the i-th and j-th columns of A. Note that the inner product between vectors is related to the angle between the vectors, i.e., $\langle a^{(i)}, a^{(j)}\rangle = \|a^{(i)}\|\|a^{(j)}\| \cos\theta_{ij}$. The proposed coarsening strategy is to match two vertices, i.e., columns, only if the angle between the vertices is such that, $\tan\theta_{ij} \le \epsilon$, for a constant $0 < \epsilon < 1$. Another feature of the proposed algorithm is that it applies a scaling to the coarsened columns in order to reduce the error. In summary, we combine two columns $a^{(i)}$ and $a^{(j)}$ if the angle between them is such that, $\tan\theta_{ij} \le \epsilon$. We replace the two columns $a^{(i)}$ and $a^{(j)}$ by

$$c^{(\ell)} = \left(\sqrt{1+\cos^2\theta_{ij}}\right) a^{(i)}$$

where $a^{(i)}$ is replaced by $a^{(j)}$ if the latter has more nonzeros. This minor modification

Algorithm 4 Hypergraph coarsening by column matching.

```
**Input:** $A \in \mathbb{R}^{m \times n}$, $\epsilon \in (0, 1)$.
**Output:** Coarse matrix $C \in \mathbb{R}^{m \times c}$.
$Idx := \{1, \ldots, n\}$
Set ip$[k] := 0$ for $k = 1, \ldots, n$, and $\ell = 1$.
**repeat**
  Randomly pick $i \in Idx$; $Idx := Idx - \{i\}$.
  **for all** $j$ with $a_{ij} \neq 0$ **do**
    **for all** $k$ with $a_{jk} \neq 0$ **do**
      ip$[k] :=$ ip$[k] + a_{ij}a_{jk}$.
    **end for**
  **end for**
  $j :=$ argmax$\{$ip$[k] : k \in Idx\}$
  $csq\theta = \frac{\text{ip}[j]^2}{\|a^{(i)}\|^2 \|a^{(j)}\|^2}$.
  **if** [ $(csq\theta \geq \frac{1}{1+\epsilon^2})$] **then**
    $c^{(\ell)} := \sqrt{1 + csq\theta} a^{(i)}$. (The denser of columns $a^{(i)}$ and $a^{(j)}$)
    $Idx := Idx - \{j\}; \ell = \ell + 1$.
  **else**
    $c^{(\ell)} := a^{(i)}$.
    $\ell = \ell + 1$.
  **end if**
  Reset nonzero values of ip to zero (A sparse operation)
**until** $Idx = \emptyset$
```

provides some control over the coarsening procedure using the parameter ϵ and, more importantly, it helps establish theoretical results for the method, see section 7.3.

The vertices can be visited in a random order, or in the ‘natural’ order in which they are listed. For each unmatched vertex i, all the unmatched neighbor vertices j are explored and the inner product between i and each j is computed. This typically requires the data structures of A and its transpose, in that a fast access to rows and columns is required. The vertex j with the highest non-zero inner product $\langle a^{(i)}, a^{(j)} \rangle$ is considered and if the angle between them is such that $\tan \theta_{ij} \leq \epsilon$ (or $\cos^2 \theta_{ij} \geq \frac{1}{1+\epsilon^2}$)), then i is matched with j and the procedure is repeated until all vertices have been matched. Algorithm 4 provides details on the procedure.

Computing the cosine of the angle between column i and all other columns is equivalent to computing the i-th row of $A^T A$, in fact only the upper triangular part of the row. For sparse matrices, the inexpensive computation of the inner product between the columns used in the algorithm is achieved by modifying the cosine algorithm in [111] developed for matrix blocks detection. Thus, loop computes all inner products of column i with all other columns, and accumulates these in the sparse 'row' ip[.]. This amounts in essence to computing the i-th row of A^TA as the combination of rows: $\sum_{a_{ij}\neq 0} a_{ij}a_{j,:}$. As indicated in the line just before the end, resetting ip[.] to zero is a sparse operation that does not require zeroing out the whole vector but only those entries that are nonzero.

The pairing used by the algorithm relies only on the sparsity pattern. It is clear that these entries can also be used to obtain a pairing based on the cosine of the angles between columns i and k. The coarse column $c^{(p)}$ is defined as the 'denser of columns $a^{(i)}$ and $a^{(j)}$'. In other models the sum is sometimes used. Note that the number of vertices (columns) in C will depend on the redundancy among the data and the ϵ value chosen, see further discussion in sec. 7.3.

Computational Cost We saw earlier that the inner products of a given column i with all other columns amounts to computing the nonzero values of the i-th row of the upper triangular part of A^TA. If we call $Adj_A(i)$ the set of nonzero indices of the i-th column of A then according to what was said above, the cost of computing ip[.] by the algorithm is

$$\sum_{j\ \in\ Adj_A(i),\ j>i} |Adj_{A^T}(j)|,$$

where $|\cdot|$ is the cardinality of the set. If ν_c (resp. ν_r) is the maximum number of nonzeros in each column (resp. row), then an upper bound for the above cost is $n\nu_r\nu_c$, which is the same upper bound as that of computing the upper triangular part of A^TA. Several simplifications and improvements can be added to reduce the cost. First, we can skip the columns that are already matched. In this way, fewer inner products are computed as the algorithm progresses. In addition, since we only need the angle to be such that $\tan\theta_{ij} \leq \epsilon$, we can reduce the computation cost significantly by stopping as soon as we encounter a column with which the angle is smaller than the threshold.

5.3.1 Multilevel SVD computations

Given a sparse matrix A, we can use Algorithm 4 repeatedly with different (increasing) ϵ values, to recursively coarsen the corresponding hypergraph, and obtain a sequence of sparse matrices $A_1, A_2, \dots, A_s$ with $A_0 = A$, where A_i corresponds to the coarse graph H_i of level i for $i = 1, \dots, s$, and A_s represents the lowest level graph H_s. This provides a reduced size matrix which will likely be a good representation of the original data. Note that, recursive coarsening will be inexpensive since the inner products required in the further levels are already computed in the first level of coarsening.

In the multilevel framework of hypergraph coarsening we apply the matrix approximation method, say using the SVD, to the coarsened data matrix $A_s \in \mathbb{R}^{m\times n_s}$ at the lowest level, where n_s is the number of data items at the coarse level s ($n_s < n$). A low-rank matrix approximation can be viewed as a linear projection of the columns into a lower dimensional space. In other words we have a matrix $\hat{A}_s \in \mathbb{R}^{d\times n_s}$ ($d < m$). Applying the same linear projection to $A \in \mathbb{R}^{m\times n}$ produces $\hat{A} \in \mathbb{R}^{d\times n}$ ($d < m$), and one can expect that $\hat{A}$ preserves certain features of A. This linear projection is then applied to the original data $A \in \mathbb{R}^{m\times n}$ to obtain a reduced representation $\hat{A} \in \mathbb{R}^{d\times n}$ ($d < m$) of the original data. Another strategy for reducing the matrix dimension is to mix the two techniques: Coarsening may still be exceedingly expensive for some types of data where there is no immediate graph available to exploit for coarsening. In this case, a good strategy would be to downsample first using the randomized methods, then construct a graph and coarsen it.

5.3.2 CSSP and graph sparsification

The multilevel coarsening technique just presented can be applied for the column subset selection problem (CSSP) as well as for the graph sparsification problem. We can use Algorithm 4 to coarsen the matrix, and this is equivalent to selecting columns of the matrix. The only required modification in the algorithm is that the columns selected are no longer scaled. The coarse matrix C contains a subset of the columns of the original matrix A and the analysis will show that it is a faithful representation of A.

For graph sparsification, we can apply the coarsening procedure on the matrix B^T, i.e., coarsen the rows of the vertex edge incidence B, which yields us fewer edges, $\tilde{B}$

with fewer rows. The analysis in the next section shows how this coarsening strategy is indeed a spectral sparsifier, shows $x^T\tilde{B}^T\tilde{B}x$ is close to x^TB^TBx. Since we achieve sparsification via matching, the structures such as clusters within the original graph are also preserved.

5.4 Analysis

In this section, we establish initial theoretical results for the coarsening technique based on column matching. In the coarsening strategy of Algorithm 4, we combine two columns $a^{(i)}$ and $a^{(j)}$ if the angle between them is such that $\tan\theta_i \leq \epsilon$. That is, we set $c^{(\ell)} = (\sqrt{1+\cos^2\theta_i})a^{(i)}$ (or $a^{(j)}$, if it has more nonzeros) in place of the two columns in the coarsened matrix C. We then have the following result which relates the Rayleigh quotients of the coarsened matrix C to that of A (indicates $AA^T \approx CC^T$, spectrally).

Lemma 3 *Given $A \in \mathbb{R}^{m\times n}$, let $C \in \mathbb{R}^{m\times c}$ be the coarsened matrix of A obtained by one level of coarsening of A with columns $a^{(i)}$ and $a^{(\hat{i})}$ matched if $\tan\theta_i \leq \epsilon$, for $0<\epsilon<1$. Then,*

$$|x^TAA^Tx - x^TCC^Tx| \leq 3\epsilon\|A\|_F^2, \tag{5.1}$$

for any $x \in \mathbb{R}^n : \|x\| = 1$.

Proof. Let (i,j) be a pair of matched column indices with i being the index of the column that is retained after scaling. We denote by I the set of all indices of the retained columns and J the set of the remaining columns.

We know that $\sigma_i^2(A) = \sigma_i(AA^T) = \max_{\|x\|=1} x^TAA^Tx$, and also $x^TAA^Tx = \|A^Tx\|_2^2 = \sum_{i=1}^n \langle a^{(i)}, x\rangle^2$. Similarly, consider $x^TCC^Tx = \|C^Tx\|_2^2 = \sum_{i\in I}\langle c_i, x\rangle^2 = \sum_{i\in I}(1+c_i^2)\langle a^{(i)}, x\rangle^2$, where indices $c_i = \cos\theta_i$. We have,

$$\begin{aligned}
|x^TAA^Tx - x^TCC^Tx| &= |\sum_{i\in I\cup J}\langle a^{(i)}, x\rangle^2 - \sum_{i\in I}(1+c_i^2)\langle a^{(i)}, x\rangle^2| \\
&\leq |\sum_{j\in J}\langle a^{(j)}, x\rangle^2 - \sum_{i\in I} c_i^2\langle a^{(i)}, x\rangle^2| \\
&= \sum_{(i,j)\in I\times J}\left[\langle a^{(j)}, x\rangle^2 - c_i^2\langle a^{(i)}, x\rangle^2\right]
\end{aligned}$$

where the set $I \times J$ consists of pairs of indices (i, j) that are matched. Next, we consider an individual term in the sum. Let column $a^{(j)}$ be decomposed as follows:

$$a^{(j)} = c_i a^{(i)} + s_i w,$$

where $s_i = \sin\theta_i$ and $w = \|a^{(i)}\|\bar{w}$ with $\bar{w}$ a unit vector that is orthogonal to $a^{(i)}$. Then,

$$\begin{aligned} |\langle a^{(j)}, x\rangle^2 - c_i^2 \langle a^{(i)}, x\rangle^2| &= \left|\langle c_i a^{(i)} + s_i w, x\rangle^2 - c_i^2\langle a^{(i)}, x\rangle^2\right| \\ &= \left|c_i^2\langle a^{(i)}, x\rangle^2 + 2c_i s_i \langle a^{(i)}, x\rangle\langle w, x\rangle + s_i^2\langle w, x\rangle^2 - c_i^2\langle a^{(i)}, x\rangle^2\right| \\ &= |\sin 2\theta_i \langle a^{(i)}, x\rangle\langle w, x\rangle + \sin\theta_i^2\langle w, x\rangle^2| \end{aligned}$$

If $t_i = \tan\theta_i$, then $\sin 2\theta_i = \frac{2t_i}{1+t_i^2}$. Using the fact that $|\langle w, x\rangle| \leq \|a^{(i)}\| \equiv \eta_i$ and $\langle a^{(i)}, x\rangle \leq \eta_i$, we get

$$\begin{aligned} |\sin 2\theta_i\langle a^{(i)}, x\rangle\langle w, x\rangle + \sin\theta_i^2\langle w, x\rangle^2| &\leq \eta_i^2 \sin 2\theta_i \left[1 + \frac{\sin^2\theta_i}{2\sin\theta_i\cos\theta_i}\right] \\ &= \eta_i^2\sin 2\theta_i\left[1 + \frac{\tan\theta_i}{2}\right] \\ &\leq \frac{2\eta_i^2 t_i + (\eta_i t_i)^2}{1 + t_i^2} \\ &\leq 2\eta_i^2 t_i + (\eta_i t_i)^2. \end{aligned}$$

Now, since our algorithm combines two columns only if $\tan(\theta_i) \leq \epsilon$ (or $\cos^2\theta \geq 1/(1+\epsilon^2)$), we have

$$|\langle a^{(j)}, x\rangle^2 - c_i^2\langle a^{(i)}, x\rangle^2| \leq 2\eta_i^2\epsilon + \eta_i^2\epsilon^2 \leq 3\epsilon\eta_i^2$$

as $\epsilon < 1$. Thus, we have

$$|x^T AA^T x - x^T CC^T x| \leq 3\epsilon \sum_{i\in I}\|a^{(i)}\|^2 \leq 3\epsilon\|A\|_F^2.$$

$\square$

Note that the above bound will be better in practice because the last inequality may not be tight. In fact improving this result could be an interesting question to investigate. We also observe that the above result will hold even if we consider combining multiple columns which are within the angle $\theta = \tan^{-1}(\epsilon)$ from each other into one in Algorithm 4. The number of columns c in the coarsened matrix C will depend on the given data (the number of pairs of columns that are within the desired angle). It can be used to develop the following error bounds.

Theorem 5 *Given $A \in \mathbb{R}^{m\times n}$, let $C \in \mathbb{R}^{m\times c}$ be the coarsened matrix of A obtained by one level coarsening of A with columns $a^{(i)}$ and $a^{(\hat{i})}$ combined if $\tan\theta_i \leq \epsilon$, for $0 < \epsilon < 1$. Let H_k be the matrix consisting of the top k left singular vectors of C as columns. Then, we have*

$$\begin{aligned} \|A - H_kH_k^TA\|_F^2 &\leq \|A - A_k\|_F^2 + 6k\epsilon\|A\|_F^2 && (5.2)\\ \|A - H_kH_k^TA\|_2^2 &\leq \|A - A_k\|_2^2 + 6\epsilon\|A\|_F^2, && (5.3)\end{aligned}$$

where A_k is the best rank k approximation of A.

Proof.

Frobenius norm error: First, we prove the Frobenius norm error bound. We can express $\|A - H_kH_k^TA\|_F^2$:

$$\begin{aligned} \|A - H_kH_k^TA\|_F^2 &= Tr((A - H_kH_k^TA)^T(A - H_kH_k^TA)) && (5.4)\\ &= Tr(A^TA - 2A^TH_kH_k^TA + A^TH_kH_k^TH_kH_k^TA)\\ &= Tr(A^TA) - Tr(A^TH_kH_k^TA)\\ &= \|A\|_F^2 - \|A^TH_k\|_F^2. \end{aligned}$$

We get the above simplifications using the equalities: $\|X\|_F^2 = Tr(X^TX)$ and $H_k^TH_k = I$. Let $h^{(i)}$ for $i = 1, \dots, k$ be the columns of H_k. Then, the second term in the above equation is $\|A^TH_k\|_F^2 = \sum_{i=1}^k \|A^Th^{(i)}\|^2$.

From Lemma 3, we have for each i,

$$|\|A^Th^{(i)}\|^2 - \|C^Th^{(i)}\|^2| = |\|A^Th^{(i)}\|^2 - \sigma_i^2(C)| \leq 3\epsilon\|A\|_F^2,$$

since $h^{(i)}$'s are the singular vectors of C. Summing over k singular vectors, we get

$$||\|A^T H_k\|_F^2 - \sum_{i=1}^{k} \sigma_i^2(C)| \le 3\epsilon k \|A\|_F^2. \tag{5.5}$$

From perturbation theory [56, Thm. 8.1.4], we have for $i = 1, \ldots, c$:

$$|\sigma_i^2(C) - \sigma_i^2(A)| \le \|AA^T - CC^T\|_2.$$

Next, Lemma 3 implies:

$$\|AA^T - CC^T\|_2 = \max_{x \in \mathbb{R}^n : \|x\|=1} |x^T(AA^T - CC^T)x| \le 3\epsilon \|A\|_F^2.$$

Hence, summing over k singular values,

$$\left| \sum_{i=1}^{k} \sigma_i^2(C) - \sum_{i=1}^{k} \sigma_i^2(A) \right| \le 3\epsilon k \|A\|_F^2. \tag{5.6}$$

Combining (5.5) and (5.6), we get

$$\left| \|A^T H_k\|_F^2 - \sum_{i=1}^{k} \sigma_i^2(A) \right| \le 6\epsilon k \|A\|_F^2.$$

Along with (5.4) this relation gives us the Frobenius norm error bound, since $\|A\|_F^2 - \sum_{i=1}^{k} \sigma_i^2(A) = \|A - A_k\|_F^2$.

Spectral norm error: Let $\mathcal{H}_k = range(H_k) = span(h^{(1)}, \ldots, h^{(k)})$ and let $\mathcal{H}_{n-k}$ be the orthogonal complement of $\mathcal{H}_k$. For $x \in \mathbb{R}^n$, let $x = \alpha y + \beta z$, where $y \in \mathcal{H}_k, z \in \mathcal{H}_{n-k}$ and $\alpha^2 + \beta^2 = 1$. Then,

$$\begin{aligned}
\|A - H_k H_k^T A\|_2^2 &= \max_{x \in \mathbb{R}^n : \|x\|=1} \|x^T(A - H_k H_k^T A)\|^2 \\
&= \max_{y,z} \|(\alpha y^T + \beta z^T)(A - H_k H_k^T A)\|^2 \\
&\le \max_{y \in \mathcal{H}_k : \|y\|=1} \|y^T(A - H_k H_k^T A)\|^2 + \max_{z \in \mathcal{H}_{n-k} : \|z\|=1} \|z^T(A - H_k H_k^T A)\|^2 \\
&= \max_{z \in \mathcal{H}_{n-k} : \|z\|=1} \|z^T A\|^2,
\end{aligned}$$

since $\alpha, \beta \leq 1$ and for any $y \in \mathcal{H}_k, y^T H_k H_k^T = y^T$, so the first term is zero and for any $z \in \mathcal{H}_{n-k}, z^T H_k H_k^T = 0$. Next,

$$\begin{aligned} \|z^T A\|^2 &= \|z^T C\|^2 + [\|z^T A\|^2 - \|z^T C\|^2] \\ &\leq \sigma_{k+1}^2(C) + 3\epsilon\|A\|_F^2 \\ &\leq \sigma_{k+1}^2(A) + 6\epsilon\|A\|_F^2 \\ &= \|A - A_k\|_2^2 + 6\epsilon\|A\|_F^2. \end{aligned}$$

Since $|\|z^T A\|^2 - \|z^T C\|^2| \leq 3\epsilon\|A\|_F^2$ from Lemma 3, $\max_{z \in \mathcal{H}_{n-k}:\|z\|=1} \|z^T C\|^2 = \sigma_{k+1}^2(C)$, and $|\sigma_i^2(C) - \sigma_i^2(A)| \leq \|AA^T - CC^T\|_2 \leq 3\epsilon\|A\|_F^2$. $\square$

We observe that Theorem 6 is similar to the results developed for randomized sampling methods, see [97, 96]. One notable difference is that to achieve the above result for a given rank k, the method based on column norms for randomized sampling requires $c = \Theta(k/\epsilon^2)$ columns to be sampled to form C. For a small ϵ, the number of columns c will be quite large. The error for a given rank k diminishes as c increases, but when k is large, a large number of columns will be required to get a good approximation. For the coarsening method, the above error bounds hold for any $k \leq c$, and the number of columns c will depend primarily on the given data. The error will be smaller if the angles between the columns that are combined are smaller. The number of columns is related to these angles and this in turn depends on the redundancy among columns of the given matrix. As future work it would be interesting to say how many distinct columns will be needed to ensure that the subspace spanned by the columns of C is a good rank k approximation to the range of A.

5.5 Numerical Experiments

Here we give few examples to illustrate the performance of coarsening in partial SVD computation. We use three term-by-document datasets and compare the sampling, coarsening and combined methods to compute the SVD. The tests are with unweighted versions of the CRANFIELD dataset (1398 documents, 5204 terms), MEDLINE dataset (1033 documents, 8322 terms) and TIME dataset (425 documents, 13057 terms).

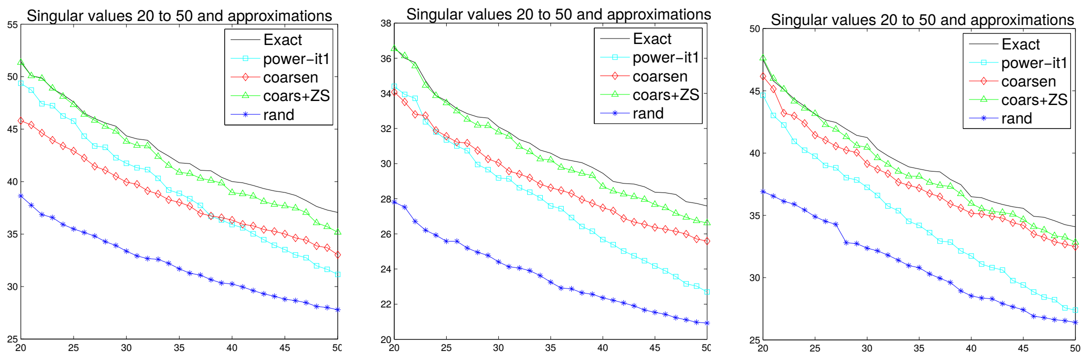


Figure 5.1: Results for the datasets CRANFIELD (left), MEDLINE (middle), and TIME (right).

Figure 5.1 illustrates the following experiment with the three datasets. Results from four different methods are plotted. The first solid curve (labeled 'exact') shows the singular values of matrix A from 20 to 50 computed using the svds function in Matlab (the results obtained by the four methods for top twenty singular values were similar). The diamond curve labeled 'coarsen', shows the singular values obtained by one level of coarsening.The star curve (labeled 'rand') shows the singular values obtained by random sampling using column norms, with a sample size equal to the size obtained with one level of coarsening. We note that the result obtained by coarsening is much better than that obtained by random sampling. However, we know that the approximations obtained by either sampling or coarsening cannot be highly accurate. In order to get improved results, we can invoke incremental SVD algorithms [5].The curve with triangles labeled 'coars+ZS' shows the singular values obtained when Zha Simon algorithm [5] was used to improve the results obtained by the coarsening algorithm. Here, we consider the singular vectors of the coarse matrix and use the remaining part of the matrix to update these singular vectors and singular values. We have also included the results obtained by one iteration of power method [67], i.e., from the SVD of the matrix $Y = (AA^T)A\Omega$, where Ω is a random Gaussian matrix of same size as the coarse matrix. We see that the smaller singular values obtained from the coarsening algorithms are better than those obtained by the one-step power method.

Table 5.1: CSSP: Coarsening versus leverage score sampling.

Dataset	Size	Rank k	c	Coarsening		levSamp
				levels	error	error
CRAN	1398	25	88	4	496.96	501.32
		50	88	4	467.49	477.25
		150	175	3	375.40	383.23
MED	1033	50	65	4	384.91	376.23
		100	130	4	341.51	339.01
TIME	425	25	107	2	411.71	412.77
		50	107	2	371.35	372.66
		50	54	3	389.69	391.91
Kohonen	4470	25	981	2	31.89	36.36
Erdos992	6100	50	924	3	100.9	99.29
FA	10617	50	2051	3	26.33	28.37
chipcool0	20082	100	1405	4	6.05	6.14

Table 5.2: Graph Sparsification: Coarsening versus leverage score sampling.

Dataset	m	r	$\frac{nnz(\tilde{K})}{nnz(K)}$	Coarsening		levSamp
				levels	error	error
sprand	1290	332	0.29	2	0.541	0.575
	1951	499	0.28	2	0.542	0.579
	2676	679	0.27	2	0.537	0.580
Maragal4	6005	460	0.11	4	0.416	0.569
rosen1	12599	1738	0.18	3	0.482	0.304
G1	19176	2486	0.14	3	0.549	0.635
bibd13-6	25428	1619	0.08	4	0.901	0.920

Column Subset Selection

In the following experiment, we compare the performance of the coarsening method against the leverage score sampling method for column subset selection. We report results for the same three term-by-document datasets used in the first set of experiments. We also include results obtained for a few sparse matrices from the SuiteSparse matrix collection.

Table 5.1 presents a few comparisons. The errors reported are the Frobenius norm errors $\|A - P_C A\|_F$, where P_C is the projector onto $span(C)$, and C is the coarsened/sampled matrix which is computed by the multilevel coarsening method or using leverage score sampling of A with the top k singular vectors as reported in the second column. Note that $P_C = H_k H_k^T$ from Theorem 6, with $k = c$. The number of columns c

in each test is reported in the third column which is the same for both methods. Recall that for CSSP, the coarsening and sampling algorithms do not perform a post-scaling of the columns that are selected. The multilevel coarsening method performs quite well, yielding results that are comparable with those of leverage score sampling. Recall that the standard leverage score sampling requires the computation of the k top singular vectors which can be substantially more expensive than coarsening especially when k is large.

Graph Sparsification

The next experiment illustrates how coarsening can be used for (spectral) graph sparsification. We again compare the performance of the coarsening approach to the leverage score sampling method [32] for graph spectral sparsification. Recall that spectral sparsification amounts to computing a sparse graph $\tilde{G}$ that approximates the original graph G such that the singular values of the graph Laplacian $\tilde{K}$ of $\tilde{G}$ are close to those of K, Laplacian of G.

Table 5.2 lists the errors obtained when the coarsening and the leverage score sampling approaches were used to compute a sparse graph $\tilde{G}$ for different sparse random graphs and a few matrices related to graphs from the SuiteSparse collection. Given a graph G, we can form a vertex edge incidence matrix B, such that the graph Laplacian of G is $K = B^T B$. Then, sampling/coarsening the rows of B to get $\tilde{B}$ gives us a sparse graph with Laplacian $\tilde{K} = \tilde{B}^T \tilde{B}$. The type of graph or the names are given in the first column of the table and the number of rows m in corresponding vertex edge incidence matrix B is given in the second column. The number of rows r in the coarse matrix $\tilde{B}$ is listed in the third column. The ratios of sparsity in $\tilde{K}$ and K are also given. This ratio indicates the amount of sparsity achieved by sampling/coarsening. Since, we have the same number of rows in the coarsened and sampled matrix $\tilde{B}$, this ratio will be the same for both methods. The error reported is the normalized mean absolute error in the singular values of K and $\tilde{K}$, Error$= \frac{1}{r}\sum_{i=1}^{r} \frac{|\sigma_i(\tilde{K}) - \sigma_i(K)|}{\sigma_i(K)}$, which tells us how close the sparser matrix $\tilde{K}$ is to K spectrally (related to the result in Lemma 3). We see that in all cases but one, the coarsening approach yields a smaller error than with leverage score sampling.

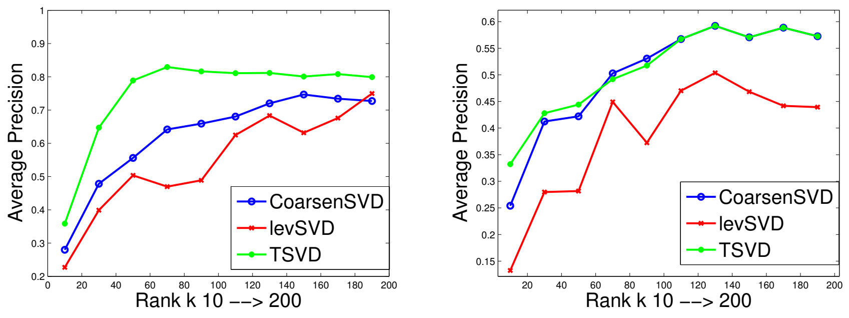


Figure 5.2: LSI results for the MEDLINE dataset (left) and TIME dataset (right).

5.5.1 Applications

In this section, we illustrate the performance of the coarsening technique in the various applications introduced in section 5.2.

Latent Semantic Indexing

The first application we consider is Latent Semantic Indexing (LSI). In LSI, we have a term-document matrix $A \in \mathbb{R}^{m\times n}$, representing m documents and n terms that frequently occur in the documents, where A_{ij} is the frequency of the j-th term in the i-th document. A query is an n-vector $q \in \mathbb{R}^n$, normalized to 1, where the j-th component of a query vector is interpreted as the frequency with which the j-th term occurs in a topic. Typically, the number of topics to which the documents are related is smaller than the number of unique terms n. Hence, finding a set of k topics that best describe the collection of documents for a given k, corresponds to keeping only the top k singular vectors of A, and obtaining a rank k approximation. The truncated SVD and related methods are often used in LSI applications. The argument is that a low rank approximation captures the important underlying intrinsic semantic associated with terms and documents, and removes the noise or variability in word usage. In this experiment, we employ the Coarsen SVD and leverage score sampling SVD algorithms to perform information retrieval techniques by Latent Semantic Indexing (LSI) [101].

Given a term-by-document data $A \in \mathbb{R}^{m\times n}$, we normalize the data using TF-IDF (term frequency-inverse document frequency) scaling. We also normalize the columns to unit vectors. Query matching is the process of finding the documents most relevant to a given query $q \in \mathbb{R}^m$.

Figure 5.2 plots the average precision against the dimension/rank k for MEDLINE and TIME datasets. When the term-document matrix A is large, the computation of the SVD factorization can be expensive for large ranks k. The multi-level techniques will find a smaller set of document vectors, denoted by$A_r \in \mathbb{R}^{m \times n_r}$, to represent A ($n_r < n$). For leverage score sampling, we sample A_r using leverage scores with k equal to the rank shown on the x axis. Just like in the standard LSI, we compute the truncated SVD of $A_r = U_d \Sigma_d V_d^T$, where d is the rank (dimension) chosen. Now the reduced representation of A is $\hat{A} = \Sigma_d^{-1} U_d^T A$. Each query q is transformed to a reduced representation $\hat{q} = \Sigma_d^{-1} U_d^T q$. The similarity of q and a_i are measured by the cosine distance between $\hat{q}$ and $\hat{a}$ for $i = 1, \ldots, n$. This example clearly illustrates the advantage of the coarsening method over randomized sampling and leverage scores. The multilevel coarsening method performs better than the sampling method in this application and in some cases it performs as well as the truncated SVD method.

Genomics - Tagging SNPs

The second application we consider is that of DNA microarray gene analysis. The data from microarray experiments is represented as a matrix $A \in \mathbb{R}^{m \times n}$, where A_{ij} indicates whether the j-th expression level exists for gene i. Typically, the matrix could have entries $\{-1, 0, 1\}$ indicating whether the expression exists (± 1) or not (0) and the sign indicating the order of the sequence, see supplementary material of [105] for details on this encoding. Article [105] used CSSP with a greedy selection algorithm to select a subset of gene expressions or single nucleotide polymorphisms (SNPs) from a table of SNPs for different populations that capture the spectral information (variations) of population. The subset of SNPs are called *tagging SNPs* (tSNPs). Here we show how the coarsening method can be applied in this application to select columns (and thus tSNPs) from the table of SNPs, which characterize the extent to which major patterns of variation of the intrapopulation data are captured by a small number of tSNPs.

We use the same two datasets as in [105], namely the Yale dataset and the Hapmap

Table 5.3: TaggingSNP: Coarsening, Leverage Score sampling and Greedy selection

Data	Size	c	Coarsen	Lev. Samp.	Greedy
Yaledataset/SORCS3	1966×53	14	0.0893	0.1057	0.0494
Yaledataset/PAH	1979×32	9	0.1210	0.2210	0.0966
Yaledataset/HOXB	1953×96	24	0.1083	0.1624	0.0595
Yaledataset/17q25	1962×63	16	0.2239	0.2544	0.1595
HapMap/SORCS3	268×307	39	0.0325	0.0447	0.0104
HapMap/PAH	266×88	22	0.0643	0.0777	0.0311
HapMap/HOXB	269×571	72	0.0258	0.0428	0.0111
HapMap/17q25	265×370	47	0.0821	0.1190	0.0533

datset. The Yale dataset[1]contains a total of 248 SNPs for around 2000 unrelated individuals from 38 populations from around the world. We consider four genomic regions (*SORCS3,PAH,HOXB,* and *17q25*). The HapMap project[2](phase I) released a public database of 1,000,000 SNP typed in different populations. From this database, we consider the data for the same four regions. Using the SNP table, an encoding matrix A is formed with entries $\{-1, 0, 1\}$, with the same meaning of the three possible values as discussed above. We obtained these encoded matrices, made available online by the authors of [105], from `http://www.asifj.org/`.

Table 5.3 lists the errors obtained from the three different methods, namely, Coarsening, Leverage Score sampling and Greedy selection [105] for different populations. If $nnz(X)$ is the number of nonzero elements in a matrix X, the error that is reported is given by $nnz(\hat{A} - A)/nnz(A)$, where A is the input encoding matrix, $\hat{A} = CC^{\dagger}A$, is the projection of A onto the sampled/coarsened C. The greedy algorithm considers each column of the matrix sequentially, projects the remaining columns onto the considered column and chooses the column that gives least error as defined above. The algorithm then repeats the procedure to select the next column and so on. This algorithm is very expensive but it performs rather well in practice. Observe that the coarsening algorithm performs better than leverage score sampling and the performance is comparable with that of the greedy algorithm in some cases. The coarsening approach is less expensive than leverage score sampling which in turn is much less expensive than the greedy algorithm.

[1]`http://alfred.med.yale.edu/`

[2]`https://www.ncbi.nlm.nih.gov/variation/news/NCBI_retiring_HapMap/`

Table 5.4: Multilabel Classification using CSSP (leverage score) and coarsening.

Data	Method	c	Train Err	Train P@k	Test Err	Test P@k
Mediamill, $d = 101, n =$	Coars	51	**10.487**	0.766	**8.707**	**0.713**
$10000, nt = 2001, p = 120$.	CSSP	51	10.520	**0.782**	12.17	0.377
Bibtex, $d = 159, n =$	Coars	80	**1.440**	**0.705**	4.533	**0.383**
$6000, nt = 1501, p = 1836$.	CSSP	80	1.575	0.618	**4.293**	0.380
Delicious, $d = 983, n =$	Coars	246	**50.943**	0.639	**74.852**	0.455
$5000, nt = 1000, p = 500$.	CSSP	246	53.222	**0.655**	77.937	**0.468**
Eurlex, $d = 3993, n =$	Coars	500	2.554	**0.591**	**73.577**	0.3485
$5000, nt = 1000, p = 5000$.	CSSP	500	**2.246**	0.504	81.989	**0.370**

Multilabel Classification

The last application we consider is that of multilabel classification (MLC). As seen in section 5.2, the most common approach to handle large number of labels in this problem is to perform a label dimension reduction assuming a low rank property of labels, i.e., assuming that only a few labels are important. Here, we propose to reduce the label dimension based on hypergraph coarsening. Article [108] presented a method for MLC based on CSSP using leverage score sampling. The idea is to replace sampling by hypergraph coarsening in this method.

Table 5.4 lists the results obtained for MLC when coarsening and leverage score sampling (CSSP) were used for label reduction in the algorithm of [108] on different popular multilabel datasets. All datasets were obtained from `https://manikvarma.github.io/downloads/XC/XMLRepository.html`. The gist of the ML-CSSP algorithm is as follows: Given data with a large number of labels $Y \in \mathbb{B}^{n \times d}$, where $\mathbb{B}$ is a binary field with entries $\{0, 1\}$, the label dimension is reduced by subsampling or coarsening the label matrix leading to $c < d$ labels. The next step is to train c binary classifiers for these reduced c labels. For a new data point, we can predict whether the data-point belongs to the c reduced labels using the c binary classifiers, by getting a c dimensional predicted label vector. We then project the predicted vector onto d dimension and then use rounding to get the final d dimensional predicted vector.

All prediction errors reported (training and test) are Hamming loss errors, i.e., the number of classes for which the predicted label vector differs from the exact label vector.

The second metric used is $Precison@k$, which is a common metric used in the MLC literature [10]. It measures the precision of predicting the first k coordinates $|supp(\hat{y}_{1:k}) \cap supp(y)|/k$, where $supp(x) = \{i|x_i \neq 0\}$. In the above results, we chose $k =$the actual sparsity of the predicted label vector. This is equivalent to checking whether or not the proposed method predicted all the labels the data belongs to correctly. Other values of k such as Precision@k for $k = 1, 3, 5$ are used, where one is checking whether the top 1,3 or 5 labels respectively are predicted correctly, ignoring other and false labels. The better of the two results is highlighted. In this application too, we see that the coarsening method performs well, outperforming the more costly CSSP method in a few cases.

Chapter 6

Dictionary learning via rank shrinkage

6.1 Introduction

We introduced the dictionary learning problem in the first chapter. In this chapter, we explore linear algebra tools for dictionary learning. The dictionary learning problem is popular and several algorithms have been proposed in the literature to address the problem [112, 37, 38]. Most of these algorithms are comprised of two stages: a *sparse coding stage* and a *dictionary update stage*. In the first stage, the dictionary D is fixed and the sparsity constraint is used to compute a sparse linear approximation X for the given signals Y. In the second stage, using the current sparse approximation X (sometimes called codes), the dictionary D is updated such that a certain cost function is minimized. Different cost functions have been used in the literature for the dictionary update stage in order to achieve different objectives. For example, the Frobenius norm with column normalization has been widely used. The dictionary learning methods iterate between the sparse coding stage and the dictionary update stage until convergence. The algorithms differ in the details of the approaches used for estimating X and updating D.

The coherence $\mu(D)$ is a property that characterizes the similarity between different atoms of the dictionary. It is defined as the maximum correlation of any two dictionary atoms

$$\mu(D) = \max_{1\leq i,j\leq N, i\neq j} \mid \langle d_i, d_j \rangle \mid, \tag{6.1}$$

also known as mutual coherence. An alternate measure that can be used to characterize the coherence property of dictionaries, and which is used in this chapter is the average coherence defined as

$$\nu(D) = \max_{1\leq j\leq N} \frac{1}{N-1} \sum_{i=1, i\neq j}^{N} \mid \langle d_i, d_j \rangle \mid . \tag{6.2}$$

A dictionary with small coherence is referred to as an incoherent dictionary. In most applications, the dictionary learned is desired to be incoherent, since incoherent dictionary atoms are distinct and yield better representations of data and signals [113, 114]. In this chapter, we present a new incoherent dictionary learning algorithm based on rank shrinkage.

Related Work: The iterative solution MOD (Method of Optimal Directions) proposed in [112], where a pursuit algorithm is used in the first stage of the iteration, results in a good but a suboptimal solution. Given the set of signals $Y = [y_1, y_2, \ldots, y_N]$, the first stage of MOD consists of estimating the sparse codes x_i, $i = 1, \ldots, N$ that constitute the columns of the matrix X by fixing D and solving

$$\hat{x}_i = \arg\min_{x_i} \parallel y_i - Dx_i \parallel_2; \ \textit{subject to} \parallel x_i \parallel_0 \leq s \quad i = 1, \ldots, N,$$

where $s \ll K$. The vectors x_i are the sparse representations of the signals y_i, and are computed using the current dictionary D. Many sparse coding algorithms have been proposed to solve the above optimization problem, e.g. see [115, 116], that can be used in this stage. The MOD algorithm is perhaps the simplest algorithm that finds D by minimizing $\parallel Y - DX \parallel_F^2$ with respect to D. This amounts to

$$D = \arg\min_{D} \parallel Y - DX \parallel_F^2 = YX^\top \left(XX^\top\right)^{-1}. \tag{6.3}$$

The low coherence of the dictionary can be enforced in the dictionary learning algorithm in two different ways. The first strategy is to add into the dictionary learning problem (6.10), a term that characterizes the incoherence objective of the learned dictionary. This is the approach adopted in [113] where the penalty term $\| G - I \|_F^2$, where $G = D^\top D$ defines the Gram matrix, is used to enforce incoherence. An alternative strategy for learning incoherent dictionaries is to include a decorrelation step after the dictionary update stage to improve the incoherence of the dictionary at each iteration of the algorithm. This is the approach adopted in [114], where the incoherence of the resulting dictionary D from the dictionary update stage is improved by minimizing the Frobenius norm between D and the final dictionary D^*, $\| D - D^* \|_F^2$ subject to the constraint $\mu(D^*) \leq \mu_0$, where μ_0 is the targeted coherence. A similar strategy of including a decorrelation step was proposed in [117], but with a different decorrelation method.

The method presented in this chapter, follows up on the MOD [112] algorithm and proposes an improvement by reducing the coherence of the learned dictionary using a rank shrinkage step. This step is added to the dictionary update stage in order to learn a dictionary with reduced coherence that is adapted to the set of signals Y.

6.2 Incoherence via rank shrinkage

In this section, we present an approach to improve the incoherence of the dictionary learned at each iteration of the algorithm based on rank shrinkage. The new method for rank shrinkage is based on the rank one decomposition of D obtained from (6.3), which we will call D_{OLS}, and nonnegative garrotte estimation problem.

The reduced rank estimation or the rank shrinkage imposes a rank constraint on the dictionary D, by estimating D under the constraint rank$(D) = r$ for $r \leq n$. First, we consider the rank one decomposition based on the singular value decomposition (SVD), and write the ordinary least square solution D_{OLS} in (6.3) as

$$D_{OLS} = \sum_{i=1}^{n} \sigma_i u_i v_i^\top = \sum_{i=1}^{n} \hat{D}_i, \tag{6.4}$$

where σ_i's are the singular values of D, u_i and v_i are the corresponding left and right singular vectors, respectively, and $\hat{D}_i = \sigma_i u_i v_i^\top$, $i = 1, \ldots, n$ are rank one matrices. Hence, the estimate of D with rank $r \leq n$ that minimizes (1.6) (using the Eckart-Young theorem [24]) is given by

$$D_r = \sum_{i=1}^{r} \sigma_i u_i v_i^\top = \sum_{i=1}^{r} \hat{D}_i. \tag{6.5}$$

To obtain an improved reduced rank dictionary, we propose an adaptive shrinkage version of (6.5), based on the extension of (6.4) to

$$D_{sh} = \sum_{i=1}^{n} \alpha_i \hat{D}_i, \tag{6.6}$$

where α_i are weights assigned to the i^{th} component (rank one matrix) $\hat{D}_i$. Next, the idea is to control the weights of all the components using the information of signals Y, and shrink some of the weights to zero, based on a criterion that involves Y. This is achieved by considering the weights α_i as regression coefficients of a certain univariate linear model, and estimate them using a suitable sparsity promoting method. Since the weights α_i can not be negative, we propose to use the nonnegative garrotte [118] method as it is well suited for this situation.

In order to form the nonnegative garrotte, we first form a vector $\bar{y}$ of length $(n * N)$ by vectorizing Y. Similarly, we form n vectors $\bar{z}_i$ of length $(n * N)$ by vectorizing $\hat{D}_i X$, $i = 1, \ldots, n$. Next, we form a matrix Z of size$(n * N) \times n$ by concatenating $\bar{z}_i$. Then, we estimate the sparse weight vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$ using the nonnegative garrotte [118] defined as

$$\begin{aligned} \boldsymbol{\alpha} &= \arg\min_{\boldsymbol{\alpha}} \parallel \bar{y} - Z\boldsymbol{\alpha} \parallel_2^2 + \lambda \sum_{i=1}^{n} \alpha_i \quad \text{subject to} \quad \alpha_i \geq 0 \\ &= \arg\min_{\boldsymbol{\alpha}} \parallel \bar{y} - \sum_{i=1}^{n} \bar{z}_i \alpha_i \parallel_2^2 + \lambda \sum_{i=1}^{n} \alpha_i \quad \text{subject to} \quad \alpha_i \geq 0. \end{aligned} \tag{6.7}$$

The optimization problem in (6.7) can be solved using a path-wise coordinate descent approach. The coordinate-wise update for the weight vector $\boldsymbol{\alpha}$ will be of form

$$\alpha_i = \left(\frac{\bar{z}_i^\top \bar{y}_i - \lambda}{\bar{z}_i^\top \bar{z}_i} \right)_+, \; i = 1, \ldots, n \tag{6.8}$$

where $\bar{y}_i = \bar{y} - \sum_{j=1, j \neq i}^{n} \bar{z}_i \alpha_i$. The parameter λ can be tuned to obtained the desired rank shrinkage in the dictionary D, and in turn control its coherence.

Algorithm 5 Stepwise description of the proposed dictionary learning algorithm

```
Input: $Y$ , $D_{ini}$, $s$, $\lambda$, $\varepsilon$ and $J$.
Output: $D, X$.
Set $D = D_{ini}$.
For $it = 1$ to $J$
1. Sparse Coding Stage:
      Find sparse coefficients $X$, by approximately solving
         $\hat{x}_i = \arg\min_{x_i} \| y_i - Dx_i \|^2;$ subject to $\| x_i \|_0 \leq s \quad i = 1, \ldots, N.$
2. Dictionary Update Stage:
      Generate the OLS solution $D_{OLS} = YX^\top \left(XX^\top\right)^{-1}$.
      2.a: Using the SVD compute $D_{OLS} = \sum_{i=1}^{n} \hat{D}_i$.
      2.b: Construct the vectors $\bar{y}$ and $\bar{z}_i$, $i = 1, \ldots, n$.
      2.c: Estimate the weights $\alpha_i$, $i = 1, \ldots, n$ as
      While $||\boldsymbol{\alpha}^{iter} - \boldsymbol{\alpha}^{iter-1}||_2^2 \geq \varepsilon$ do
         For $i = 1$ to $n$
         Compute $\bar{y}_i = y - \sum_{j=1, j \neq i}^{n} \bar{z}_i \alpha_i$
         Compute the components $\alpha_i = \left(\frac{\bar{z}_i^\top \bar{y}_i - \lambda}{\bar{z}_i^\top \bar{z}_i}\right)_+$
         end
      $iter = iter + 1$
      end while
      2.d: Form $D = \sum_{i=1}^{n} \alpha_i \hat{D}_i$.
end
```

Algorithm: The proposed algorithm is summarized in Algorithm 5, which includes the rank shrinkage step to obtain a more stable and incoherent dictionary D.

6.3 Analysis

The purpose of the rank shrinkage step in Algorithm 5 is to remove collinearity between the atoms (decorrelate). In this section, we show that the rank shrinkage step indeed reduces the coherence of the dictionary.

Proposition 1 *The rank shrinkage step in Algorithm 5 reduces the mutual coherence $\mu(D)$ of the learned dictionary.*

The following Lemma which gives a relation between the smallest non zero singular value and the mutual coherence of a matrix, and the arguments that follow this lemma provide the theoretical justification for the above proposition.

Lemma 4 *Given a dictionary $D = [d_1, d_2, \ldots, d_K]$, such that $\|d_k\| = 1$, $k = 1, \ldots, K$ and $|\langle d_i d_j \rangle| < 1$ for any $i \neq j$, Let us define* $\inf(D) := \min_{t \notin Null(D), \|t\|=1} \|Dt\|$. *Suppose that*

$$\inf(D) = \min_{t \notin Null(D), \|t\|=1} \|Dt\| \geq \sqrt{1-\eta}.$$

Then we have $\mu(D) \leq \eta$.

Proof. For the given dictionary D, let $G = D^\top D$ define the corresponding Gram matrix. Let us write $G = I - H$. The diagonal entries of H are zero. Then, for any vector t of norm 1, with $t \notin Null(D)$, we have

$$\|Dt\|^2 = \langle D^T D t, t \rangle = \langle (I - H)t, t \rangle = 1 - \langle Ht, t \rangle \geq 1 - \eta.$$

As a result, for any vector $t \notin Null(D)$:

$$\langle Ht, t \rangle \leq \eta. \tag{6.9}$$

Since the mutual coherence is the maximum off-diagonal entry of G, we are interested in the off-diagonal entries h_{ij} (which are $-g_{ij}$) for $i \neq j$. Let i, j be any pair with $i \neq j$ and take the vector

$$t_{ij} = \frac{1}{\sqrt{2}}[e_i + \sigma e_j] \text{ with } \sigma = \text{sign}(h_{ij}).$$

Then a little calculation shows that

$$\langle Ht_{ij}, t_{ij} \rangle = |h_{ij}|.$$

Next, consider

$$\|Dt_{ij}\|^2 = \frac{1}{2}\|d_i \pm d_j\|^2 = \frac{1}{2}[\|d_i\|^2 + \|d_j\|^2 \pm 2\langle d_i, d_j \rangle] = 1 \pm \langle d_i, d_j \rangle > 0.$$

This is due to the assumption that $|\langle d_i, d_j \rangle| < 1$ for $i \neq j$. Therefore, t_{ij} does not belong to the null space of D, and from (6.9) we get $|h_{ij}| < \eta$. This proves the lemma. $\square$

Remark 4 *If we assume that* $\min_{t \notin Null(D), \|t\|=1} \|Dt\| \geq 1 - \eta$*, then we would get a bound of the form* $\mu(D) \leq \eta(2 - \eta)$*.*

Observe that in Lemma 4, the term $\inf(D)$ is the smallest non-zero singular value of the dictionary D. The argument is as follows: If $\inf(D)$ increases after rank shrinkage, one can choose a lower value for the parameter η in Lemma 4 such that $\inf(D) \geq \sqrt{1-\eta}$, still holds. Then, by the statement of Lemma 4, $\mu(D) \leq \eta$ satisfies for this lower value of η, which means $\mu(D)$ must reduce. Therefore, we need to show that the smallest non-zero singular value of the dictionary increases after our rank shrinkage step, for the coherence of the dictionary to decrease.

Let $\sigma_i, i = 1, \ldots, n$ be the singular values of the dictionary D before rank shrinkage labeled decreasingly, then $\inf(D) = \sigma_n$. The singular values of the dictionary D_r after rank shrinkage will be $\alpha_i \sigma_i, i = 1, \ldots, r$. Thus, for Proposition 1 to hold, we need

$$\alpha_r \sigma_r > \sigma_n.$$

Clearly, $\sigma_r > \sigma_n$, and the coefficients α_i's are chosen such that they penalize only the smaller singular values (corresponding to near collinear subspaces). For the relevant atoms in the dictionary, the inner product $\bar{z}_i^\top \bar{y}_i$ in the updates of the coefficients α_i (eq. (6.8)), the coefficients which control the rank) will be high, and α_i will be large. So, such atoms are not removed (instead are promoted). Hence, we suggest to use a small tuning parameter λ in eq. (6.8). More importantly, since *the overall energy of the dictionary needs to be constant before and after rank shrinkage* since the columns of the dictionaries must have unit norms, the nonzero singular values of the rank shrunk matrix must be relatively higher to balance the energy from the singular values that were removed. This justifies Proposition 1.

A reduction in $\mu(D)$ will likely reduce the average coherence $\nu(D)$. We have the following relation between the two measures : $\nu(D) \leq \frac{K}{K-1}\mu(D)$. (If the maximum value of correlation is reduced, the maximum of the average of correlations is likely to reduce.)

Coherence and the eigenvalues of the Gram matrix: In section 6.2, we saw how the eigenvalues of the Gram matrix $D^\top D$ that are close to zero (due to near collinearity) results in poor least squares solutions. The following argument gives further insight on the relation between the eigenvalues of the Gram matrix (which are squares of the singular values of the dictionary), collinearity, least squares solutions and mutual coherence.

Consider the sparse coding stage (updating X). The columns of X in the sparse coding stage are obtained as the solutions of

$$\hat{x}_i = \arg\min_x \|y_i - D_{Ii}x\|_2^2 \qquad i = 1, ..., N,$$

and their variance-covariance matrix is given by

$$\operatorname{Var}(\hat{x}_i) = \sigma^2 \left(D_{Ii}^\top D_{Ii}\right)^{-1} = \sigma^2 U_{Ii} \Lambda_{Ii}^{-1} U_{Ii} \qquad i = 1, ..., N, \tag{6.10}$$

where Ii represents the set of selected atoms with K_i cardinality, σ^2 is the noise variance and $D^\top D$ is the Gram matrix with eigen-decomposition $D^\top D = U\Lambda U^\top$ and rank r. We will drop the index i below.

The variance of each component of $\hat{x}_i$ is given by

$$\operatorname{Var}(\hat{x}_{i_j}) = \sigma^2 \sum_{l=1}^{r} \frac{u_{jl}^2}{\lambda_l} = \sigma^2 \sum_{l=1}^{r} \frac{a_{jl}^2}{\lambda_l^2},$$

where u_{jl} is the l^{th} component of u_j (column of U) and $a_{jl} = u_{jl}\lambda_l$ is the l^{th} component of $a_j = u_j\Lambda$. Then, the total variance is given by

$$\sum_{j=1}^{r} \operatorname{Var}(\hat{x}_{i_j}) = \sigma^2 \sum_{j=1}^{r}\sum_{l=1}^{r} \frac{a_{jl}^2}{\lambda_l^2} = \sigma^2 \sum_{l=1}^{r}\sum_{j=1}^{r} \frac{a_{jl}^2}{\lambda_l^2} = \sigma^2 \sum_{l=1}^{r} \frac{1}{\lambda_l}. \tag{6.11}$$

Now we observe that, if any one of the eigenvalues is very close to zero, the mean of the variances of the estimated sparse codes will increase to a very large extent. We saw in section 6.2 that, the eigenvalues are close to zero if the correlations between the dictionary atoms are high, that is, it depends on the extent of multicollinearity.

If the eigenvalues are decreasingly ordered $\lambda_1 \geq \ldots \geq \lambda_r$, then we have the following relations:

$$\sum_{l=1}^{r} \frac{1}{\lambda_l} \leq \sum_{l=1}^{r} \frac{1}{\lambda_r} = \frac{r}{\lambda_r}$$

and

$$\mathrm{Var}(\hat{x}_{i_j}) = \sigma^2 \left(D_{Ii}^\top D_{Ii} \right)^{-1}_{jj} \geq \sigma^2,$$

which lead to

$$r\sigma^2 \leq \sum_{j=1}^{r} \mathrm{Var}(\hat{x}_{i_j}) \leq \frac{r\sigma^2}{\lambda_1}.$$

If all the dictionary atoms are orthogonal (incoherent), the above inequality becomes an equality, since in this case, all eigenvalues are 1. If any one of the eigenvalues is close to zero, the variances of the estimated sparse codes will be very large according to (6.11). Hence, if the correlation among the dictionary atoms is considerable (coherent), the variance of the sparse code coefficients will be too large due to the inversion formula given in (6.10) (the analysis of the individual effect of the variables will be less meaningful here). Hence, it is necessary to quantify multicollinearity.

Incoherence and the mutual coherence $\mu(D)$ measure characterize the departure from orthogonality. If the dictionary atoms are strongly coherent, the matrix $D_{Ii}^\top D_{Ii}$ is ill-conditioned, and while the least squares estimator still exists, it will be very unstable (due to large variance). Recall that, the smallest eigenvalue of D_{Ii} is smaller than that of D by the interlacing theorem. Therefore, reducing the rank , i.e., a smaller r, will mean a larger nonzero smallest eigenvalue λ_r (in turn larger $\inf(D)$) and a reduced variance, and will result in more stable least squares solution.

6.4 Numerical Experiments

In this section, we present two numerical examples to demonstrate the performance of the proposed method. We consider the dictionary recovery problem of a small random dictionary with uniform distribution. We generate $N = 2000$ signals Y from a random dictionary with uniform distribution D with $n = 20, k = 50$ and $s = 4$. We add a small amount of noise to the signals Y (the strength of the signal is $SNR = 20dB$). The objective is to recover a dictionary $\tilde{D}$ from the signals Y that is as close to the original dictionary D as possible using the dictionary learning (DL) algorithms. Figure 6.1 compares the results obtained for the five different dictionary learning (DL) algorithms considered.

The left plot in figure 6.1 plots the number of dictionary atoms (averaged over 10

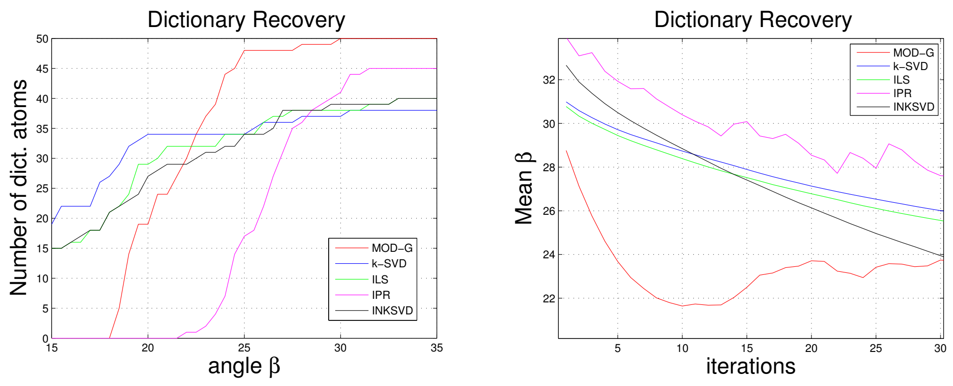


Figure 6.1: Recovery of a random dictionary with uniform distribution.

trials) in the dictionary recovered that are within the given angle β (in degrees) from the original dictionary atoms for the five different dictionary learning algorithms. The angle β is defined as

$$\beta_i = \frac{\cos^{-1}(\langle \tilde{d}_i, d_i \rangle)}{\|\tilde{d}_i\|_2 \|d_i\|_2},\ i = 1, \ldots, K.$$

All algorithms were run for 50 iterations and 10 trials. For the proposed algorithm (MOD-G, for MOD+garotte) we set $\lambda = 10$, and for INKSVD and IPR we set $\mu_0 = 0.8$ (the parameters were tuned until we obtain similar average coherence by all the three methods). In all the five algorithms, for the sparse coding stage, we use the Orthogonal Matching Pursuit (OMP) algorithm [116], and we initialize the dictionary D_{ini} with randomly chosen samples of Y. The right plot in figure 6.1 plots the mean angle β of the dictionaries obtained at each iteration for the different DL algorithms. We observe that the proposed algorithm requires fewer iterations (converges faster) to recover a closer dictionary than the other algorithms.

Audio signal representation: In the next experiment, we assess the performance of the proposed DL algorithm in recovering a set of audio signals, such that the dictionary has bounded average coherence and provides good approximation to the signals. For this, we will consider the same dataset considered in the INKSVD paper [114] and the IPR paper [117], and compare the performances of the three incoherent DL algorithms. The audio signals considered are excerpts of a $16kHz$ guitar recording named

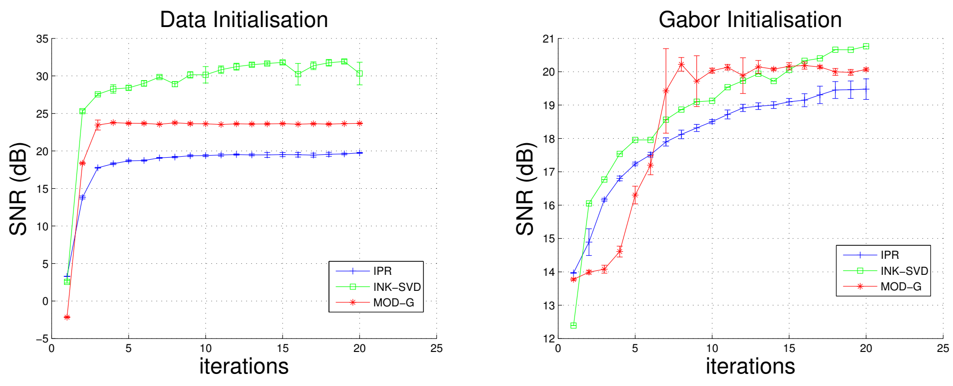


Figure 6.2: Audio signal representations.

`music03_16kHz`, and is part of the data available in the SmallBox[1] toolbox. The toolbox also contains all the codes to reproduce the following experiment (expect the code for the proposed algorithm).

We consider exactly the same experiment demonstrated in [114, 117], where the recording is divided into 50% overlapping blocks of 256 samples (16 ms each) and the resulting signals is arranged as columns of the signal matrix Y. The size of Y is 256×624. Next, the dictionary is initialized to be twice overcomplete, ie., the size of the dictionary is 256×512. The three incoherent DL algorithms (MOD-G, IPR and INKSVD) were run for 20 iterations, setting $s = 12$ non-zero coefficients in each sparse representation. In IPR and INKSVD, we set $\mu_0 = 0.5$ (for values below this, the performances of all three methods were poor and were hard to compare), and in MOD-G we set $\lambda = 16$, such that all three methods give $\nu(D)$ around 0.37 after 20 iterations. Figure 6.2 depicts the performances of the three methods when the dictionaries were initialized using i) a randomly chosen subset of the training data and ii) Gabor dictionary [38].

The left plot of figure 6.2 plots the SNR of the recovered signal obtained by the three incoherent DL algorithms for each of iterations, when the dictionary was initialized using a randomly chosen subset of the training data. The right plot gives the SNR v/s iterations plot for the three algorithms for Gabor dictionary initialization. The

[1]SmallBox is an open source MatLab toolbox, containing codes and datasets for testing and benchmarking various dictionary learning algorithms `http://www.small-project.eu/software-data/smallbox/`.

experiments were run over 5 independent trials. The plots give the averages and the standard deviations of the SNRs obtained at each iteration over the different trials. We observe that the proposed method is better than IPR, but the performance of INKSVD is superior to the other two methods.

However, we observe that each iteration of MOD-G is inexpensive compared to IPR and INKSVD. The 20 iterations of MOD-G algorithm took on an average 320 secs, averaged over the 10 (5+5) trials, to run on a 3.3 GHz Intel Core i5 machine executed using Matlab R2013a and using `cputime` function. The 20 iterations of IPR took a total of 1110 secs on average to run. IPR was over 3 times slower than MOD-G, this is because in each iteration of IPR, we need to compute an eigenvalue decomposition of a coherent constrained Gram matrix to threshold the eigenvalues, and also compute an SVD to rotate the dictionary. There are many additional matrix-matrix products in an IPR iteration compared to MOD-G. The 20 iterations of INKSVD took an average of 2067 secs. INK-SVD takes longer (almost 7 times longer than MOD-G) to compute less coherent dictionaries. This is because INK-SVD acts in a greedy fashion by decorrelating pair of atoms until the target mutual coherence is reached (or until a maximum number of iterations) and therefore the number of pairs of atoms to decorrelate increases for low values of the target coherence. For additional results and timing comparisons between IPR and INKSVD, see [117]. Hence, INKSVD gives a superior performance at the expense of higher per iteration runtime cost. The proposed DL algorithm produces comparable results, and is significantly faster (in terms of per iteration runtime) compared to the other two incoherent DL algorithms.

Part II

Applications of Coding Theory

Chapter 7

Low rank approximation - Codes for sampling

7.1 Introduction

Many scientific computations, signal processing, data analysis and machine learning applications lead to large dimensional matrices that can be well approximated by a low dimensional (low rank) basis. It is more efficient to solve many computational problems by first transforming these high dimensional matrices into a low dimensional space, while preserving the invariant subspace that captures the essential information of the matrix. Several algorithms have been proposed in the literature for finding low rank approximations of matrices [22, 67, 96]. Recently, research focussed on developing techniques that use randomization for computing low rank approximations and decompositions of such large matrices [67]. It was found that randomness provides an effective way to construct low dimensional bases with high reliability and computational efficiency. Similar ideas based on random sampling have been proposed in the recent literature for solving least squares (ℓ_2) linear regression problems [119, 120, 67].

Randomization techniques for matrix approximations aim to compute a basis that approximately spans the range of an $m \times n$ input matrix A, by sampling the matrix A using random matrices, e.g. i.i.d Gaussian [67]. This task is accomplished by first forming the matrix-matrix product $Y = A\Omega$, where Ω is an $n \times \ell$ random matrix of smaller dimension $\ell \ll \{m, n\}$, and then computing the orthonormal basis of $Y = QR$

that identifies the range of the reduced matrix Y. It can be shown that $A \approx QQ^\top A$ with high probability. It has been shown that structured random matrices, like subsampled random Fourier transform (SRFT) and Hadamard transform (SRHT) matrices can also be used in place of fully random matrices [121, 120].

The input matrices, whose low rank approximation is to be computed, usually have very large dimensions (e.g., in the order of $10^6 - 10^9$ [67, 122]). In order to form a Gaussian (a fully) random matrix that samples the input matrix, we need to generate a large quantity of random numbers. This could be a serious practical issue (in terms of time complexity and storage). This issue can be addressed by using the structured random matrices, like SRFT and SRHT matrices. An important practical advantage of using these structured random matrices is that their structure allows the computation of matrix-matrix product at a cost of $O(mn \log_2 \ell)$ making the algorithms fast (also known as fast transforms) for general dense input matrices. However, with these matrices, mixing of columns might not be as uniform (some of the entries of the Fourier transform might have very large magnitude), and there is potential loss in the accuracy (i.e., the performance of SRHT/SRFT matrices in place of Gaussian matrices with same number of samples ℓ (or even slightly higher) is worse).

Another drawback with fast transforms is that for parallel and distributed applications, particularly when the input matrix is sparse and/or its columns are distributively stored, it is found that FFT-like algorithms are significantly slower due to communication issues or other machine related issues (machines are optimized for matrix-vector operations) [122]. Also for a rank-k approximation, these matrices require sampling $\ell = O(k \log k)$ columns. Other practical issues arise such as: the Fourier Transform matrices require handling complex numbers and the Hadamard matrices exist only for the sizes which are in powers of 2. All these drawbacks can be overcome if the code matrices presented in this chapter are used for sampling the input matrices.

In digital communication, information is encoded by adding redundancy to predominantly binary vectors or codewords, that are then transmitted over a noisy channel [40], (introduced in the first chapter). These codewords are required to be far apart in terms of some distance metric for noise-resilience. Coding schemes usually generate codewords that maintain a fixed minimum Hamming distance between each other, hence they are widespread and act like random vectors. We can define probability measures for matrices formed by stacking up these codewords as seen in the first chapter. Here, we explore the idea of using subsampled versions of these code matrices as sampling (sketching) matrices in the randomized techniques for matrix approximations.

Similar to Fourier and Hadamard sampling matrices, fast multiplication is possible with code matrices from certain class of codes due to their structure. Hence, fast approximations can be achieved for general dense input matrices, since the matrix-matrix product $A\Omega$ can be computed in $O(mn\log_2 \ell)$ cost with such code matrices. In addition, the shortcomings of SRFT/SRHT matrices in parallel and distributed environments, and in data streaming models can be overcome by using code matrices. For certain code matrices, the logarithmic factor in the number of samples is not required. This is a significant theoretical result that shows that order optimality can be achieved in the number of samples required with partially random matrices.

One of the key applications where the randomized approximation (or sketching) algorithms are used is in approximately solving overdetermined least squares regression problem faster [119, 120, 67]. Here, we are given a matrix $A \in \mathbb{R}^{n\times d}$ and a vector $b \in \mathbb{R}^n$, with $n \gg d$. The goal is to solve the least squares regression problem $\min_x \|Ax - b\|_2$ faster, (where $\|.\|_2$ is ℓ_2 norm) and output a vector x' such that, with high probability,

$$\|Ax' - b\|_2 \leq (1+\epsilon)\|A\hat{x} - b\|_2,$$

where $\hat{x}$ is the ℓ_2 minimizer given by the Moore-Penrose pseudo inverse of A, i.e., $\hat{x} = A^{\dagger}b$ [56]. For details on the applications where we encounter such extremely overdetermined linear system of equations, we refer to [122]. The idea of randomized approximation is to use a sampling (sketching) matrix to reduce the dimensions of A and b, and then solve the smaller problem to obtain x'.

In this chapter, we advocate the use of error correcting coding matrices for randomized sampling of large matrices in low rank approximations and other applications, and show how specific classes of code matrices can be used in different computational environments to achieve the best results possible (amongst the existing sampling matrices).

The key theoretical result we present is that, the $(1+\epsilon)$ optimal (Frobenius and spectral norms) error bounds can be achieved with $O(k/\epsilon)$ samples when certain classes of code matrices (code matrices with dual distance $> k$) are used for sampling. We discuss how different classes of code matrices with desired properties (for sampling) can be generated and used in practice. We also discusses the advantages of code matrices over other classes of sampling matrices in the parallel and distributed environments, and also in the data streaming models. With the sizes of the datasets increasing rapidly, we believe such advantages of code matrices become significantly important, making them more appealing for a range of applications where such randomized sampling is used.

7.2 Construction of subsampled code matrix

For an input matrix A of size $m \times n$ and a target rank k, we choose $r \geq \lceil \log_2 n \rceil$ as the dimension of the code (length of the message vector) and $\ell > k$ as the length of the code. The value of ℓ will depend on the coding scheme used, particularly on the dual distance of the code. We consider an $[\ell, r]$-linear coding scheme and form the sampling matrix as follows: We draw the sampling test matrix say Ω as

$$\Omega = \sqrt{\frac{2^r}{\ell}} DS\Phi, \tag{7.1}$$

where

- D is a random $n \times n$ diagonal matrix whose entries are independent random signs, i.e., random variables uniformly distributed on $\{\pm 1\}$.
- S is a uniformly random downsampler, an $n \times 2^r$matrix whose n rows are randomly selected from a $2^r \times 2^r$ identity matrix.
- Φ is the $2^r \times \ell$ code matrix, generated using an $[\ell, r]$-linear coding scheme, with BPSK mapping and scaled by $2^{-r/2}$ such that all columns have unit norm.

Intuition The design of a Subsampled Code Matrix (SCM) is similar to the design of SRFT and SRHT matrices. The intuition for using such a design is well established in [123, 67]. The matrix Φ has entries with magnitude $\pm 2^{-r/2}$ and has orthonormal columns when a coding scheme with dual distance of the code ≥ 3 is used.

The scaling $\sqrt{\frac{2^r}{\ell}}$ is used to make the energy of the sampling matrix equal to unity, i.e., to make the rows of Ω unit vectors. The objective of multiplying by the matrix D is twofold. The first purpose is to flatten out the magnitudes of input vectors, see [123] for the details. For a fixed unit vector $\boldsymbol{x}$, the first component of $\boldsymbol{x}^\top DS\Phi$ is given by $(\boldsymbol{x}^\top DS\Phi)_1 = \sum_{i=1}^n x_i\varepsilon_i\phi_{j1}$, where ϕ_{j1} are components of the first column of the code matrix Φ, with the indices j's are such that $S_{ij} = 1$ for $i = 1, \dots, n$ and ε_i is the Rademacher variable from D. This sum has zero mean and since entries of Φ have magnitude $2^{-r/2}$, the variance of the sum is 2^{-r}. The Hoeffding inequality shows that

$$\mathbb{P}\{|(\boldsymbol{x}^\top DS\Phi)_1| \geq \tilde{t}\} \leq 2e^{-2^r\tilde{t}^2/2}.$$

That is, the magnitude of the first component of $\boldsymbol{x}^\top DS\Phi$ is about $2^{-r/2}$. Similarly, the argument holds for the remaining entries. Therefore, it is unlikely that any one of the ℓ components of $\boldsymbol{x}^\top DS\Phi$ is larger than $\sqrt{4\log \ell/2^r}$ (with a failure probability of $2\ell^{-1}$).

The second purpose of multiplying by D is as follows: The code matrix Φ with a dual distance $> k$ forms a deterministic k-wise independent matrix. Multiplying this Φ matrix by D (with independent random signs on the diagonal) results in a k-wise independent random matrix. Note that uniform downsampling of the matrix will not affect this property. Hence, the subsampled code matrix SCM Ω will be a k-wise independent random matrix.

The downsampler S is a formal way of saying, if $n < 2^r$, we choose n out of 2^r possible codewords to form the sampling matrix Ω. Uniform downsampling is used in the theoretical analysis to get an upper bound for the singular values of Ω. In practice, we choose n numbers between 1 to 2^r, use the binary representation of these numbers as the message vectors (form M) and use the generator matrix G of the coding scheme selected to form the sampling matrix Ω, using (1.7) and BPSK mapping. For dense input matrices, it is advantageous to choose these numbers (message vectors) to be 1 to $2^{\lceil \log_2 n \rceil}$, to exploit the availability of fast multiplication.

7.2.1 Algorithm

We use the same prototype algorithm as discussed in [67] for the low rank approximation and decomposition of an input matrix A. The subsampled code matrix (SCM) Ω given in (7.1), generated from a chosen coding scheme is used as the sampling test matrix. The algorithm is as follows:

Algorithm 6 Prototype Algorithm

```
Input: An $m \times n$ matrix $A$, a target rank $k$.
Output: Rank-$k$ factors $U, \Sigma$, and $V$ in an approximate SVD $A \approx U\Sigma V^\top$.
1. Form an $n \times \ell$ subsampled code matrix $\Omega$, as described in (7.1), using an $[\ell, r]-$linear coding scheme, where $\ell > k$ and $r \geq \lceil \log_2 n \rceil$.
2. Form the $m \times \ell$ sample matrix $Y = A\Omega$.
3. Form an $m \times \ell$ orthonormal matrix $Q$ such that
   $Y = QR$.
4. Form the $\ell \times n$ matrix $B = Q^\top A$.
5. Compute the SVD of the small matrix $B = \hat{U}\Sigma V^\top$.
6. Form the matrix $U = Q\hat{U}$.
```

Computational cost Many, if not most of the structured codes can be decoded using the Fast Fourier Transform (FFT). The corresponding $2^r \times \ell$ code matrix Φ of such structured codes (after BPSK mapping) will have every column of Φ equal to some column of a $2^r \times 2^r$ Hadamard matrix, see definition 2.2 in [124]. Hence, for a general dense matrix in RAM, the matrix-matrix product $Y = A\Omega$ with these structure code matrices can be computed in $O(mn \log_2 \ell)$ time using the 'Trimmed Hadamard transform' technique described in [124].

Fast multiplications are possible with matrices from another class of codes known as cyclic codes. In cyclic codes, a circular shift of a codeword results in another codeword of that code. So, a $2^r \times \ell$ code matrix Φ generated using an $[\ell, r]$-cyclic code scheme will consist of $2^r/\ell$ blocks of circulant matrices of size $\ell \times \ell$ (when appropriately rearranged). It is known that the matrix-vector products with circulant matrices can be computed in $O(\ell \log_2 \ell)$ operations via FFT [56]. So, for a general dense input matrix, the matrix-matrix product $Y = A\Omega$ with such cyclic code matrices can be computed in $O(mn \log_2 \ell)$ time.

7.3 Analysis

This section discusses the performance (error) analysis of the subsampled code matrices (SCM) as sampling matrices in Algorithm 6. We will prove that an approximation error of $(1+\epsilon)$ times the best rank-k approximation (Frobenius norm error) possible for a given matrix A can be achieved with code matrices. That is,

$$\|A - \hat{A}_k\|_F \leq (1+\epsilon)\|A - A_k\|_F,$$

where $\hat{A}_k$ is the rank-k approximation obtained from Algorithm 6 and A_k is the best rank-k approximation. In order to prove this, we show that SCM satisfies two key properties, the Johnson Lindenstrauss Transforms (JLT) and the subspace embedding properties via the k-wise independence property of the codes. We also derive the bounds for the spectral norm error and the singular values obtained, based on the deterministic error bounds in the literature for the algorithm for a given sampling matrix Ω.

7.3.1 Setup

Let A be an $m \times n$ input matrix with SVD given by $A = U\Sigma V^\top$, and partition its SVD as follows

$$A = U \begin{array}{cc} k & n-k \\ \left[\begin{array}{cc} \Sigma_1 & \\ & \Sigma_2 \end{array}\right. & \left.\right] \end{array} \begin{array}{cc} n & \\ \left[\begin{array}{c} V_1^\top \\ V_2^\top \end{array}\right] & \begin{array}{c} k \\ n-k \end{array} \end{array}. \tag{7.2}$$

Let Ω be the $n \times \ell$ test (sampling) matrix, where ℓ is the number of samples. Consider the matrices

$$\Omega_1 = V_1^\top \Omega \quad \text{and} \quad \Omega_2 = V_2^\top \Omega. \tag{7.3}$$

The objective of any low rank approximation algorithm is to approximate the subspace that spans the top k left singular vectors of A. Hence, for a given sampling matrix Ω, the key challenge is to show that Ω_1 is full rank. That is, we need to show that for any orthonormal matrix V of dimension k, with high probability $V^\top\Omega$ is well conditioned [67]. This is true if the test matrix Ω satisfies the subspace embedding property, and it is said to preserve the geometry of an entire subspace of vectors V.

7.3.2 Subsampled code matrices, JLT and subspace embedding

Here, we define the two key properties which will be used frequently in our theoretical analysis. First is the Johnson-Lindenstrauss Transform (JLT) [125] which played a key role in the development of embedding-based randomized sampling. Sarlos [126] developed the important relation between JLT and random matrix sampling (also known as subspace embedding). The JLT property is defined as [126]:

Definition 1 (Johnson-Lindenstrauss Transform) *A matrix $\Omega \in \mathbb{R}^{n\times\ell}$ forms a Johnson-Lindenstrauss Transform with parameters ϵ, δ, d or JLT(ϵ, δ, d) for any $0 < \epsilon, \delta < 1$, if for any d-element set $V \subset \mathbb{R}^n$, and for all $v \in V$ it holds*

$$(1-\epsilon)\|v\|_2^2 \leq \|\Omega^\top v\|_2^2 \leq (1+\epsilon)\|v\|_2^2$$

with probability $1-\delta$.

The other key property which the code matrices need to satisfy is the subspace embedding property defined below.

Definition 2 (Subspace Embedding) *A matrix $\Omega \in \mathbb{R}^{n\times\ell}$ is a $(1\pm\epsilon)$ ℓ_2-subspace embedding for the row space of an $m\times n$ matrix A, if for an orthonormal basis $V \in \mathbb{R}^{n\times k}$ that spans the row space of A, for all $x \in \mathbb{R}^k$*

$$\|\Omega^\top Vx\|_2^2 = (1\pm\epsilon)\|Vx\|_2^2 = (1\pm\epsilon)\|x\|_2^2,$$

where $\|\Omega^\top Vx\|_2^2 = (1\pm\epsilon)\|x\|_2^2$ stands for $(1-\epsilon)\|x\|_2^2 \leq \|\Omega^\top Vx\|_2^2 \leq (1+\epsilon)\|x\|_2^2$.

The above definition is useful when the sampling is achieved column-wise. A similar definition for row-wise sampling holds for an orthonormal matrix $U \in \mathbb{R}^{m\times k}$ which spans the column space of A, see [28]. The above definition simplifies to the following condition:

$$\|V^\top\Omega\Omega^\top V - I\|_2 \leq \epsilon. \tag{7.4}$$

The matrix Ω with subspace embedding property, satisfying the above condition is said to approximately preserve the geometry of an entire subspace of vectors [123].

Recall the construction of the 'tall and thin' $n \times \ell$ subsampled error correcting code matrix Ω. The critical requirement to prove the $(1+\epsilon)$ optimal error bound is to show that these matrices satisfy the two key properties: JLT and subspace embedding. The subspace embedding property will also imply that Ω_1 will be full rank, which will enable us use the deterministic bounds developed in the literature to derive the bounds for the spectral norm error and the singular values obtained.

Johnson-Lindenstrauss Transform

We saw the definition of JLT above, which says that a matrix Ω that satisfies JLT(ϵ, δ, d) preserves the norm for any vector v in a d-element subspace $V \subset \mathbb{R}^n$. We will use two key results developed in the literature to show that code matrices with certain mild properties satisfy the JLT property.

The first result is by Ailon and Liberty [124], where they show a matrix Ω which is 4-wise independent will satisfy the JLT property, see Lemma 5.1 in [124]. Interestingly, they give the 2 error correcting dual BCH codes as examples for such 4-wise independent matrices and also demonstrate how fast multiplications can be achieved with these code matrices. However, a minor drawback with using 4-wise independent matrices is that the maximum entries of A need to be restricted.

The second (stronger) result is by Clarkson and Woodruff [119] (see Theorem 2.2), where they show if Ω is a $4\lceil \log(\sqrt(2)/\delta)\rceil$-wise independent matrix, then Ω will satisfy the JLT property. Recall that the SCM matrix Ω defined in eq. (7.1) will be a random k-wise independent matrix if the dual distance of the code is $> k$. Thus, any error correcting code matrix with a dual distance > 4 (more than 2 error correcting ability) will satisfy the JLT property.

One of the important results related to JLT that is of interest for our theoretical analysis is the matrix multiplication property. This is defined in the following lemma, which is Theorem 2.8 in [28]. We can see similar results in Lemma 6 in [126] and Theorem 2.2 in [119].

Lemma 5 *For $\epsilon, \delta \in (0, 1/2)$, let Ω be a random matrix (or from a distribution $\mathcal{D}$) with n rows that satisfies (ϵ, δ, d)-JLT property. Then for A, B matrices with n rows,*

$$\mathbf{Pr}\left[\|A^\top B - A^\top \Omega \Omega^\top B\|_F \leq 3\epsilon \|A\|_F \|B\|_F\right] \geq 1 - \delta. \tag{7.5}$$

We will see that the above lemma is one of the two main ingredients required to prove $(1+\epsilon)$ optimal error bound. The other ingredient is the subspace embedding property.

Subspace Embedding

One of the primary results developed in the randomized sampling algorithms literature was establishing the relation between the Johnson-Lindenstrauss Transform (JLT) and subspace embedding. The following lemma which is corollary 11 in [126] gives this important relation.

Lemma 6 *Let $0 < \epsilon, \delta < 1$ and f be some function. If $\Omega \in \mathbb{R}^{n\times\ell}$ satisfies a JLT-(ϵ, δ, k) with $\ell = O(k \log(k/\epsilon)/\epsilon^2.f(\delta))$, then for any orthonormal matrix $V \in \mathbb{R}^{n\times k}, n \geq k$ we have*

$$\mathbf{Pr}(\|V^\top \Omega \Omega^\top V - I\|_2 \leq \epsilon) \geq 1 - \delta.$$

The above lemma shows that, any sampling matrix Ω satisfying JLT and having length $\ell = O(k \log(k/\epsilon)/\epsilon^2)$ satisfies the subspace embedding property. Thus, any SCM Ω with a dual distance > 4 will also satisfy the subspace embedding property (since they satisfy JLT as we saw in the previous section). The subspace embedding property implies that the singular values of $V^\top\Omega$ are bounded, i.e., $V^\top\Omega$ is well conditioned with high probability. This result is critical since it shows that the SCM matrices can preserve the geometry of the top k-singular vectors of the input matrix A.

Observe that with the above analysis, we will require $\ell = O(k \log(k/\epsilon))$ number of samples for the subspace embedding property to be satisfied, which is similar to a subsampled Fourier or Hadamard matrix. Next, we show that for the subspace embedding property to be satisfied, we will require only $O(k/\epsilon)$ number of samples for certain types of code matrices.

We know that the code matrices display some of the properties of random matrices, particularly when the distance of the code is high. Indeed a code with dual distance above k supports k-wise independent probability measure and SCM Ω will be a random matrix with k-wise independent rows. This property of SCM helps us use the following lemma given in [119, Lemma 3.4] which states,

Lemma 7 *Given an integer k and $\epsilon, \delta > 0$. If $\Omega \in \mathbb{R}^{n\times\ell}$ is $\rho(k + \log(1/\delta))$-wise independent matrix with an absolute constant $\rho > 1$, then for any orthonormal matrix*

$V \in \mathbb{R}^{n\times k}$ and $\ell = O(k\log(1/\delta)/\epsilon)$, with probability at least $1-\delta$ we have

$$\|V^\top \Omega\Omega^\top V - I\|_2 \le \epsilon.$$

Thus, a sampling SCM matrix Ω which is $\lceil k+\log(1/\delta)\rceil$-wise independent satisfies the subspace embedding property with the number of samples (length) $\ell = O(k/\epsilon)$. Hence, an SCM Ω with dual distance $> \lceil k+\log(1/\delta)\rceil$ will preserve the geometry of V with $\ell = O(k/\epsilon)$.

In summary, any SCM with dual distance > 4 satisfies the JLT property, and will satisfy the subspace embedding property if $\ell = O(k\log(k/\epsilon))$. If the dual distance is $> k$, then the SCM can preserve the geometry of V with $\ell = O(k/\epsilon)$.

7.3.3 Deterministic Error bounds

In order to derive the bounds for the spectral norm error and the singular values obtained, we will use the deterministic error bounds for Algorithm 6 developed in the literature [67, 127]. Algorithm 6 constructs an orthonormal basis Q for the range of Y, and the goal is to quantify how well this basis captures the action of the input matrix A. Let $QQ^\top = P_Y$, where P_Y is the unique orthogonal projector with range$(P_Y) =$ range(Y). If Y is full rank, we can express the projector as : $P_Y = Y(Y^\top Y)^{-1}Y^\top$. We seek to find an upper bound for the approximation error given by, for $\xi \in \{2, F\}$

$$\|A - QQ^\top A\|_\xi = \|(I-P_Y)A\|_\xi.$$

The deterministic upper bound for the approximation error of Algorithm 6 is given in [67]. We restate theorem 9.1 in [67] below:

Theorem 6 (Deterministic error bound) *Let A be $m\times n$ matrix with singular value decomposition given by $A = U\Sigma V^\top$, and fixed $k \ge 0$. Choose a test matrix Ω and construct the sample matrix $Y = A\Omega$. Partition Σ as in (7.2), and define Ω_1 and Ω_2 via (7.3). Assuming that Ω_1 is full row rank, the approximation error satisfies for $\xi \in \{2, F\}$*

$$\|(I-P_Y)A\|_\xi^2 \le \|\Sigma_2\|_\xi^2 + \|\Sigma_2\Omega_2\Omega_1^\dagger\|_\xi^2. \tag{7.6}$$

An elaborate proof for the above theorem can be found in [67]. Using the submultiplicative property of the spectral and Frobenius norms, and the Eckart-Young theorem mentioned earlier, equation (7.6) can be simplified to

$$\|A - QQ^\top A\|_\xi \leq \|A - A_k\|_\xi \sqrt{1 + \|\Omega_2\|_2^2 \|\Omega_1^\dagger\|_2^2}. \tag{7.7}$$

Recently, Ming Gu [127] developed deterministic lower bounds for the singular values obtained from randomized algorithms, particularly for the power method [67]. Given below is the modified version of Theorem 4.3 in [127] for Algorithm 6.

Theorem 7 (Deterministic singular value bounds) *Let $A = U\Sigma V^\top$ be the SVD of A, for a fixed k, and let $V^\top \Omega$ be partitioned as in (7.3). Assuming that Ω_1 is full row rank, then Algorithm 6 must satisfy for $j = 1, \ldots, k$:*

$$\sigma_j \geq \sigma_j(\hat{A}_k) \geq \frac{\sigma_j}{\sqrt{1 + \|\Omega_2\|_2^2 \|\Omega_1^\dagger\|_2^2 \left(\frac{\sigma_{k+1}}{\sigma_j}\right)^2}} \tag{7.8}$$

where σ_j are the jth singular value of A and $\hat{A}_k$ is the rank-k approximation obtained by our algorithm.

The proof for the above theorem can be seen in [127]. In both the above theorems, the key assumption is that Ω_1 is full row rank. This is indeed true if the sampling matrix Ω satisfies the subspace embedding property.

7.3.4 Error Bounds

The following theorem gives the approximation error bounds when the subsampled code matrix (SCM) is used as the sampling matrix Ω. The upper and lower bounds for the singular values obtained by the algorithm are also given.

Theorem 8 (Error bounds for code matrix) *Let A be $m \times n$ matrix with singular values $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \ldots$. Generate a subsampled code matrix Ω from a desired coding scheme as in (7.1) with $r \geq \lceil \log_2(n) \rceil$ as the dimension of the code. For any code matrix Ω with* **dual distance** > 4 **and length** $\ell = O(k \log(k/\epsilon)/\epsilon^2 . f(\delta))$ *the following three bounds hold with probability at least $1 - \delta$:*

1. The Frobenius norm error satisfies,

$$\|A - \hat{A}_k\|_F \le \|A - A_k\|_F(1+\epsilon). \tag{7.9}$$

2. The spectral norm error satisfies,

$$\|A - \hat{A}_k\|_2 \le \|A - A_k\|_2\sqrt{1 + \frac{3n}{\ell}}. \tag{7.10}$$

3. The singular values obtained satisfy:

$$\sigma_j \ge \sigma_j(\hat{A}_k) \ge \frac{\sigma_j}{\sqrt{1 + \left(\frac{3n}{\ell}\right)\left(\frac{\sigma_{k+1}}{\sigma_j}\right)^2}}. \tag{7.11}$$

If the code matrix Ω has **dual distance** $\ge \lceil k + \log(1/\delta)\rceil$*, then the above three bounds hold for* **length** $\ell = O(k\log(1/\delta)/\epsilon)$*.*

Proof. [Proof - Frobenius norm Error] As we have been alluding to in the previous sections, the $(1+\epsilon)$ optimal Frobenius norm error given in eq. (7.9) is related to the JLT and the subspace embedding properties. The following lemma gives this relation which is Lemma 4.2 in Woodruff's monograph [28].

Lemma 8 *Let Ω satisfy the subspace embedding property for any fixed k-dimensional subspace M with probability $9/10$, so that $\|\Omega^\top y\|_2^2 = (1 \pm 1/3)\|y\|_2^2$ for all $y \in M$. Further, suppose Ω satisfies the $(\sqrt{\epsilon/k}, 9/10, k)$-JLT property such that the conclusion in Lemma 5 holds, i.e., for any matrices A, B each with n rows,*

$$\mathbf{Pr}\left[\|A^\top B - A^\top\Omega\Omega^\top B\|_F \le 3\sqrt{\epsilon/k}\|A\|_F\|B\|_F\right] \ge 9/10.$$

Then the column space of $A\Omega$ contains a $(1+\epsilon)$ rank-k approximation to A.

From the analysis in section 7.3.2 (in particular from Lemma 5 and 6), we know that both the conditions in the above lemma are true for SCM with dual distance > 4 and length $\ell = O(k\log(k/\epsilon)/\epsilon^2.f(\delta))$, when appropriate ϵ and δ are chosen. Since $\hat{A}_k = QQ^\top A$, where Q is the orthonormal matrix spanning the column space of $A\Omega$, we obtain the Frobenius error bound in eq. (7.9) from the above lemma.

Clarkson and Woodruff [119] gave the Frobenius norm error bound for low rank approximation using k-wise independent sampling matrices. The error bound in (7.9) for SCM with dual distance $> k$ is straight from the following lemma which is a modification of Theorem 4.2 in [119].

Lemma 9 *If $\Omega \in \mathbb{R}^{n\times\ell}$ is a $\rho(k+\log(1/\delta))$-wise independent sampling matrix, then for $\ell = O(k\log(1/\delta)/\epsilon)$, with probability at least $1-\delta$, we have*

$$\|A - \hat{A}_k\|_F \leq \|A - A_k\|_F(1+\epsilon). \tag{7.12}$$

Proof of this lemma is clear from the proof of Theorem 4.2 in [119]. □

Proof. [Proof - Spectral norm Error] The proof of the approximate error bounds given in (7.10) follows from the deterministic bounds given in sec. 7.3.3. We start from equation (7.7) in Theorem 6, the terms that depend on the choice of the test matrix Ω are $\|\Omega_2\|_2^2$ and $\|\Omega_1^\dagger\|_2^2$.

We know that the SCM Ω satisfies the subspace embedding property for the respective dual distances and lengths mentioned in the Theorem 8. This also ensures that the spectral norm of $\Omega_1^\dagger$ is under control. We have the condition $\|V_k^\top\Omega\Omega^\top V_k - I\|_2 \leq \epsilon_0$, implying

$$\sqrt{1-\epsilon_0} \leq \sigma_k(V_k^\top\Omega) \leq \sigma_1(V_k^\top\Omega) \leq \sqrt{1+\epsilon_0}.$$

Then from Lemma 3.6 in [121], we have

$$\|\Omega_1^\dagger\|_2^2 = \frac{1}{\sigma_k^2(\Omega_1)} \leq \frac{1}{(1-\epsilon_0)}.$$

In Lemma 8, we chose $\epsilon_0 = 1/3$ to prove the $(1+\epsilon)$ approximation. So, we have

$$\|\Omega_1^\dagger\|_2^2 \leq 3/2.$$

Next, we bound the spectral norm of Ω_2 as follows $\|\Omega_2\|_2^2 = \|V_2^\top\Omega\|_2^2 \leq \|V_2\|_2^2\|\Omega\|_2^2 = \|\Omega\|_2^2 = \sigma_1^2(\Omega)$, since V_2 is an orthonormal matrix. So, we need an upper bound on the top singular value of SCM Ω, which we derive from the following two lemmas. The first lemma shows that if a code has dual distance ≥ 3, the resulting code matrix Φ has orthonormal columns.

Lemma 10 (Code matrix with orthonormal columns) *A code matrix Φ generated by a coding scheme which results in codes that have dual distance ≥ 3, has orthonormal columns.*

Proof. If a code has dual distance 3, then the corresponding code matrix (stacked up codewords as rows) is an orthogonal array of strength 2 [43]. This means all the tuples of bits, i.e., $\{0,0\},\{0,1\},\{1,0\},\{1,1\}$, appear with equal frequencies in any two columns of the codeword matrix C. As a result, the Hamming distance between any two columns of C is exactly 2^{r-1} (half the length of the column). This means after the BPSK mapping, the inner product between any two codewords will be zero. It is easy to see that the columns are unit norm as well. $\square$

If there is no downsampling in Ω, then the singular values of Ω will simply be $\sqrt{n/\ell}$, due to the scaling in (7.1) of the orthonormal matrix and since $r = \log_2 n$. If we downsample the rows of Φ to form Ω, then the above fact helps us use Lemma 3.4 from [123] which shows that randomly sampling the rows of a matrix with orthonormal columns results in a well-conditioned matrix, and gives bounds for the singular values. The following lemma is a modification of Lemma 3.4 in [123].

Lemma 11 (Row sampling) *Let Φ be a $2^r \times \ell$ code matrix with orthonormal columns and let*

$$M = 2^r . \max_{j=1,\dots,2^r} \|e_j^\top \Phi\|_2^2.$$

For a positive parameter α, select the sample size

$$n \geq \alpha M \log(\ell).$$

Draw a random subset T from $\{1,\dots,2^r\}$ by sampling n coordinates without replacement. Then

$$\sqrt{\frac{(1-\nu)n}{2^r}} \leq \sigma_\ell(S_T\Phi) \text{ and } \sigma_1(S_T\Phi) \leq \sqrt{\frac{(1+\eta)n}{2^r}} \tag{7.13}$$

with failure probability at most

$$\ell . \left[\frac{e^{-\nu}}{(1-\nu)^{(1-\nu)}}\right]^{\alpha\log(\ell)} + \ell . \left[\frac{e^{\eta}}{(1+\eta)^{(1+\eta)}}\right]^{\alpha\log(\ell)},$$

where $\nu \in [0,1)$ *and* $\eta > 0$.

The bounds on the singular values of the above lemma are proved in [123] using the matrix Chernoff bounds.

Since n is fixed and $M = \ell$ for code matrices (all the entries of the matrix are $\pm 2^{-r/2}$), we get the condition $n \geq \alpha \ell \log(\ell)$. So, α is less than the ratio $n/\ell \log(\ell)$ and this ratio is typically more than 10 in the low rank approximation applications. For $\alpha = 10$, we choose $\nu = 0.6$ and $\eta = 1$, then the failure probability is at most $2\ell^{-1}$. Since we use the scaling $\sqrt{\frac{2r}{\ell}}$, the bounds on the singular values of the subsampled code matrix Ω will be

$$\sqrt{\frac{2n}{5\ell}} \leq \sigma_\ell(\Omega) \text{ and } \sigma_1(\Omega) \leq \sqrt{\frac{2n}{\ell}}. \tag{7.14}$$

Thus, we obtain $\|\Omega_2\|_2^2 \|\Omega_1^\dagger\|_2^2 = 3n/\ell$. We substitute this value in (7.7) to get the spectral norm error bounds in (7.10). □

Similarly, we obtain the bounds on the singular values given in (7.11) by substituting the above value of $\|\Omega_2\|_2^2 \|\Omega_1^\dagger\|_2^2$ in (7.8) of Theorem 7.

We observe that the upper bounds for the spectral norm error obtained in (7.10) for the SCM is similar to the bounds obtained for Gaussian random matrices and structured random matrices like SRFT/SRHT given in the review article by Halko et.al [67]. For the structured random matrices, $(1+\epsilon)$ optimal Frobenius norm error has been derived in [120]. We have a similar $(1+\epsilon)$ optimal Frobenius norm error obtained for subsampled code matrices with dual distance > 4 in (7.9). Importantly, we show that this optimal error bound can be achieved with number of samples $\ell = O(k/\epsilon)$ as opposed to $O(k \log k/\epsilon)$ required for structured random matrices when the dual distance of the code is $> k$. Details on how to generate such code matrices with dual distance $> k$ and length $\ell = O(k/\epsilon)$.

7.3.5 Least squares regression problem

In this section, we extend the framework to solve the least squares (ℓ_2) regression problem. As discussed in the introduction, the idea of randomized approximations is to reduce the dimensions of $A \in \mathbb{R}^{n \times d}$ and $b \in \mathbb{R}^n$ with $n \gg d$, by pre-multiplying them by a sampling matrix $\Omega \in \mathbb{R}^{n\times\ell}$, and then to solve the smaller problem quickly,

$$\min_x \|\Omega^\top Ax - \Omega^\top b\|_2. \tag{7.15}$$

Let the optimal solution be $x' = (\Omega^\top A)^\dagger\Omega^\top b$. Here we analyze the performance of SCM as the sampling matrix Ω. We require the sampling matrix Ω to satisfy the JLT and the subspace embedding properties, which are indeed satisfied by any SCM with dual distance > 4. Hence, we can use the results developed by Sarlos [126], and Clarkson and Woodruff [119] for our analysis. Similar to the earlier analysis, we can expect improved performance when SCM with dual distance $> k$ are used (k-wise independence property of codes). For this, we use the bounds derived by Clarkson and Woodruff [119] for random sign matrices. The following theorem which is a modification of Theorem 3.1 in [119] gives the upper bound for the regression problem in such cases.

Theorem 9 *Given $\epsilon, \delta > 0$, suppose $A \in \mathbb{R}^{n\times d}, b \in \mathbb{R}^n$ and A has rank at most k. If Ω is a $\rho(k+\log(1/\delta))$-wise independent with an absolute constant $\rho > 1$, and x' and $\hat{x}$ are solutions as defined before, then for $\ell = O(k\log(1/\delta)/\epsilon)$, with probability at least $1-\delta$, we have*

$$\|Ax' - b\|_2 \le (1+\epsilon)\|A\hat{x} - b\|_2.$$

This theorem shows that, if the code matrix is $\lceil k+\log(1/\delta)\rceil$-wise independent (i.e., dual distance $> \lceil k+\log(1/\delta)\rceil$), we can get $\epsilon-$approximate solution for the regression problem with $\ell = O(k\log(1/\delta)/\epsilon)$ samples. Thus, for the regression problem too, we have the $\log k$ factor gain in the number of samples over other structured random matrices (SRHT) given in [121, 120].

7.4 Choice of error correcting codes

7.4.1 Codes with dual-distance at least $k+1$

The requirement of k-wise independence of codewords translates to the dual distance of the code being greater than k. Since a smaller code (less number of codewords, i.e., smaller r) leads to less randomness in sampling, we would like to use the smallest code with dual distance greater than k.

One of the choices of the code can be the family of dual BCH codes. As mentioned earlier, this family has length ℓ, dimension $t\log(\ell+1)$ and dual distance at least $2t+1$. Hence, to guarantee dual distance at least k, the size of the code must be $2^{\frac{k\log(\ell+1)}{2}} = (\ell+1)^{k/2}$. We can choose n vectors of length $\frac{k\log(\ell+1)}{2}$ and form the codewords by simply multiplying these with the generator matrix (over $\mathbb{F}_2$) to form the subsampled code matrix. Therefore, forming these code matrices will be much faster than generating $n\times\ell$ i.i.d Gaussian random matrices or random sign matrices which have k-wise independent rows.

In general, from the Gilbert-Varshamov bound of coding theory [40], it is known that linear codes of size $\sim\sum_{i=0}^{k}\binom{\ell}{i}$ exist that have length ℓ and dual distance greater than k. The construction of these code families are still randomized. However, when $k=O(\ell)$, or the dual distance is linearly growing with the code length, the above construction of dual BCH code does not hold in general. Infinite families of codes that have distance proportional to the length are called *asymptotically good codes*. The Gilbert-Varshamov bound implies that asymptotically good linear codes of size $\sim 2^{\ell h(\frac{k}{\ell})}$ exist[1], that have length ℓ and dual distance greater than k.

7.4.2 Choice of the code matrices

Depending on the types of input matrices and the computational environments, we can choose different types of code matrices that best suit the applications. If the input matrix is a general dense matrix which can be stored in the fast memory (RAM), we can choose any structured code matrix with dual distance >4, $r=\lceil\log_2 n\rceil$ (or choose message vectors to be 1 to $2^{\lceil\log_2 n\rceil}$) and $\ell=O(k\log k)$ (eg., dual BCH codes), so that the fast multiplication technique can be exploited (the log factor will not be an issue). This will be similar to using any other structured random matrices like SRFT or SRHT. In fact, Hadamard matrices are also a class of linear codes, with variants known as Hadamard codes, Simplex codes or 1st-order Reed-Muller codes. The dual distance of Hadamard code is 3. However, with code matrices (say dual BCH codes), subsampling of columns is not required, thus reducing randomness and cost.

If the input matrix is sparse and/or is distributively stored, and for parallel implementation, we can choose a code matrix with dual distance $>k$ and generate them as

[1] $h(x)\equiv -x\log_2 x-(1-x)\log_2(1-x)$ is the binary entropy function

Table 7.1: Classes of sampling matrices with subspace embedding properties

Matrix Classes	ℓ	Runtime	Randomness
Gaussian (or random sign)	$O(k/\epsilon^2)$	$O(mn\ell)$	$n\ell$
SRFT/SRHT [123, 120]	$O(k\log(kn)\log(k/\epsilon^2)/\epsilon^2)$	$O(mn\log\ell)$	$\Theta(n)$
Count Sketch [28]	$(k^2+k)/\epsilon^2$	$O(\mathrm{nnz}(A))$	n
Code matrix (dual distance≥ 4)	$O(k\log(k/\epsilon)/\epsilon^2)$	$O(mn\log\ell)$	n
Code matrix (dual distance$\geq k$)	$O(k/\epsilon^2)$	$O(mn\log\ell)$	$\Theta(n)$

mentioned earlier. These code matrices are not structured and we can treat them as dense transforms (any random matrices). For SRFT/SRHT sampling matrices, we need to communicate $O(k\log k)$ columns, but for code matrices with dual distance $> k$, the log factor is not necessary. This will help us overcome the issues with SRFT/SRHT for sparse input matrices and in parallel and distributed applications. These code matrices are easy to generate (than i.i.d Gaussian random matrices), the log factor in the number of samples is not necessary, and thus, using code matrices in these applications will reduce randomness and cost significantly. When using code matrices, we also have computations gains in the cost of generating the sampling matrices, since the code matrices are deterministic, and also require lower number of random numbers to be generated.

Summary of the classes of sampling matrices We summarize in Table 7.1 a list of some of the classes of optimal sampling matrices which satisfy the subspace embedding property. The table lists the sampling complexity ℓ required for achieving the $(1+\epsilon)$ optimal bounds and the runtime cost for computing the matrix product $Y = A\Omega$. The table also lists the amount of random numbers (randomness) required for each of sampling matrices. A comprehensive list with systematic description of these and more classes of sampling matrices, expect the last two classes can be found in [122].

7.5 Numerical Experiments

Now, we illustrate the performance of subsampled code matrices as sampling matrices. We compare the performance of dual BCH code matrices against the performance of random Gaussian matrices and subsampled Fourier transform (SRFT) matrices for different input matrices from various applications.

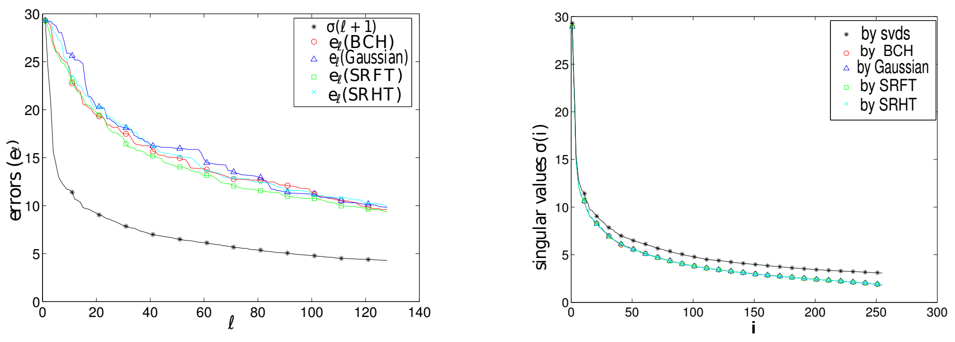


Figure 7.1: (Left) Randomized SVD using dual BCH code, Gaussian, SRFT and SRHT matrices as sampling matrix for input matrix `Kohonen`.

Our first experiment is with a 4770×4770 matrix named Kohonen from the Pajek network (a directed graph's matrix representation), available from the SuiteSparse Matrix Collection [66]. Such graph Laplacian matrices are commonly encountered in machine learning and image processing applications. The performance of the dual BCH code matrix, Gaussian matrix, subsampled Fourier transform (SRFT) and Hadamard (SRHT) matrices are compared as sampling matrices Ω. For SRHT, we have to subsample the rows as well (similar to code matrices) since the input size is not a power of 2. All experiments were implemented in matlab v8.1, on an Intel I-5 3.6GHz processor.

Figure 7.1(Left) gives the actual error $e_\ell = \|A - Q^{(\ell)}(Q^{(\ell)})^\top A\|$ for each ℓ number of samples when a subsampled dual BCH code matrix, a Gaussian matrix, SRFT and SRHT matrices are used as sampling matrices, respectively. The best rank-ℓ approximation error $\sigma_{\ell+1}$ is also given. Figure 7.1(Right) plots the singular values obtained, for $\ell = 255$ and different sampling matrices Ω used. The top 255 exact singular values of the matrix are also plotted. We observe that, in practice, the performance of all four sampling matrices are similar.

Table 7.2 compares the errors e_ℓ for ℓ number of samples, obtained for a variety of input matrices from different applications when subsampled dual BCH code, Gaussian and SRFT matrices were used. All matrices were obtained from SuiteSparse database [66]. Matrices lpi_ceria3d and deter3 are from linear programming problems. S80PI_n1 and dw4096 are from an eigenvalue/model reduction problem. Delaunay, EPA, ukerbe1, FA (network) and Kohonen are graph Laplacian matrices. qpband is from an optimization problem. The table depicts two sets of experiments (divided by the line). The first set

Table 7.2: Comparison of errors

			Dual BCH	Gaussian	SRFT
Matrices	Sizes	ℓ	Error	Error	Error
lpiceria3d	4400×3576	63	16.61	17.55	17.32
Delaunay	4096×4096	63	6.386	6.398	6.383
deter3	21777×7647	127	9.260	9.266	9.298
EPA	4772×4772	255	5.552	5.587	5.409
Kohonen	4770×4770	511	4.297	4.294	4.261
ukerbe1	5981×5981	127	3.093	3.0945	3.092
dw4096	8192×8192	127	108.96	108.93	108.98
FA	10617×10617	127	2.19	2.17	2.16
qpband	20000×20000	63	4.29	4.30	4.26

(top five examples) illustrates how errors vary as the sample size ℓ is increased. The second set (bottom four) illustrates how the errors vary as the size of the matrix increases. For the last matrix (qpband) we could compute the decomposition for only $\ell = 63$ due to memory restrictions. We observe that, for small ℓ, in the first five examples the error performance of code matrices is slightly better than that of Gaussian matrices. For higher ℓ, the error remains similar to the error for Gaussian matrices. All input matrices are sparse, hence we cannot use the fast transforms. We still see that code matrices take less time than both Gaussian and SRFT. In practice, we can use code matrices in place of fully random (Gaussian) matrices or structured random matrices due to the advantages of code matrices over the other sampling matrices, as discussed in the previous sections.

Chapter 8

Group testing with codes

8.1 Introduction

The group testing problem involves efficient identification of a small number k of defective elements in population of a large size n [128]. The idea is to test the elements in groups with the premise that most tests will return negative results, clearing the entire group. If the test result is positive, then the group contains at least one defective. The collection of tests is said to form a *group testing scheme* if the outcomes of the tests enable us to identify any small subset of defectives of size say k.

A nonadaptive group testing scheme with m tests is typically described by an $m \times n$ binary incidence matrix A, where each row corresponds to a test, and $A_{ij} = 1$ if and only if the ith test includes the jth element. The result of the test is positive if the indices of ones in the row overlap with the indices of the defective configuration. The smallest possible number of tests is known to satisfy $m = \Theta(\frac{k^2}{\log k} \log n)$ [128].

A construction of group testing schemes using matrices of error-correcting codes and code concatenation appeared in the foundational paper by Kautz and Singleton [129]. Many later constructions of group testing schemes also rely on codes and code concatenations [130, 131]. Other explicit constructions of non-adaptive group testing schemes with $m = O(k^2 \log n)$ are also suggested, see [128]. In order to improve the tradeoff between the parameters of the scheme, construction of schemes that permit a small probability of error (false positives) is suggested. Such schemes were considered under the name of *weakly separated designs* in [132].With this relaxation, it is possible to reduce the number of tests to $\Theta(k \log n)$ [133]. An explicit (non-probabilistic) construction of almost disjunct matrices with the number of tests proportional to $k^{3/2}\sqrt{\log n}$ was presented in [134] and was subsequently improved to $k \log^2 n/\log k$ in [133]. The construction of [129] and many others above are based on constant weight error-correcting codes. Estimates of the parameters of the group testing schemes from constant weight codes were obtained using the minimum distance of the code [129] and more recently using the average distance [134, 133].

In this chapter, we perform experiments with test matrices constructed from the codewords of a fixed (low) weight in binary BCH codes. The resulting matrices are sparse in the sense that most of their entries are zero. We show that these matrices perform extremely well in experiments, outperforming random matrices. We also give some theoretical justification for this construction. Sharpening the estimates of the error probability is currently an open problem.

We derive an estimate of the probability of false positive that is based on the dual distance d' of constant weight codes. Constant weight codes with a given d' are known as combinatorial designs (of strength $d'-1$). A design of strength t (an t-design, or, in more detail, an t-(n, w, λ) design) is a collection of w-subsets of an n-set V, called blocks, such that every r elements of V are contained in the same number λ of blocks. The use of t-designs for constructing disjunct matrices is not new, see, [128, §7.4].

8.2 Definitions and Notation

Definition 3 *An $m \times n$ binary matrix A is called k-*disjunct* if the support*[1] *of any of its columns is not contained in the union of the supports of any other k columns.*

A k-disjunct matrix gives a group testing scheme that identifies any defective set up to size k. Conversely, any group testing scheme that identifies any defective set up to size k must be a $(k-1)$-disjunct matrix [128]. Disjunct matrices support a simple identification algorithm that runs in time $O(nk)$. Note that, any element that participates in a test with a negative outcome is not defective. After we perform all the tests and weed out all the non-defective elements from negative tests, the disjunctness

[1]The support of a vector $x \in \mathbb{F}_q^n$ is the set $\mathrm{supp}(x) := \{i : x_i \neq 0\}$.

property of the matrix guarantees that all the remaining elements are defective.

Definition 4 *For any $\epsilon > 0$, an $m \times n$ matrix A with columns $a_1, a_2, \dots, a_N$, is called* k-disjunct(ϵ) *if* $\Pr(\{I \in \binom{[n]}{k}, j \in [n] \setminus I \mid \operatorname{supp}(a_j) \subseteq \bigcup_{t \in I} \operatorname{supp}(a_t)\}) \le \epsilon$.

The union of supports of a randomly and uniformly chosen subset of k columns of a k-disjunct(ϵ) matrix does not contain the support of any other random column with probability at least $1-\epsilon$. The next fact follows from the definition of disjunct matrices and the decoding procedure [128, p. 134].

Proposition 2 *A k-disjunct(ϵ) matrix defines a group testing scheme that can identify all items in a random defective configuration of size k and with probability ϵ identifies any randomly chosen item outside of the defective configuration as defective (false-positive).*

A *code* of length m is a subset of the vector space $\mathbb{F}_q^m$. The minimum Hamming distance between distinct codewords of $\mathcal{C}$ is called the *distance of the code.* We use the notation $\mathcal{C}(m, n, d)$ to refer to the code of length m, cardinality n and distance d. If in addition all the codevectors of the code $\mathcal{C}$ contain exactly w nonzero entries, we call it a *constant weight code* and use the notation $\mathcal{C}(m, n, d, w)$. Finally we refer to a linear code of length m and dimension r as an $[m, r]$ code.

8.3 Numerical Experiments

This work is motivated in part by the experimental results which show that matrices formed of codewords of a fixed weight obtained from binary codes perform very well in the group testing problem for identifying defective entries. Here, we present a few simulation results using matrices formed by codewords of fixed weight obtained from different BCH codes, and compare their performance with best possible randomly generated sparse matrices.

The results are presented in Fig. 8.1. Each of the eight plots in Fig. 8.1 presents results of identification of defectives for two group testing schemes, one using a matrix formed of the fixed-weight codewords of the BCH code and the other using a random binary matrix. The experiments were organized as follows. For instance, for the first

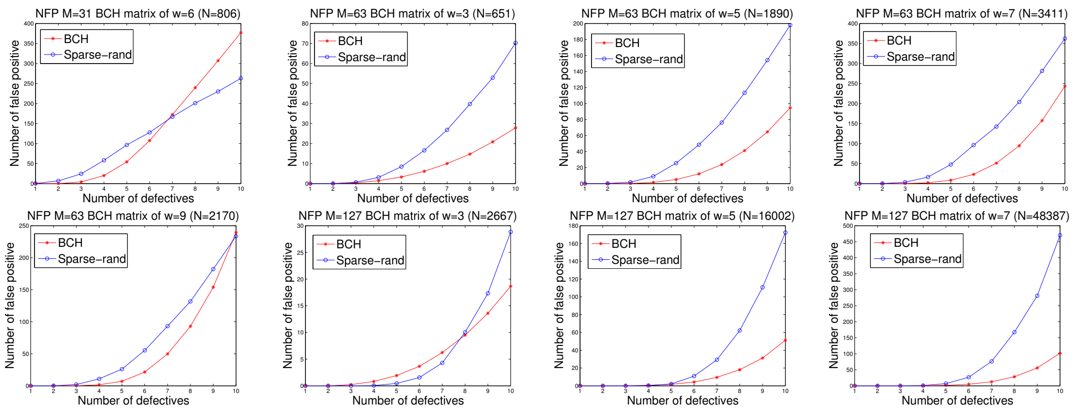


Figure 8.1: Number of false positives with fixed-weight BCH codeword matrices and sparse random matrices, averaged over 300 trials.

plot (top left corner) we formed a test matrix A by using the codewords of weight $w = 6$ in the $[m = 31, r = 21]$ BCH code as its columns. There are $n = 806$ such codewords in the code, which enables us to construct a 31×806 test matrix. This means that we can test a set of $n = 806$ items for the presence of defectives using $m = 31$ tests. To compare this construction with the scheme based on random matrices, we constructed a sparse random matrix assigning the entries independently to 0 or 1 with probability of 1 equal to $\tilde{p} = 1/(k+1)$ and 0 with probability $1 - \tilde{p}$. It is known (and also experimentally verified) that $\tilde{p} = 1/(k+1)$ gives the best performance among such random matrices [128]. Each experiment consisted of generating a random vector with k defectives randomly inserted among n items and performing the identification procedure. This experiment is repeated 300 times; then we compute the average number of false positives found by the two group testing schemes.

Similar experiments were performed for the other group testing schemes shown in Fig. 8.1. In the second plot, we used a $[63, 57]$-BCH codeword matrix with constant weight $w = 3$, $n = 651$ and $m = 63$, i.e., the matrix formed by the codewords of weight 3 of the Hamming code of length 63 (they are known to support a 2-design). Other examples are similar (the values of (m, n, w) are as reported in the plots).

We note that the fixed-weight BCH codeword matrices consistently in most cases perform much better than sparse random matrices in terms of the number of false

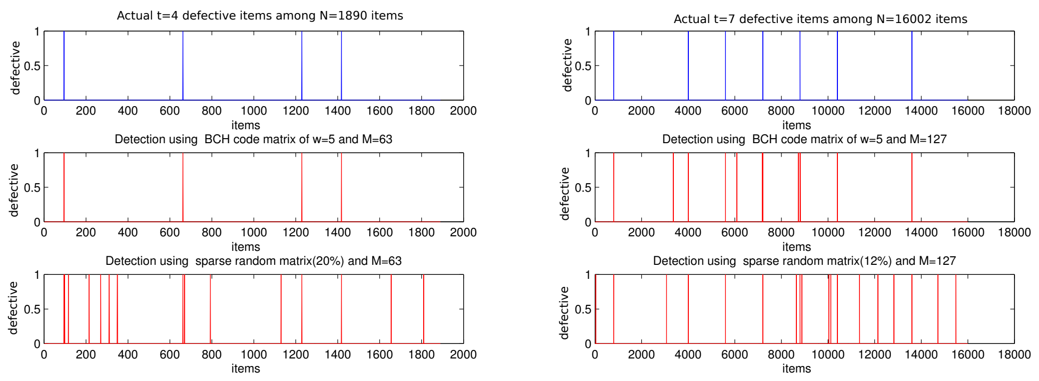


Figure 8.2: Two group testing examples.

positives detected, and the gap widens (in most cases) with the increase of the number of defectives.

To exemplify this improvement, we show in Fig. 8.2 two individual experiments performed for fixed-weight BCH testing matrices used to generate Fig. 8.1, namely, the matrices with the parameters $n = 1890, m = 63, w = 5$ and $n = 16002, m = 127, w = 5$. In the left column in Fig. 8.2 we show the locations of the actual defective elements inserted in the population and the defective vectors identified by the BCH-based matrix (middle plot) and the random matrix (bottom plot) for the first set of parameters. We see that the BCH matrix locates the defectives exactly while the random matrix inserts a large number of false positives compared to the actual number $k = 4$. In the right column of the plot we show similar results for an individual experiment for the second set of parameters. Here the BCH matrix-based scheme inserts a few false defectives, while the random matrix adds many more.

8.4 Estimates of error probability

In this section we cite some known, and present some new results on constant weight codes with which we attempt to explain the observed performance of constant weight almost disjunct matrices. The following well-known result of [129] has been the basis of a large number of construction of testing matrices.

Proposition 3 *An (m, n, d, w) constant weight binary code $\mathcal{C}$ provides a k-disjunct matrix, where $k = \lfloor \frac{w-1}{w-d/2} \rfloor$.*

This proposition implies that a group testing scheme can be obtained from constant weight codes with large distance. However, the set of codewords of weight 5 in a $[63, 51]$ BCH code forms a $(63, 1890, 3, 5)$ constant weight code with distance $d = 5$, so Proposition 3 clearly fails to explain the performance of the group testing scheme obtained from this code.

Extending the theory of Kautz-Singleton to almost-disjunct matrices, [133] provides a bound on the false-positive probability of a constant weight code matrix in terms of its distance distribution. Define the *average distance* D of a code $\mathcal{C}$:

$$D(\mathcal{C}) = \frac{1}{|\mathcal{C}|} \min_{x \in \mathcal{C}} \sum_{y \in \mathcal{C}} d_{\mathrm{H}}(x, y).$$

Here d_{H} denotes the Hamming distance. Define also the second-moment of the distance distribution as follows:

$$D_2(\mathcal{C}) = \frac{1}{|\mathcal{C}|^2} \sum_{\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}} d_{\mathrm{H}}(\boldsymbol{x}, \boldsymbol{y})^2.$$

One of the main results of [133] is the following theorem.

Theorem 10 *Let $\mathcal{C}$ be a constant weight binary code $\mathcal{C}$ of size n, minimum distance d and average distance D such that every codeword has length m and weight w. The test matrix obtained from the code is k-disjunct(ϵ) for the largest k such that the inequality*

$$d \geq D - \frac{3(w - k(w - D/2))^2}{(\ln 1/\epsilon)(2k(w - D/2) + w)}$$

holds true.

Paper [133] also provides a more refined estimate that relies on the second moment of the distance distribution.

Theorem 11 *Let $\mathcal{C}$ be a constant-weight (m, N, d, w) binary code with average distance D and the second moment of the distance distribution D_2. The test matrix obtained*

Table 8.1: Minimum distance and average distance

Parameters	$d = d_{\min}$	$D = d_{avg}$	D_2
$m = 31, w = 6$	6	9.6894	96.7742
$m = 63, w = 3$	4	5.7231	33.1797
$m = 63, w = 5$	6	9.2112	86.1239
$m = 63, w = 7$	8	12.4481	157.3620
$m = 63, w = 9$	10	15.4357	241.8802
$m = 127, w = 3$	4	5.8605	34.5917
$m = 127, w = 5$	6	9.6050	93.0134

Table 8.2: Estimates of the error probability ϵ from Theorem 10

Parameters	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 6$	$k = 7$	$k = 8$	$k = 9$	$k = 10$
$m = 31, w = 6$	0.101	0.354	0.671	0.906	0.998	0.963	0.848	0.699	0.550	0.416
$m = 63, w = 3$	0.013	0.026	0.048	0.080	0.122	0.175	0.238	0.308	0.384	0.463
$m = 63, w = 5$	0.032	0.080	0.157	0.261	0.384	0.514	0.642	0.756	0.851	0.923
$m = 63, w = 7$	0.047	0.138	0.284	0.463	0.644	0.800	0.914	0.980	1.000	0.981
$m = 63, w = 9$	0.058	0.197	0.414	0.649	0.842	0.961	1.000	0.972	0.896	0.794
$m = 127, w = 3$	0.012	0.017	0.025	0.035	0.046	0.061	0.078	0.097	0.119	0.143
$m = 127, w = 5$	0.028	0.047	0.073	0.106	0.146	0.193	0.246	0.303	0.364	0.427

Table 8.3: Estimates of the error probability ϵ from Theorem 11

Parameters	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 6$	$k = 7$	$k = 8$	$k = 9$	$k = 10$
$m = 31, w = 6$	0.043	0.164	0.431	0.781	0.993	0.896	0.578	0.271	0.093	0.023
$m = 63, w = 3$	0.012	0.026	0.048	0.079	0.120	0.172	0.233	0.302	0.377	0.454
$m = 63, w = 5$	0.022	0.048	0.091	0.158	0.249	0.362	0.493	0.628	0.758	0.867
$m = 63, w = 7$	0.024	0.060	0.130	0.246	0.410	0.607	0.802	0.945	1.000	0.951
$m = 63, w = 9$	0.024	0.072	0.181	0.376	0.641	0.891	1.000	0.895	0.632	0.347
$m = 127, w = 3$	0.012	0.018	0.025	0.035	0.046	0.060	0.077	0.096	0.118	0.142
$m = 127, w = 5$	0.023	0.033	0.048	0.066	0.089	0.116	0.150	0.189	0.234	0.285

from the code is k-disjunct(ϵ) for the largest k such that the inequality

$$d \geq D + \frac{3t(D_2 - D^2)}{2(w - t(w - D/2))} - \frac{3(w - t(w - D/2))}{\ln 1/\epsilon} \tag{8.1}$$

holds true.

In order to compute the estimates based on these theorems, we computed the distance distributions of a number of constant weight codes obtained from low-weight codewords of the BCH codes of length 63 and 127 mentioned above, and found D and D_2 for these codes. The results are listed in Table 8.1. Using these values, we can find the estimates of the error probability ϵ given by Theorems 10 and 11. The results are summarized in Tables 8.2 and 8.3, respectively. Although they represent a large improvement over the initial estimates of Kautz-Singleton, they still do not match the performance in actual experiments. For example, with codewords of weight 5 in a BCH code of length 127, even with 3 defectives the predicted false positive probability is 0.0479, whereas the number of false positives in the experiments is close to zero.

Next we will show that better estimates can be achieved in many cases. To formulate the result we need to define the *dual distance* of a constant weight code. The distance distribution of a constant weight code $\mathcal{C}(m,n,d,w)$ is a set of numbers $b_0, b_1, \ldots, b_w$, where

$$b_i = \frac{1}{|\mathcal{C}|}|\{(x,y) \in \mathcal{C}^2 : w - |\operatorname{supp}(x) \cap \operatorname{supp}(y)| = i\}| \tag{8.2}$$

for $i = 0, 1, \ldots, w$. Note that $b_0 = 1$. The *dual distance* d' of $\mathcal{C}$ is defined as

$$d'(\mathcal{C}) = \min\Big\{j \geq 1 : b'_j := \frac{1}{|\mathcal{C}|}\sum_{i=0}^{w} b_i Q_j(i) > 0\Big\},$$

where $Q_j(i)$ is the value of the Hahn polynomial of degree j; see [128].

Now we are ready to state the main theorem in our analysis.

Theorem 12 *Let $\mathcal{C}$ be an (m,n,d,w) constant weight code with dual distance d' and let $w < m/2$. Let k be the maximum number of defective items and suppose that $k < m/w$. For any even $\ell < d'$ the probability of a false positive test result for the group testing scheme constructed from $\mathcal{C}$ is bounded above as*

$$\epsilon \leq B(\ell,k)\Big(\frac{e\ell(m-w)}{2(m-kw)^2}\Big)^{\ell/2}\sum_{i=0}^{\ell/2}\Big(\frac{(m-w)\ell}{2ew^2}\Big)^i, \tag{8.3}$$

where $B(\ell,k) = \min\{(18\ell k)^{\ell/2}, k^\ell\}$. In addition, if $m \geq \max\{4w^2k/\ell^2, w + 2ew^2/\ell\}$, then

$$\epsilon \leq k\Big(\frac{2\ell^2(m-w)}{(\log\ell)w(m-kw)}\Big)^{\ell}. \tag{8.4}$$

In the case of $\ell = 2$ we have

$$\epsilon < \frac{k}{m-1}\frac{(m-w)^2}{(m-wk)^2}. \tag{8.5}$$

Table 8.4: Estimates of the error probability ϵ from (8.5), Thm. 12

Parameters	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 6$	$k = 7$	$k = 8$	$k = 9$	$k = 10$
$m = 63, w = 3$	0.016	0.036	0.060	0.089	0.126	0.172	0.230	0.305	0.403	0.533
$m = 63, w = 5$	0.016	0.039	0.070	0.117	0.188	0.299	0.484	–	–	–
$m = 63, w = 7$	0.016	0.042	0.086	0.165	0.322	0.688	–	–	–	–
$m = 127, w = 3$	0.008	0.017	0.026	0.037	0.049	0.062	0.076	0.092	0.110	0.130
$m = 127, w = 5$	0.008	0.017	0.028	0.041	0.057	0.075	0.098	0.125	0.158	0.199
$m = 127, w = 7$	0.008	0.018	0.030	0.046	0.067	0.095	0.131	0.181	0.251	0.352

Proof: (outline) The ideas of the proof are as follows. First we show that as long as $t < d'$, the tth central moment of the distance distribution of the code equals the tth moment of the hypergeometric random variable with the pmf $f_X(i) = \binom{w}{i}\binom{m-w}{w-i}/\binom{m}{w}, i = 0, 1, \ldots, w$. This fact relies on simple properties of the Johnson association scheme. After that we use a condition similar to $w > k(w - d/2)$ (see the proof of Prop. 3) as sufficient for identification and write a Chernov-type bound that it is violated, where the random variables are the indices of k random columns of the matrix A. This gives the estimate of ϵ in the form of the tth moment of a sum of certain independent random variables which can be bounded above using classical inequalities such the Rosenthal inequalities or other similar estimates. The full development of these arguments is rather long; see [135].

Using this information together with Theorem 12, we can compute upper bounds for the false-positive probabilities obtained from Eq. (8.5) for the various fixed-weight BCH codeword matrix examples given in Fig. 8.1. The results are listed in Table 8.4[2]. We can see that the bounds are small, especially for smaller k, even when we have $\ell = 2$. Compared to all previous results, this gives better estimates of ϵ.

[2] '-' in the table means that the computed estimate is trivial.

Chapter 9

Multilabel classification with group testing and codes

9.1 Introduction

In the multilabel classification problem, we are given a set of labeled training data $\{(x_i, y_i)\}_{i=1}^n$, where $x_i \in \mathbb{R}^p$ are the input features for each data instances and $y_i \in \{0,1\}^d$ are vectors indicating the corresponding labels (classes) the data instances belong to. The vector y_i has a one in the jth coordinate if the ith data point belongs to jth class. We wish to learn a mapping (prediction rule) between the features and the labels, such that, we can predict the class label vector y of a new data point x correctly. Such multilabel classification problems occur in many domains such as text mining, computer vision, music, and bioinformatics, and modern applications involve large number of labels. Popular applications with many labels include image and video annotation [136], web page categorization [137], text and document categorization [138], and others [139]. In most of these applications, the label vectors y_i are sparse (with average sparsity of $k \ll d$), i.e., each data point belongs to a few (average k out of d) classes. The multiclass classification is an instance of the multilabel classification, where all data points belong to only one of the d classes ($k = 1$).

The simple binary classification problem, where $d = 2$ and $k = 1$ is well-studied, and several efficient algorithms have been proposed in the literature. A natural approach used to solve the multiclass ($d > 2, k = 1$) classification problem is to reduce the problem into a set of binary classification problem, and then employ the efficient binary classifiers to solve the individual problems. Popular methods based on this approach are: one-vs-all, all-pairs, and the error-correcting output code (ECOC) [140] methods. In ECOC method, m-dimensional binary vectors (typically codewords from an error correcting code with $m \leq d$) are assigned to each class, and m binary classifiers are learned. For the jth classification, the jth coordinate of the corresponding codeword is used as the binary label for each class. In the modern applications, where d is typically very large, this approach is found to be very efficient due to the reduction of the class dimension.

Related Work: The idea of ECOC approach has been extended to the multilabel classification (MLC) problem. In the multiclass classification, using codewords for each class in ECOC is equivalent to multiplying the code matrix to the label vectors (since the label vectors are basis vectors). In the multilabel setting, the d dimensional label vectors are reduced to m dimensional by multiplying a random matrix $A \in \mathbb{R}^{m \times d}$ to the label vector y. This reduction method was analyzed from the compressed sensing point of view in [141], with the assumption of output sparsity, i.e., y is sparse (with average sparsity k). Using compressed sensing (CS) theory, the results in [141] show that for a linear hypothesis class and under the squared loss, a random embedding (random code matrix) of the classes to $m = O(k \log d)$ dimensions does not increase the $L2$ risk of the classifier. However, the CS approach requires solving an optimization problem to recover the label vector. Constructions with faster recovery algorithms exist but we cannot obtain $L2$ norm results with them.

Alternatively, embedding based approaches have been proposed to reduce the effective number of labels. These methods reduce the label dimension by projecting label vectors onto a low dimensional space, based on the assumption that the label matrix $Y = [y_1, \ldots, y_n]$ is low-rank. The various embedding methods proposed in the literature mainly differ in the way this reduction is achieved. The reduction is achieved using SVD in [107], while column subset selection is used in [108]. These embedding methods capture the label correlation, and Euclidean distance error guarantees are established. However, the low rank assumption breaks down in many situations [139], e.g., data is power law distributed [142].

The state of the art embedding method called SLEEC (Sparse Local Embedding for

Extreme Classification) [139] overcomes the limitations of previous embedding methods by first clustering the data into smaller regions, and then performs local embeddings of label vectors by preserving distances to nearest label vectors. However, this method also has many shortcomings, see [142]. Moreover, most of these embedding based methods are very expensive. They involve eigenvalue or singular value decompositions and matrix inversions, and may require solving convex optimization problems, all of which become impractical for very large d. In all the embedding methods and the CS method, the reduced label space is a real space (no longer binary). Hence we need to use regressors for training and cannot leverage the efficient binary classifiers for effective training for the model. Prediction will also involve rounding/thresholding of real values. This is additional work, and choosing a right threshold is sometimes problematic.

Contributions: In this chapter, we present a novel reduction approach to solve the MLC problem. Our approach assumes output sparsity (sparse label vectors with $k \ll d$) similar to the CS approach, but reduces a large binary label vector to a *binary* vector of smaller size. Since the reduced label vectors are binary, we can use the efficient binary classifiers for effective training for the model. Our prediction algorithm is extremely simple and does not involve any matrix inversion or solving optimization algorithm. The prediction algorithm can also detect and correct errors.

We make the crucial observation that, the MLC problem can be solved using the group testing (GT) premise. That is, the problem of estimating the (few) classes of a data instance from a large set of classes, is similar to identifying a small set of items from a large set. We consider a group testing binary matrix A and reduce the label vectors y_i's to smaller binary vectors z_i using the boolean OR operation $z_i = A \vee y_i$ (described later). We can now use binary classifiers on z_i for training. The m classifiers learn to test whether the data belongs to a group (of labels) or not. During prediction, the label vector can be recovered from the predictions of the classifiers using a simple inexpensive algorithm (requiring no matrix inversion or solving optimization algorithms). A low prediction cost is extremely desirable in real time applications.

The idea of grouping the labels helps overcome the issues most existing methods encounter; e.g., when the data has power law distribution [142], that is many labels have very few training instances (which is the case in most popular datasets), and tail labels. Since the classifiers in our approach learn to test for groups of labels, we will have more training instances per group yielding effective classifiers. It is well known that the one-vs-rest is a highly effective method (expensive), and recently a (doubly) parallelized version of this method called DiSMEC [142] was shown to be very effective. Our approach is similar to one-vs-rest, but the classifiers test for a group of labels, and we require very few classifiers ($O(\log d)$ instead of d).

We establish Hamming loss error bounds for the proposed approach. Due to the error correcting capabilities of the algorithm, even if a fraction of classifiers mis-classify, we can achieve zero prediction error. Numerical experiments with various datasets illustrate the superior performance of our group testing approach with different GT matrices. Our method is extremely inexpensive compared to the CS approach and especially compared to the embedding based methods, making it very desirable for real time applications too.

9.2 MLC via Group testing

Preliminaries: Recall the group testing problem and the testing schemes we discussed in the previous chapter. We know that a k-disjunct matrix gives a group testing scheme that identifies any defective set up to size k exactly. Next, we have the following important definition.

Definition 5 (Error Correction) *An $m \times d$ binary matrix A is called* (k, e)-disjunct*, $e \geq 1$, (k-disjunct and e-error detecting) if for every set S of columns of A with $|S| \leq k$, and $i \notin S$, we have $|\operatorname{supp}(A^{(i)}) \backslash \cup_{j \in S} \operatorname{supp}(A^{(j)})| > e$, where $A^{(i)}$ denote the ith column of A.*

A (k, e)-disjunct matrix can detect up to e errors in the measurements and can correct up to $\lfloor e/2 \rfloor$ errors. This property is stronger than k-disjunct and k-disjunct(ϵ) properties (not to be confused with the latter).

Several random and deterministic constructions of k-disjunct matrices have been proposed in the literature [129, 128], some of which we discussed in the previous chapter. Matrices from error correcting codes and expander graphs have also been designed, which we will discuss later.

Algorithm 7 MLGT Training

```
**Input:** Training data $\{(x_i, y_i)\}_{i=1}^n$, group testing matrix $A \in \mathbb{R}^{m\times d}$, a binary classifier algorithm $\mathcal{C}$.
**Output:** $m$ classifiers $\{w_j\}_{j=1}^m$.
for $i = 1, \dots, n$. do
    $z_i = A \vee y_i$.
end for
for $j = 1, \dots, m$. do
    $w_j = \mathcal{C}(\{(x_i, z_{ij})\}_{i=1}^n)$.
end for
```

Algorithm 8 MLGT Prediction

```
**Input:** Test data $x \in \mathbb{R}^p$, the group testing matrix $A \in \mathbb{R}^{m\times d}$ which is $(k, e)$-disjunct, $m$ classifiers $\{w_j\}_{j=1}^m$.
**Output:** predicted label $\hat{y}$.
Compute $\hat{z} = [w_1(x), \dots, w_m(x)]$.
Set $\hat{y} \leftarrow 0$.
for $l = 1, \dots, d$ do
    if $|A^{(l)} \backslash \hat{z}| < e/2$ then
        $\hat{y}_l = 1$.
    end if
end for
```

Now, we present our main idea of adapting the group testing scheme to the multilabel classification problem (MLGT).

Training. Suppose we are given n training instances $\{(x_i, y_i)\}_{i=1}^n$, where $x_i \in \mathbb{R}^p$ are the input features for each instances and $y_i \in \{0,1\}^d$ are corresponding label vectors. We begin by assuming that each data instance belongs to at most k classes (the label vector y is k sparse). We consider a (k, e)-disjunct matrix $A \in \mathbb{R}^{m\times d}$. We then compute the reduced measured (label) vectors z_i for each label vectors $y_i, i = 1, \dots, n$ using the boolean OR operation $z_i = A \vee y_i$. We can now train m binary classifiers $\{w_j\}_{j=1}^m$ based on $\{x_i, z_i\}_{i=1}^n$ with jth entry of z_i indicating which class (1/0) the ith instance belongs to for the jth classifier. Algorithm 7 summarizes our training algorithm.

Prediction. In the prediction stage, given a test data $x \in \mathbb{R}^p$, we use the m classifiers $\{w_j\}_{j=1}^m$ to obtain a predicted reduced label vector $\hat{z}$. We know that a k sparse label vector can be recovered exactly, if the group testing matrix A is a k-disjunct matrix. With a (k, e)-disjunct matrix, $e \geq 1$, we can recover the k sparse label vector exactly, even if $\lfloor e/2 \rfloor$ binary classifiers mis-classify, using the following decoder.

Decoder : Given a predicted reduced label vector $\hat{z}$, and a group testing matrix A, set the coordinate position of $\hat{y}$ corresponding to $l \in [1, \ldots, d]$ to 1 if and only if $|\operatorname{supp}(A^{(l)}) \setminus \operatorname{supp}(\hat{z})| < e/2$.

That is, we set the lth coordinate of $\hat{y}$ to 1, if the number of coordinates that are in the support of the corresponding column $A^{(l)}$ but are not in the predicted reduced vector $\hat{z}$, is less than $e/2$. The decoder returns the exact label vector even if up to $e/2$ binary classifiers make errors. Algorithm 8 summarizes our prediction algorithm.

Note that the prediction algorithm is very inexpensive (requires no matrix inversion or solving optimization). It is equivalent to an AND operation between a binary sparse matrix and a binary (likely sparse) vector, which should cost less than a sparse matrix vector product $O(nnz(A)) \approx o(kd)$, where $nnz(A)$ is the number of nonzero entries of A. It is an interesting future work to design an even faster prediction algorithm.

9.3 Constructions

In order to recover a k sparse label vector exactly, we know that the group testing matrix A must be a k-disjunct matrix. With a (k, e)-disjunct matrix, our algorithm can extract the sparse label vector exactly even if $e/2$ binary classifiers make errors (mis-classify). Here, we present the results that will help us construct specific GT matrices with the desired properties.

9.3.1 Random Constructions

Proposition 4 (Random Construction) *An $m \times d$ random binary $\{0, 1\}$ matrix A where each entry is 1 with probability $\rho = \frac{1}{k+1}$, is $(k, 3k \log d)$-disjunct with very high probability, if $m = O(k^2 \log d)$.*

If we tolerate a small ε fraction of sparsity label misclassifications (i.e., εk errors in the recovered label vector), which we call ε-tolerance group testing (recall the k-disjunct(ε) property), then we can follow the analysis of Theorem 8.1.1 in [128], to show that it is sufficient to have $m = O(k \log d)$ number of classifiers. Further, we can derive the following result.

Theorem 13 *Suppose we wish to recover a k sparse binary vector $y \in \mathbb{R}^d$. A random binary $\{0,1\}$ matrix A where each entry is 1 with probability $\rho = 1/k$ recovers $1-\varepsilon$ proportion of the support of y correctly with high probability, for any $\varepsilon > 0$, with $m = O(k \log d)$. This matrix will also detect $e = \Omega(m)$ errors.*

The proofs of the above proposition and theorem can be found in [10].

9.3.2 Concatenated code based constructions

Kautz and Singleton [129] introduced a two-level construction in which a q-ary ($q > 2$) Reed-Solomon (RS) code is concatenated with a unit-weight binary code. The construction starts with a q-ary ($q > 2$) RS code of length $q-1$, and replaces the q-ary symbols in the codewords by unit weight binary vectors of length q. That is, the q-ary symbols are replaced as $0 \to 100\dots0; 1 \to 010\dots0; q-1 \to 0\dots01$. This gives us a binary matrix with $m = q(q-1)$ rows. This matrix belongs to a broad class of error correcting codes called the *constant weight codes* (each codeword/column has a constant number of ones w). For this Kautz-Singleton construction, $w = q-1$.

Proposition 5 (Kautz-Singleton construction) *A Kautz-Singleton construction with $(k \log_k d)$-ary Reed-Solomon (RS) code is a $(k, (k-1)\log_k d)$-disjunct matrix with $m = \Theta(k^2 \log_k^2 d)$.*

Proof. A constant weight code matrix is k disjunct matrix with $k = \lfloor \frac{w-1}{w-h/2} \rfloor$, where w is the weight and h is the distance of the code, (see, Theorem 7.3.3 in [128]). The distance of the q-ary RS code is $h = 2(q - \log_q(d))$. Hence, we get $k = \frac{q-2}{\log_q d - 1}$. So, for a k-disjunct matrix, we choose $q = k \log_k d$. A code with distance h will have $e = h/2$ (by using Corollary 8.3.2 in [128]). Thus, $e = q - \log_q d \approx (k-1)\log_k d$. $m = q(q-1) = \Theta(k^2 \log_k^2 d)$. □

Many code based constructions have been proposed with the optimal length of $m = \Theta(k^2 \log_k d)$ [143]. One such code based construction of interest is the Algebraic-Geometric codes. Considering $q = r^2$, where r is an integer, we can generate a family of Algebraic-Geometric (AG) codes of length m_q, satisfying $m_q \geq r^{a+1} - r^a + 1$, where a is an even integer. Using the Kautz-Singleton mechanism, we can convert this AG code to a binary code that has constant weight $w = m_q$. The length of the binary code will be $m = qm_q$.

Proposition 6 *We can construct an Algebraic-Geometric code matrix that recovers* $1-\varepsilon$ *proportion of nonzeros in* y *with high probability, for* $\varepsilon > 0$*, with* $m \geq 16k \log_{2k} d \log(d/\varepsilon)$*. This matrix will also detect* $e = \left(8\log(d/\varepsilon) - \frac{8\log(d/\varepsilon)}{\sqrt{2k}-1} - 1\right)\log_{2k} d$ *errors.*

Proof. The proof follows from the results developed in [143]. For a q-ary Algebraic-Geometric Code with $q \geq 2k$, that is converted to a binary code using Kautz-Singleton mechanism, we have the $1-\varepsilon$ recovery guarantees for $m \geq \frac{16k\log d}{\log 2k}\log(d/\varepsilon)$. We know if the code has a distance h, then $e = h/2$. The q-ary AG code satisfies

$$h \geq 2m/q - 2\log_q d - \frac{2m}{q(\sqrt{q}-1)}.$$

We get the value for e upon substitution. □

9.3.3 Expander graphs

Expander graphs have popularly been used in many applications. In an expander graph, every small set of vertices "expands": the are "sparse" yet very "well-connected" (see formal definition below). With high probability a random graph is a good expander. Construction of "lossless" expanders have been notoriously difficult.

Definition 6 (Unbalanced Lossless Expander Graphs) *A* (k,ϵ)*-unbalanced bipartite expander graph is a bipartite graph* $G(L,R,E), |L| = d, |R| = m$*, where* L *is the set of left nodes and* R *is the set of right nodes, with regular left degree* ℓ *such that for any* $S \subset L$*, if* $|S| \leq k$ *then the set of neighbors* $N(S)$ *of* S *has the size* $N(S) > \epsilon\ell|S|$*.*

The following proposition describe the expander property of random graphs.

Proposition 7 *A random construction of bipartite graphs* $G(L,R,E)$ *with* $|L| = d$ *with overwhelming probability, is* (k,ϵ)*-lossless* ℓ*-regular expander where* $\ell = O(\log d/\epsilon)$ *with* $|R| = m = O(k\ell/\epsilon)$*.*

The trade-off of this proposition is close to the best we can hope for. The proof can be shown by simple random choice and can be found in [144]. The next definition and the subsequent two claims are from [144]. First, let us now connect a lossless expander with disjunct matrix.

Definition 7 *A bipartite graph $G(L, R, E)$ is called (k, e)-disjunct if, for every left vertex $i \in L$ and every set $S \subseteq L$ such that $|S| \leq k$ and $i \notin S$, we have $|N(i) \backslash N(S)| > e$.*

It can be seen that the bipartite adjacency matrix A of a disjunct graph G is a disjunct matrix.

Proposition 8 *Let G be a (k, e)-disjunct graph with adjacency matrix A. Then for every pair of $y, y' \in \{0, 1\}^d$ of k -sparse vectors, we have $\Delta(A \vee y, A \vee y') > e$, where $\Delta(\cdot)$ denotes the Hamming distance between vectors.*

The following proposition relates expander graphs with disjunct graphs.

Proposition 9 *Let G be a ℓ-regular (k, ϵ)-lossless expander. Then, for every $\alpha \in [0, 1)$, G is $(k - 1, \alpha\ell)$-disjunct provided that $\epsilon < \frac{1-\alpha}{\ell}$.*

Combining these comments, we get the following:

Proposition 10 (Random Graphs) *The adjacency matrix of a randomly constructed bipartite graph is, with overwhelming probability, k-disjunct with $m = O(k^2 \log(d/k))$. More generally, for every $\alpha \in [0, 1)$, random graphs are (k, e)-disjunct, with $e = \Omega(\alpha k \log d/(1 - \alpha^2))$ with $m = \Omega(\alpha k^2 \log(d/k)/(1 - \alpha^2))$.*

There is an explicit construction of unbalanced (k, ϵ)-lossless expanders for any setting of d and m presented in [145]. These constructions yield explicit k-disjunct graphs with $m = O(k^2 \text{quasipoly}(\log d))$.

With all the above constructions, we can correct a reasonably large number of e errors by the binary classifiers. The number of classifiers required for MLGT will be $m = O(k^2 \log d)$ which is more than the CS approach where $m = O(k \log d)$. However, our analysis is for the worst case: as we saw in Theorem 13, if we tolerate a small ε fraction of error in recovery, we can achieve $m = O(k \log d)$ for MLGT as well. Moreover, MLGT yields *zero* prediction error for a k sparse label vector even if up to $e/2$ classifiers mis-classify. With MLCS, we only get an ϵ error guarantees and with respect to 2-norm.

9.4 Error Analysis

Here we summarize the theoretical error guarantees for multilabel classification using group testing (MLGT).

Theorem 14 *Consider MLGT with an $m \times d$ binary matrix A, and a label vector y with sparsity at most k. Suppose A is (k, e)-disjunct, and we use Algorithm 8 during prediction. Let $\hat{y}$ be the predicted label vector and $\Delta(\cdot)$ denote the Hamming distance between vectors. If t number of binary classifiers that make errors in prediction, then we have*

- *If $t \leq \lfloor e/2 \rfloor$, then the prediction error $\Delta(y, \hat{y}) = 0$.*
- *If $t > \lfloor e/2 \rfloor$, $\Delta(y, \hat{y}) \leq w(t - e/2)$ (Hamming error), where w is the maximum weight of rows in A. In particular, the error rate (average error per class) will be $\frac{w}{d}(t - e/2)$.*

If A is a k-disjunct with ε error tolerance, then the prediction error will be at most $(w(t - e/2) + \varepsilon k)$.

Proof. When, $t \leq e/2$, we know that the decoding algorithm will still recover the exact label vector due to the error correcting property of the (k, e)-disjunct matrix. When, $t > e/2$, $e/2$ of the errors are corrected. For every remaining $t - e/2$ errors, if w is the maximum weight of rows in A, a maximum of w errors can occur in the predicted label. This is because, the support different $|A^{(l)} \backslash \hat{z}|$ can change for a maximum of w columns. Hence, the error can be at most $w(t - e/2)$, and the error rate will be $\frac{w}{d}(t - e/2)$. For the k-disjunct matrix with ε error tolerance, the decoding algorithm can make up to εk errors in addition to $w(t - e/2)$. □

Let us see how the error-rate of various group testing constructions translate to MLGT. In the case of a random matrix construction, we have $w \approx d/k$. So, the error rate for this matrix will be $(t - e/2)/k$. From proposition 4, we can take $m = k^2 \log d$, and $e = 3k \log d$. Hence, the error rate for a random (k, e) disjunct matrix will be $t/k - 3/2 \log d$, for any $t > 3/2k \log d$. For any t less than this the error rate will be zero. Similarly, we can see that the randomized construction of Thm. 13 with $m = O(k \log d)$ rows, gives the average error rate is $(t/k - O(\log d) + \varepsilon k/d)$ for $t > k \log d$.

Corollary 5 (Constructions) *For MLGT, we have the following results for different constructions:*

- *If A is constructed via randomized construction of Prop. 4 with $m = O(k^2 \log d)$ rows, then the average error rate is $t/k - \frac{3}{2}\log d$ for $t > 3/2 \log d$.*
- *If A is constructed via randomized construction of Thm. 13 with $m = O(k \log d)$ rows, then the average error rate is $(t/k - O(\log d) + \varepsilon k/d)$ for $t > k \log d$.*
- *If A is constructed deterministically via Kautz-Singleton Reed-Solomon codes construction of Prop. 5 with $m = O(k^2 \log_k d)$ rows, then the average error rate is $\frac{t}{k \log_k d} - O(1)$ for $t > k \log_k d$.*
- *If A is constructed via expander graph-based construction of Prop. 10 with $m = O(k^2 \log(d/k))$ rows, then the average error rate is $t/k - \log(d/k)$ for $t > k/2 \log(d/k)$.*

The error rate is zero for smaller number of mis-classifications t.

Proof. In the case of a random matrix construction, we have $w \approx d/k$. So, the error rate for this matrix will be $(t - e/2)/k$. From proposition 1, we can take $m = k^2 \log d$, and $e = 3k \log d$. Hence, the error rate for a random (k, e) disjunct matrix will be $t/k - 3/2 \log d$. The properties of the rest of the matrices can be proved in similar way. □

9.5 Numerical Experiments

In this section, we illustrate the performance of the proposed group testing approach in the multilabel classification problems (MLGT) via numerical experiments on various datasets.

Datasets: We use some popular publicly available multilabel datasets in our experiments. All datasets were obtained from The Extreme Classification Repository[1] [139]. Details about the datasets and the references for their original sources can be found in the repository. Table 9.1 gives the statistics of these datasets. In the table, d = #labels, $\bar{k}$ =average sparsity per instance, n = #instances and p = #features.

[1] `https://manikvarma.github.io/downloads/XC/XMLRepository.html`

Table 9.1: Dataset statistics

Dataset	d	k	n	p
Mediamill	101	4.38	30993	120
Bibtex	159	2.40	4880	1839
Delicious	983	19.03	12920	500
RCV1-2K	2456	4.79	623847	47236
EurLex-4K	3993	5.31	15539	5000
AmazonCat-13K	13330	5.04	1186239	203882
Wiki10-31K	30938	18.64	14146	101938

Constructions: For MLGT, we consider three different group testing constructions. The Kautz-Singleton construction with q-ary Reed-Solomon (RS) codes, where we use RS codes [40] with $q = 16$ and $m = 240$; and $q = 8$ and $m = 56$. To get desired number of codewords (equal to number of labels), we use appropriate message length. For example, if $d \leq 4096, q = 16$, then we use message length of 3, and if $d \leq 65536$, we use message length of 5. We also use two random GT constructions, namely, the random expander graphs and the sparse random constructions discussed in sec. 9.3. For MLCS (compressed sensing approach), we again consider three different types compression matrices, namely, random Gaussian matrices, compressed Hadamard matrices and random expander graphs.

Evaluation metrics: Two evaluation metrics are used to analyze the performances of the different methods. First is the Hamming loss error, the Hamming distance between the predicted vector $\hat{y}$ and the actual label vector y, $\Delta(y, \hat{y})$. This metric tells us how close is the recovered vector $\hat{y}$ is from the exact label vector y, and is more suitable for binary vectors. Hamming loss captures the information of both correct predictions and false labels. All prediction errors reported (training and test) are Hamming loss errors. The second metric used is $Precison@k$ (P@k), which is a popular metric used in MLC literature [137]. This measures the precision of predicting the first k coordinates $|\operatorname{supp}(\hat{y}_{1:k}) \cap \operatorname{supp}(y)|/k$. Since we cannot score the labels, we use $k = nnz(y)$ the output sparsity of the true label for this measure. This is equivalent to checking whether the method predicted all the labels the data belongs to correctly or not (ignoring misclassification). When $Precision@k$ for $k = 1, 3, 5$ are used, one is checking whether the top 1, 3 or 5 labels are predicted correctly (ignoring other and false labels).

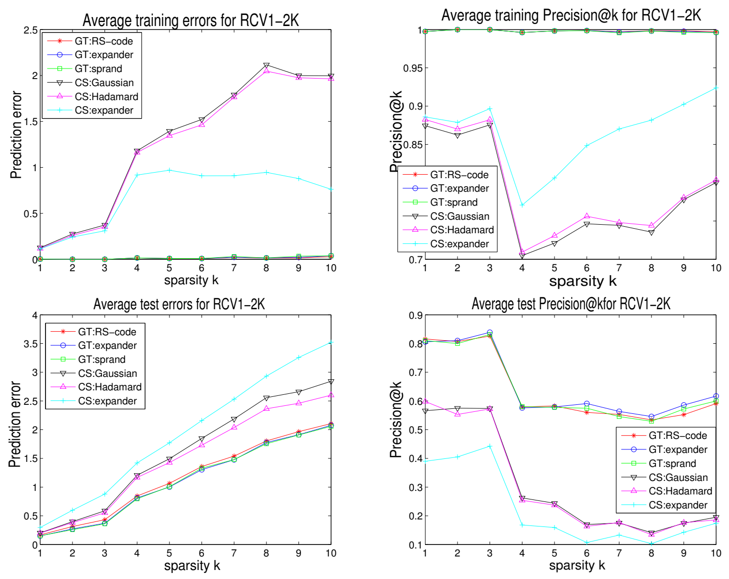


Figure 9.1: Average training and test errors and $Precison@k$ versus sparsity k for RCV1-2K for different MLGT and MLCS methods.

MLGT vs MLCS: In the first set of experiments, we compare the performances of the group testing approach (MLGT) and the compressed sensing approach (MLCS) using different group testing constructions and different compression matrices. A least squares binary classifier was used $\{w_j\}_{j=1}^m$ for MLGT. Least squares regression with ℓ_2 regularization (ridge regression) is used as the regressors for MLCS and other embedding based methods. Orthogonal Matching Pursuit (OMP) [116] was used for sparse recovery in MLCS.

Figure 9.1 plots the average training and test errors and average $Precison@k$ against the sparsity k of the label vectors (data with label sparsity k used) obtained for MLGT and MLCS methods with the three different matrices respectively. The dataset used was RCV1-2K. This dataset has at least 2000 training points and 500 testing points for each label sparsity ranging from 1 to 10. We observe that the training error for MLGT methods are almost zero and training $Precison@k$ almost one. Results with test data for MLGT are also impressive, achieving $Precison@k$ of almost 0.8 for small k.

We observe that, the MLGT method with all the three GT constructions outperforms the MLCS method. This is because, the binary classifiers are optimally trained on the reduced binary vectors and since the matrices used were k-disjunct, we had zero

Table 9.2: MLGT v/s OvsA

		MLGT			OvsA		
Dataset	d	Err	P@k	time	Err	P@k	time
Medmill	101	**0.68**	**0.24**	5.9s	2.07	0.17	13.1s
Bibtex	159	**1.17**	0.11	14.1s	2.55	**0.16**	36.6s

Table 9.3: Comparisons with embedding methods. Average Test errors.

		MLGT		MLCS		ML-CSSP		PLST		SLEEC	
Dataset	m	Err	time	Err	time	Err	time	Err	time	Err	time
Mediamill	40	**2.27**	2.35s	4.10	4.04s	4.44	13.4s	10.10	**2.13s**	4.37	4.55s
Bibtex	50	**1.57**	**7.81s**	2.93	14.7s	5.63	23.9s	7.39	12.1s	4.85	38.3s
Delicious	150	4.99	18.1s	9.43	29.9s	5.66	47.7s	15.66	**17.3s**	**4.79**	18.7s
RCV2K	200	**1.14**	**154s**	4.37	387s	24.60	339s	20.98	290s	4.94	210s
EurLex	200	**3.03**	**160s**	8.55	367s	9.30	337s	15.40	297s	4.84	455s
AmznC	250	**4.49**	**12mn**	11.10	22mn	13.66	19mn	7.50	18mn	6.93	1.2hr
Wiki10	300	**5.58**	**283s**	14.22	18mn	8.30	17mn	15.15	17mn	6.62	54mn

recovery error in most cases. Hence, the predicted labels for training data were extremely accurate. The results on test data are also better for MLGT in almost all cases. We also observed that MLGT is significantly faster than MLCS (as expected) because, MLCS uses an optimization algorithm (OMP) for recovery of labels, see Table 9.3 for runtimes.

In the prediction algorithm of MLGT, we have a parameter e, the number of errors the algorithm should try to correct. The ideal value for e will depend on the GT matrix used, the values of m, k and d. However, note that we can test for different values of e at no additional cost. That is, once we compute the Boolean AND between the predicted reduced vector and the GT matrix (the dominant operation), we can get different prediction vectors for a range of e and choose an e that gives the highest training P@k.

One vs all: We next compare MLGT against the one versus all (OvsA) method on two small datasets. Note that OvsA required d classifiers to be trained, hence is impractical for larger datasets, and we will need a distributed implementation such as DiSMEC [142]. Table 9.2 gives the results for MLGT and OvsA methods for two small datasets. $n = 5000$, $nt = 1000$, and for MLGT $m = 50$. The table lists the Hamming test errors and *Precision@k* (P@k) for the two methods. The table also gives the overall runtimes for the two methods. We note that wrt. to both metrics, MLGT performs better than OvsA. This is due to two reasons. First, MLGT groups the labels hence has more training samples per group, yielding better classifiers. Second, the error correction by the prediction algorithm corrects few classification errors. Clearly, MLGT is faster than OvsA. However, OvsA gives better training errors.

Embedding methods: In the next set of experiments, we compare the performance of MLGT against the popular embedding based methods. We compare with the following methods. ML-CSSP, is an embedding method based on column subset selection [108]. PLST, is Principal Label Space Transformation [107], an embedding method based on SVD. SLEEC, Sparse Local Embeddings for Extreme Classification [139], is the state of the art embedding method based on clustering using nearest neighbors and then embedding in the cluster space. For MLGT, we use the random expander graph constructions. For MLCS, we use random Gaussian matrices. Same least squares regressor was used in all the latter four methods.

Table 9.3 lists the test (Hamming) errors obtained for the different methods on various datasets. We use smaller datasets since the embedding based methods do not scale well for large datasets. We also used only 2000 training points and 500 test points in each cases. We observe that MLGT outperforms the other methods in most cases. The datasets have very sparse label (avg. sparsity of around $\bar{k} \approx 4$), but the outputs of MLCSSP and PLST are not very sparse. Hence, we see high Hamming error for these two methods, since they yield a lot of false labels. Moreover, these embedding methods are significantly more expensive than MLGT for larger datasets. The runtimes for each method are also listed in the table.

The runtimes reported (using cputime in Matlab) includes construction of compression matrices, multiplying the matrix to the label vectors (boolean OR/SVD computation), training the m classifiers, and prediction of n training and nt test points. SLEEC performs reasonably well on all datasets (the ideal parameters to be set in this algorithm for each of these datasets were provided by the authors online), and gives better P@k than MLGT for some datasets. For Delicious dataset, the value of k is high and SLEEC beats MLGT. However, SLEEC algorithm has many parameters to set, and for larger datasets, the algorithm is very expensive compared to all other methods.

Part III

Machine Learning for Scientific Data Applications

Chapter 10

Union of Intersections

10.1 Introduction

In many scientific fields, the development of new sensing and imaging technologies has resulted in generation of large volumes of data. These large datasets bring with them opportunities of new discoveries and insights into the fundamentals of nature. Realizing this potential requires novel machine learning and statistical data analysis algorithms that are both interpretable and predictive. Statistical-machine learning algorithms for scientific data should satisfy the bi-criteria of returning results that are simultaneously predictive and interpretable. By predictive, we mean that it can predict (e.g., reconstruct) the data with high accuracy; by interpretable, we mean that the results give insight into the (bio)-physical processes that generated the data. Interpretability usually entails the sparse selection and accurate estimation of a small number of physically meaningful features of the data. However, these bi-criteria are often at odds, and methods that robustly (few assumptions on the data/noise) achieve both are lacking. Such methods could provide insights into natural phenomena through the extraction of physically or biologically interpretable models.

In this chapter, we present *Union of Intersections* (UoI), a flexible, modular, and scalable framework for enhanced model selection and estimation. UoI satisfies the bi-criteria of low-variance, nearly unbiased estimation of a small number of interpretable features while maintaining high-quality prediction accuracy. We first introduce the framework via. the LASSO regression problem. We then, present a novel two stage algorithm $UoI\text{-}NMF_{cluster}$ for NMF inspired by the UoI framework $UoI\text{-}NMF_{cluster}$ yields: a) more accurate parts-based decompositions of noisy data, b) a sparse and accurate weight matrix, and c) high-accuracy reconstructions of the de-noised data. Together, these improvements enhance the performance and interpretability of NMF application to noisy data, and suggest similar approaches may benefit other matrix decomposition algorithms. Hence, we adapt the UoI framework for CUR decomposition to develop the two stage $UoI\text{-}CUR$ algorithm.

Modern technologies such as Electrocorticography (ECoG) and near-infrared spectroscopy (NIRS) have resulted in the collection of large volumes of neurophysiological data. Extracting key features from such data will provide better insight into functioning of the brain and may result in new discoveries. The novel machine learning algorithms we develop in this chapter based on UoI are applied to different Neuroscience data to exact interpretable results.

10.2 UoI for regression

Union of Intersections (UoI) is a flexible, modular, and scalable framework for statistical-machine learning problems proposed in [11]. The core concept of the UoI framework is to separate feature selection from feature estimation, and use bootstrap resampling to determine stable features and estimate the parameter values for those features to maximize predictive accuracy. In UoI-based methods, model selection is first performed through intersection (compressive) operations which induce sparsity, followed by model estimation through union (expansive) operations which reduces the variance of estimates.

For example, consider the regression problem with ℓ_1 regularization: Given the data $(Y_1, X_1), \ldots, (Y_n, X_n)$, with univariate response Y and p-dimensional predictor variable X, we wish to minimize

$$L(\beta, \lambda) = \|Y - X\beta\|_2^2 + \lambda\|\beta\|_1.$$

In the UoI_{Lasso} algorithm, we (1) calculate model supports (S_j) (location of nonzero entries of β) using an intersection operation across different bootstrap resamples of the data for a range of regularization parameters (λ: increases in λ shrink all values of β

towards 0), constructing a family of model supports [$\mathbf{S} : S_j \subset S_{j-k}$ for $\Delta\lambda = \lambda_j - \lambda_{j-k}$ sufficiently large]; (2) combine the pure model selection (obtained from the intersection operation) with model estimation using a union operation to obtain better selection, estimation and prediction accuracy. Further details on the UoI_{Lasso} algorithm can be found in [11]. The main contribution of this thesis is adapting this UoI framework to develop new algorithms for NMF and CUR decomposition. We also apply the new methods to neurophysiological data to extract interpretable results from the Neuroscience viewpoint.

10.3 UoI for nononegative matrix factorization

With the ever growing collection of large volumes of scientific data, development of interpretable machine learning tools to analyze such data is becoming more important. Dimensionality reduction and low rank approximations/decompositions are popular tools used in many applications to analyze high dimensional data. However, methods such as principal component analysis (PCA) often yield uninterpretable results, as the eigenvectors can be additive combinations of up to all the data features. Alternate matrix decomposition methods such as Nonnegative Matrix Factorization (NMF) and CUR decomposition, which we saw in the first chapter, have been shown to perform well in some scientific applications [146, 147].

Here we present a novel, noise-robust NMF algorithm ($UoI\text{-}NMF_{cluster}$) that gives more accurate parts based decompositions and sparser weight matrices with improved reconstruction of denoised data. $UoI\text{-}NMF_{cluster}$ is inspired by the Union of Intersections (UoI) framework [11], and incorporates three innovations: (i) completely separate bases (H) learning from weight (W) estimation, (ii) learn bases (H) by clustering NMF results across bootstrap resamples of the data, and (iii) use UoI to estimate ultra-sparse weights (W) to maximize data reconstruction accuracy.

$UoI\text{-}NMF_{cluster}$ is a two stage algorithm which computes sets of bases over bootstrap resamples of the data using a standard NMF algorithm, and clusters the bases to learn the best stable and uncorrelated set of k bases. The algorithm then directly uses UoI applied to the non-negative least squares problem to compute a sparse weight matrix that best reconstructs the original input data given the selected bases. Using this two stage process, our method ensembles different models (bases), selects the stable bases using clustering, and achieves sparse, low-variance solutions (weights W in our case) without imposing a prior, see [11] for the discussion.

The goal of $UoI\text{-}NMF_{cluster}$ is not to solve a single optimization problem to obtain a single NMF, but to extract stable bases and learn sparse weights that map these bases to the data with high accuracy. Our algorithm has some resemblance to popular *ensemble* methods [148], which improve prediction accuracy by combining different models (parameter estimates). However, ensemble methods often include more features (expand the feature space) to predict the response variables, which make them hard to interpret [11]. $UoI\text{-}NMF_{cluster}$ generates a bootstrapped ensemble of potential bases, and uses clustering to extract uncorrelated bases that are stable to the bootstrap procedure. It then selects few bases (i.e., sparse model selection) and uses bagging to estimate their contributions (W) without imposing an explicit prior on the weight distribution, thus giving low-bias and low-variance estimates. Through this approach, we find that $UoI\text{-}NMF_{cluster}$ learns interpretable and predictive structure from complex, noisy data.

10.3.1 UoI-NMFcluster

Here, we present the $UoI\text{-}NMF_{cluster}$ algorithm. First, lets look at the notation used.

Notation: The input matrix A is assumed to be of size $m \times n$ with m data points in $\mathbb{R}^n$, is decomposed into a basis matrix $H \in \mathbb{R}^{k \times n}$ with k rows and a weight matrix $W \in \mathbb{R}^{m \times k}$. We denote the output of $UoI\text{-}NMF_{cluster}$ by $\hat{H}, \hat{W}$ of best rank $\hat{k}$. The different sets of NMFs obtained for different bootstraps are denoted by the pairs $\{W_i, H_i\}_{i=1}^{B_1}$, where B_1 is the number of bootstrap samples used. The set of integers $1, \ldots, n$ is denoted by $[n]$. A matrix that contains the indices of the nonzero entries of W is denoted by W_{idx}. For a given rank k_1, the matrix where the basis matrices $\{H_i\}_{i=1}^{B_1}$ are stacked up is denoted by $\tilde{H}^{(k_1)}$.

The $UoI\text{-}NMF_{cluster}$ algorithm has the following two stages:

Bases Learning: We compute the matrices H_i and W_i for different bootstrap samples of the data $i = 1, \ldots, B_1$, and for different ranks k using a standard NMF algorithm. The multiplicative update algorithm [149] for the KL divergence error metric gave us the best results (any other NMF algorithm can also be used). The next step is to learn the best basis matrix $\hat{H}$ from all the sets of bases $\{H_i\}_{i=1}^{B_1}$ learned over different bootstraps. The objective is to learn a set of bases that are stable parts based decomposition of the data.

We make the observation that bases which are stable, i.e., similar bases (near duplicates) that appear from different bootstrap samples are individual parts of the data, and are close to each other spatially (are dense points in the spatial distributions). Also, different parts of the data are dissimilar and must be apart from each other spatially. Intuitively, one part should be different than other parts. That is, the dense clusters formed by similar bases must be well separated. The noisy or spurious bases learned will be different for each bootstrap samples and these are typically spread out spatially. Hence, in order to learn a stable parts based decomposition of the data, i.e., extract the stable (similar) bases from the set of bases learned over different bootstraps, and ignore the noisy and spurious bases, we employ the popular robust density based clustering algorithm called DBSCAN (Density Based Spatial Clustering of Applications with Noise) [150].

We cluster the $k \cdot B_1$ bases learned over different bootstraps using the DBSCAN algorithm. The DBSCAN algorithm has two parameters, namely, the threshold *Eps* and the least minimum number of points per cluster ($MinPts$). We choose $MinPts \approx B_1/2$ because the stable bases should be learned for at-least half of the bootstrap samples. The threshold *Eps* can be chosen using the strategy proposed in [150]. The algorithm naturally clusters spatially dense points into individual clusters, hence, similar (stable) parts based bases which are spatially dense, are grouped into different clusters. The algorithm leaves out all noisy points (points not within the *Eps*-neighborhood of a cluster) without assigning them to a group. Therefore, the clusters we obtain for DBSCAN have only similar (stable) parts based bases. We choose the centers of these clusters as the best (stable) bases $\hat{H}$.

Bases Selection and Weight Estimation: Once the best bases $\hat{H}$ is learned, we next update/recompute the weight matrices $\{W_i\}_{i=1}^{B_1}$ based on $\hat{H}$. We use the same UoI strategy as in UoI_{Lasso} (UoI for ℓ_1 regression described above) for intersecting the supports (intersect the location of nonzeros) of each rows of the weights based on new

Algorithm 9 *$UoI\text{-}NMF_{cluster}$*

```
**Input:** Data $A \in \mathbb{R}_+^{m \times n}$, rank $k$, and number of bootstrap resamples, $B_1, B_2$.
**Output:** $\hat{W} \in \mathbb{R}_+^{m \times k}$ and $\hat{H} \in \mathbb{R}_+^{k \times n}$.
**1. Bases Learning and Selection**
**for** $i = 1$ to $B_1$. **do**
  i) Generate $r_{id} \in [n]$ random indices.
  ii) $[H_i, W_i] = NMF_{KL}(A(r_{id}, :), k)$.
  iii) $R_{id}(:, i) = r_{id}$; $\tilde{H} = [\tilde{H}; H_i]$.
**end for**
**1a) Choose the best set of bases**
1. Cluster the stacked matrix $\tilde{H}$ using DBSCAN.
2. Set centers of the clusters as new best bases set $\hat{H}$.
**1b) Update weights and intersection of supports**
Recompute $\{W_i\}_{i=1}^{B_1}$ wrt. $\hat{H}$ using NNLS.
**for** $l = 1$ to $n$ **do**
  $[r, c] = find(R_{id} = l)$.
  For all $r$, intersect the support of rows of $W_{[r]}$ and save in $W_{idx}$.
**end for**
**2. Weight Estimation**
**for** $i = 1$ to $B_2$ **do**
  Generate new $r'_{id} \in [n]$ random indices.
  **for** $l = 1$ to $LEN(r'_{id})$. **do**
    $w_{idx} = W_{idx}(r'_{id}(l))$;
    Compute $w_{idx}$ entries of $\{W_i\}_{i=1}^{B_1}$ using NNLS.
  **end for**
**end for**
$\hat{W}$ = entrywise-mean($\{W_i\}_{i=1}^{B_1}$).
```

$\{W_i\}_{i=1}^{B_1}$'s estimated over bootstrap samples.

First, for selection, we compute a new weight matrix W_i for each bootstrap sample $i = 1, \ldots, B_1$, using $\hat{H}$ and the nonnegative least squares (NNLS) or nonnegative LASSO regression method. For each row of weights, we then compute the intersection of the support over all bootstrap samples in which this row was considered. This gives us a sparse index matrix W_{idx} with the intersected support of each row of $\{W_i\}_{i=1}^{B_1}$ over different bootstraps.

Next, for estimation, we consider a new set of bootstrap samples (B_2) and use bagging to comput the weights of the output coefficient matrix $\hat{W}$. We use the sparse coefficient index matrix W_{idx}, the best basis $\hat{H}$, and NNLS to compute the rows of new

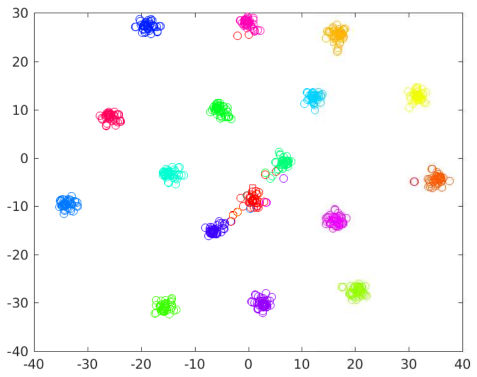


Figure 10.1: 2D visualization of clusters of bases

W_i over B_2 bootstrap samples. For each row of W_i, for all bootstrap samples in which this row is considered, we employ NNLS to compute the weights of the coefficient whose indices are in the rows of W_{idx}. The mean of the weights the across bootstrap samples is chosen as the optimal estimate of the weights, $\hat{W}$, which will be sparse (because W_{idx} is sparse), and have low-bias (no explicit regularization) and low-variance (from bagging).

Algorithm 9 describes our *UoI-NMFcluster* algorithm. The algorithm can be modified for other NMF variants and other methods for clustering.

Geometric interpretation and uniqueness

Here, we present the theoretical intuition to use clustering across bases learned from bootstrap samples to obtain more stable parts based decompositions. The uniqueness of the solutions to the NMF problem was discussed in [151], using a geometric interpretation of NMF with simplicial cones.

Geometric Interpretation: There is an unknown H-simplex whose vertices are the rows of $H \in \mathbb{R}_+^{k \times n}$. We observe m points $A \in \mathbb{R}_+^{m \times n}$ that lie in the H-simplex. The goal is to identify the vertices of the H-simplex.

An important observation in [151, 152] is that, if the input data points come from a simplex (without loss of generality), the bases learned by an NMF algorithm will be the vertices of this simplex (non-overlapping bases with separated supports). In this case, the data is called "separable".

Separability: A NMF is separable if all the vertices $H(j,:)$'s appear in the observed points $A(i,:)$'s.

The separability of data was shown to be the key required property for unique solutions for NMF. Article [152] showed that subset separability of data (a milder condition of separability) is sufficient for obtaining unique solutions.

A NMF algorithm will return a unique solution (learn the vertices) when the data are separable (uniqueness guarantees are shown in the literature only when the NMF is separable or subset separable). However, such separability conditions are hard to test, and are unlikely to hold when the data are noisy. If the data are from a simplex with added noise, the NMF algorithms may learn some of the simplex vertices as bases, along with the noise learned as one or more additional bases, because NMF is an additive model. We show in our numerical experiments that basic NMF algorithms indeed learn a few of these pure bases (vertices) and other bases are related to noise.

Generating bases from multiple bootstrap samples makes it likely that all the simplical vertices will be present with in the superset of bases, i.e., rows of $\tilde{H}$ will likely have many points near the vertices (the 'parts' bases) which are dense spatially, and a few other noisy points related to noisy bases. The noisy bases are widespread and unlikely to be near the vertices. Hence, density based clustering across all the points and using the centroids will give us the vertices of the simplex. DBSCAN ignores the few noisy points that are spread out. Figure 10.1 gives a 2D visualization obtained by tSNE algorithm [153], of the clustered (spatial) distribution of bases learned over different bootstrap samples for the Swimmer dataset, described in the next section. The colors indicate the clusters assigned by the DBSCAN algorithm with red depicting noise points. Therefore, for separable data with noise, using density based clustering and extracting the cluster centroids will likely return the vertices of the simplex, i.e., the stable part based bases.

10.3.2 Numerical Experiments on Synthetic Data

In this section, we illustrate the performance of our proposed algorithm on few synthetic datasets. We compare $UoI\text{-}NMF_{cluster}$ (with $B_1 = 20$ and $B_2 = 10$) to basic NMF (using ALS with multiple initial conditions), sparse NMF (as implemented by SPAMS library), and TSVDNMF. and for sparse NMF, we used different parameters λ

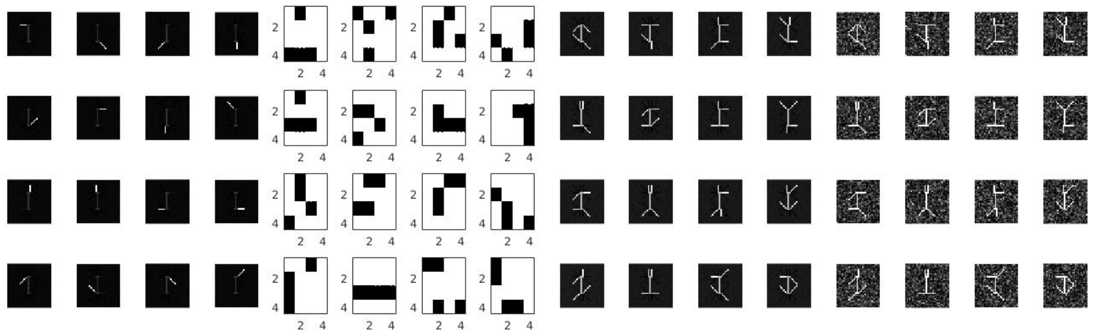


Figure 10.2: $UoI\text{-}NMF_{cluster}$ for noisy Swimmer data.

incrementally, and reported the best results. We find that $UoI\text{-}NMF_{cluster}$ yields parts-based, noise-free bases, and thus reconstructions obtained are also noiseless (denoised data).

Swimmer Dataset - In the first experiment, we consider the swimmer dataset [151], the canonical example of separable data: each image can be reconstructed from a subset of non-overlapping bases. We compared the performance of $UoI\text{-}NMF_{cluster}$ against other NMF algorithms with multiple random initial conditions for the swimmer dataset corrupted by heavy additive noise (Absolute Gaussian noise, $|\mathcal{N}(0, 0.25)|$). The dataset contains 256 images of size 32×32 each. We concatenated 10 noisy sets of these 256 images (2560 in total) as the input matrix (of size 2560×1024).

Figure 10.2 illustrates the performance of $UoI\text{-}NMF_{cluster}$ algorithm on this noisy data. The first 4×4 images of the figure show the 16 bases (parts) learned by $UoI\text{-}NMF_{cluster}$. The second set (4×4 images) displays the sparse weights estimated to reconstruct 16 randomly chosen images. The third set depicts the recovered images $\hat{A} = \hat{W}\hat{H}$, and the last 4×4 image set gives the original noisy input images A.

The resulting bases from $UoI\text{-}NMF_{cluster}$ are remarkably good parts based decompositions of the denoised data, even though the input matrix had very high noise. We see that $UoI\text{-}NMF_{cluster}$ learns all the 16 bases (parts) almost exactly. Thus, for data generated from bases that are vertices of a simplex, our algorithm yields the unique solution that exists, even when the observed data is highly noisy, and hence not separable. We also observe that the weights learned are sparse due to the intersection operation of UoI, resulting in the algorithm choosing only bases that are relevant for

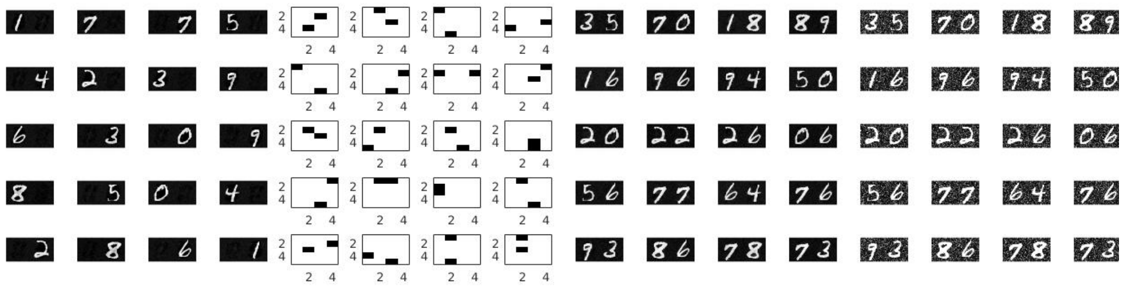


Figure 10.3: UoI results for noisy MNIST two digits data.

the reconstruction of the original data. The nonzero patterns of the weights given in Fig. 10.2 shows that exactly four bases are chosen for reconstruction for all the images. We clearly see that the recovered images are denoised versions of the noise corrupted input images. When the input data are noisy, most basic NMF algorithms tend to learn the noise as a separate bases (due to the additive nature of the factorization).

MNIST 2-digit data - Next, we use the popular handwritten digit images from the MNIST dataset [154]. The dataset contains different sets of handwritten digits from 0 to 9 (by different individuals), and in this experiment we select one such set and concatenate two of these images to form 2-digit handwritten numbers (00 to 99). We have 100 such concatenated images. We consider noise ($|N(0, 0.2)|$) corrupted images (10 repetitions, hence we have 1000 images) for training the NMF algorithms. The goal is to learn the individual digits (at units and tens place).

Figure 10.3 shows the results. The first 5×4 images show the bases learned by our algorithm. The estimated weights to reconstruct 20 randomly chosen images are given along with the reconstructions and the original noisy images. $UoI\text{-}NMF_{cluster}$ gives better single digit bases than basic NMF, as well as sparse NMF. We also observe the weights learned are quite sparse, exactly two in most cases. Note that, in contrast to the swimmer data set, in this example, the bases (digits) are not quite vertices of a simplex, and hence even noiseless data is not quite separable. Thus, the learned bases are not perfectly decorrelated (e.g., a nine and a one and a seven are all highly correlated). Yet, $UoI\text{-}NMF_{cluster}$ learns these 20 bases quite accurately.

Table 10.1 summarizes the results obtained by the different NMF algorithms (first column) on different datasets (second column). The algorithms are : basic NMF is

Table 10.1: Comparison of NMF algorithms.

Methods	Data	Error (n.less)	MSE	Error (noisy)	$nnz(\hat{W})$	$nnz(\hat{H})$
$UoI\text{-}NMF_{cluster}$	Swimmer	16.8	0.0015	195.1	4	22
basic NMF	Swimmer	54.2	0.0052	202.6	3	41.5
sparse NMF	Swimmer	60.4	0.0055	206.2	5	60
TSVDNMF	Swimmer	71.3	0.0236	240.5	3	80
$UoI\text{-}NMF_{cluster}$	MNIST	36.61	0.0029	194.3	2	105.5
basic NMF	MNIST	48.69	0.0102	153.07	3	144.5
sparse NMF	MNIST	59.06	0.0268	192.02	2	149
TSVDNMF	MNIST	78.83	0.0580	256.77	3	156

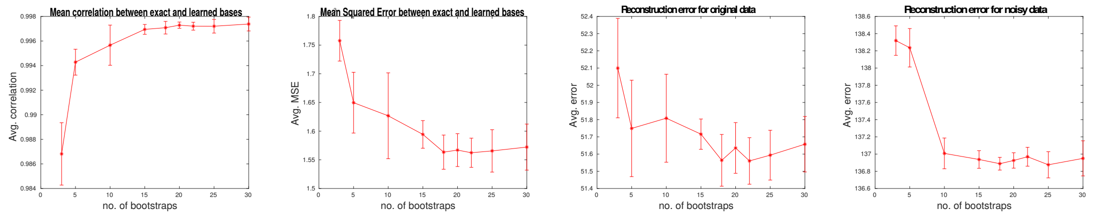


Figure 10.4: Performance wrt. number of bootstrap resamples.

the basic NMF algorithm with KL divergence error metric, $UoI\text{-}NMF_{cluster}$ is the proposed algorithm with basic NMF-KL as the inside algorithm, sparse NMF and TSVD-NMF [155].

We give the error $\|A - WH\|_F$ for the reconstruction of original noiseless data the third column and list the average mean squared error (MSE) between the exact and the learned bases in the fourth. The reconstruction error $\|A - WH\|_F$ when the noisy training data was recovered are listed in the fifth column. We also give the median nonzeros per row in the weights W and the bases H learned by the different algorithms in the last two columns respectively.

Number of bootstraps: We know that the number of bootstraps B_1 and B_2 used in $UoI\text{-}NMF_{cluster}$ are parameters which we can tune. In this experiment, we try to understand the influence of the number of bootstrap samples used on the quality of results obtained. We apply $UoI\text{-}NMF_{cluster}$ with different number of bootstraps B_1 on a noisy Swimmer dataset (five noisy sets with $\sigma^2 = 0.2$), and plot the results.

Figure 10.4 plots the mean pairwise correlation and the average MSE between the exact and the learned bases as a function of the number of bootstraps B_1 in the first two plots. The reconstruction errors for noisy and original data using the bases learned by the $UoI\text{-}NMF_{cluster}$ for the different number of bootstraps used are plotted in the last two plots. All plots show the mean and the error bars over 5 trials. We see that, as the number of bootstraps increases, the quality of bases learned improves up to a certain number and then for large number of bootstraps, the quality remains the same. This is because, as the number of bootstraps increase, the density of the clusters increase and the DBSCAN algorithm performance improves. There seems to be a number (between 15-20) beyond which, DBSCAN is able to select all 16 bases exactly. Increasing the number of bootstraps beyond will not have much effect on the quality of bases learned.

The reconstruction errors for the noisy and original data also decrease initially as the number of bootstraps increases due to improvement in the basis quality. The improvement in the results are also due to improved weight learning by the UoI framework. But, for larger B_1 (> 25), the error slightly increases for recovering noisy bases because the weights learned become very sparse due to the intersection operation. We see the peak performance occurs for around 20-22 bootstrap resamples. Hence, we chose $B_1 = 20$ in all our experiments. The effect of the number of bootstraps B_2 in the weight estimation stage will be similar to effect of the number of bootstraps in the feature estimation stage of UoI_{LASSO}. For discussion related to this, see [11]. Additional experimental results can be found in [12].

10.3.3 Experiments on Scientific Data

In the following experiments, we demonstrate the performance of $UoI\text{-}NMF_{cluster}$ in two scientific applications.

Mass Spectrometry Imaging of Mouse Brains: Mass Spectrometry Imaging (MSI) is a modern chemical imaging technique that has enabled investigation of metabolic processes at very high resolution (subcellular to centimeter range) [156]. In MSI, a laser is raster scanned across a surface and molecules are desorbed from the surface at each location. These ions/molecules are then collected and analyzed by mass spectrometry, which yields a large number of spectral images. MSI may contain spectral images with up to a million pixels and are typically collected over 10^4 to 10^6 frequency bins (m/z). Hence, such MSI data present many analysis and interpretation challenges due to the

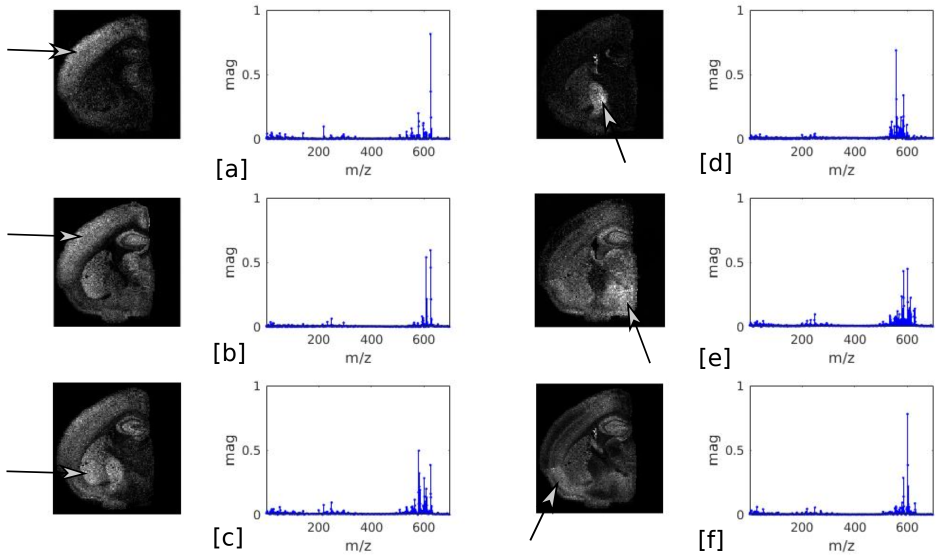


Figure 10.5: $UoI\text{-}NMF_{cluster}$ results for mouse brain MSI dataset.

size and complexity of the data. The objective of using NMF (or any other dimensionality reduction techniques) on MSI data is to reduce the large volume of measured data into easier to interpret smaller blocks. The goal is to identify important locations/pixel positions and the corresponding chemical composition.

Here, we consider the MSI data (NIMS) of the coronal section of mouse brain from OpenMSI[1] [157]. The data contains 120×122 size spectral images, computed at 80339 frequency m/z bins. The data is very noisy and preprocessing is necessary. The data were preprocessed by using background subtraction, smoothing and peak picking as mentioned in [147]. This reduces the data points from 80339 to 697, after peak picking.

Figure 10.5 shows the six bases learned by $UoI\text{-}NMF_{cluster}$ and the corresponding weight distribution learned for the respective bases. We found that many of the bases learned by $UoI\text{-}NMF_{cluster}$ corresponded to spatially localized, anatomically defined parts of the mouse brain (e.g., sensory cortex [a,b], hippocampus/putamen [c], and globus paladus [d], hypothalamus [e] and piriform cortex [f], pointed by arrows). We also observe from the weight distributions that, these parts (bases) appear at different frequency m/z bins, in line with the notion that these different regions of the brain have different chemical compositions.

[1]`https://openmsi.nersc.gov/`

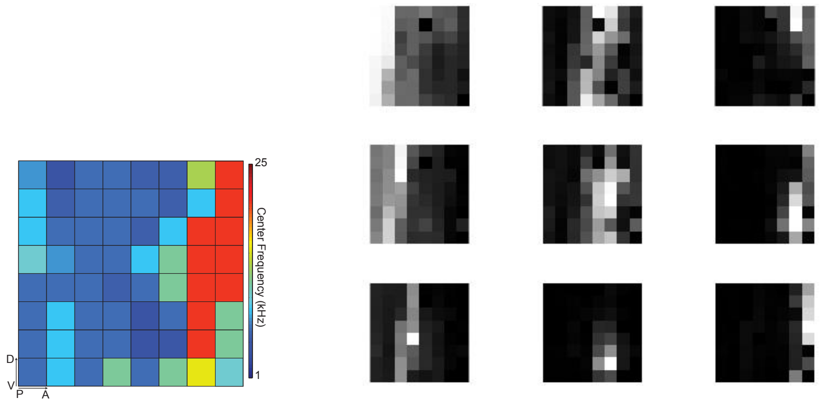


Figure 10.6: Rat ECoG recordings: Left - ToneMap of the auditory cortex. Right - $UoI\text{-}NMF_{cluster}$ bases.

Electrophysiological Data from Rat Cortex: The objective of this experiment is to learn a set of bases (channel responses/ neuron firing patterns) from the ECoG recordings for certain stimuli. With the ever increasing number of simultaneously recorded neural signals, neuroscience has seen a resurgence in the application of dimensionality reduction algorithms to summarize high-dimensional data. However, the primary method used in the field, PCA, has made the interpretation of the physical meaning of the derived axes opaque (a common critique of PCA). Here, our goal was to determine if $UoI\text{-}NMF_{cluster}$ could extract a physically meaningful bases directly from neural recordings when there is a known spatial organization of neural response properties (otherwise, how would we know what success would look like?). To this end, we applied $UoI\text{-}NMF_{cluster}$ to neural recordings taken from the auditory cortex of a rat, which has a well characterized spatial organization of frequency representations across the cortical surface (i.e., tonotopy).

In the experiment, we used the neural response recordings collected from the primary auditory cortex of an anesthetized rat using a 64 channel μECoG. Following standard procedures in the field, at each electrode, we determined the neural response by extracting the analytic amplitude from the 'high-gamma band' [70-150Hz], which correlates well with multi-unit spiking activity. These responses were z-scored relative to the baseline statistics for each channel individually. The response was for auditory stimuli consisting of 210 different sounds (30 frequencies and 7 amplitudes) with 20 repetitions (trials) each. The number of time steps used was 101. Hence, the data was of size $64 \times 4200 \times 101$. This data was preprocessed by doing peak response picking. For each stimuli, the peak response after the stimulus starts (after 40 time steps), for each channel was chosen as that channel's output for that particular stimulus. Hence, the data was reduced to 64×4200 (with each 20 set of columns corresponding to the 210 stimuli each).

From these data, for each recording channel ('pixel'), we can determine the sound frequency which gave the largest response across amplitudes (the center frequency). In Figure 10.6 (left), we color code each pixel in the array according to its center frequency (color bar on right). Here, we see a general posterior-to-anterior (left-to-right) progression of center frequencies going from low center frequencies to high center frequencies, with relative isotonic representations along the dorsal-ventral (top-to-bottom) axis. In neuroscience, this spatial organization of frequency representations is known as tonotopy. We note that, while we can summarize the responses in this way, the underlying data is more complex, with each electrode giving a graded response as a function of both amplitude and frequency, and the data on single-trials are noisy: thus, these data are not separable.

Figure 10.6(right) gives the bases learned by $UoI\text{-}NMF_{cluster}$ on this data. Bases are plotted as 8×8 grid to represent the ECoG grid for visualization as per the channel grid location, and are ordered according to the location of large values. Here, we see that the different bases reflect the tonopic organization of the underlying cortical tissue. That is, the different bases are constrained in the anterior-posterior axis (i.e., across columns) while being extensive in the dorsal-ventral axis (i.e., across rows), and generally tile the grid across the anterior-posterior axis.

10.4 UoI for CUR decomposition

In the previous section, we saw how the UoI-framework helped us to develop a novel NMF algorithm that was robust to noise and yielded interpretable parts based decompositions of the data. $UoI\text{-}NMF_{cluster}$ selects the right set of bases and estimates the weights separately to reconstruct the data optimally and avoid the reconstruction of the noise. These superior results from $UoI\text{-}NMF_{cluster}$ suggest that a similar approach may result in improved results from other data decomposition algorithms.

In the first chapter, we introduced the CUR decomposition, one of the popular dimensionality reduction method used in many applications. Here, we adapt the UoI framework for CUR decomposition to develop the two stage *UoI-CUR* algorithm. The algorithm reduces the variance of column/row sampling via the intersection operation.

UoI-CUR: The *UoI-CUR algorithm* is as follows: We consider the bootstrap resampling approach (for column selection, we subsample rows and vice versa). We compute the different subsets of columns (and rows) C_i for the different bootstrap samples $i = 1, \dots, B_1$, and for different ranks k using leverage score sampling.

Intersection: We then intersect the support (indices) of the subsets of columns (and rows) C_i over the bootstraps to obtain a smaller intersected subset $\hat{C}^{(k)}$ (for different ranks k). This intersection operation reduces the variance in sampling.

Union: Next, we obtain a larger union set of columns by taking union of the intersected subsets $\hat{C}^{(k)}$ over different ranks k.

10.4.1 Experiments - Tagging gene expressions

Analysis of gene expression DNA microarray data has become popular for studying a variety of biological processes [105]. In the microarray data, we have m genes (from m individuals possibly from different populations) and a series of n arrays probe genome-wide expression levels in n different samples, possibly under n different experimental conditions. Hence, the data from microarray experiments is represented as a matrix $A \in \mathbb{R}^{m \times n}$, where A_{ij} indicates whether the jth expression level exists for gene i. Typically, the matrix could have entries $\{-1, 0, 1\}$ indicating whether the expression exists (± 1) or not (0) and the sign indicating the order of the sequence. In chapter 5, we saw how CUR decomposition and coarsening techniques can be used to select a subset of gene expressions or single nucleotide polymorphisms (SNPs) called the *tagging SNPs* (tSNPs) which best represent the gene pools. In this section, we demonstrate how the *UoI-CUR* algorithm performs in this application.

We consider here the same two datasets which we considered in chapter 5. Table 10.2 lists the errors obtained from the three different methods, namely,UoI-CUR, basic CUR and Greedy selection [105] for different populations. The error reported is again given by $nnz(\hat{A} - A)/nnz(A)$, where A is the input encoding matrix, C is the sampled/coarsened

Table 10.2: TaggingSNP: UoI-CUR, basic CUR and Greedy selection

Data	Size	c	$UoI\text{-}CUR$	Basic CUR	Greedy CUR
Yaledataset/SORCS3	1966×53	30	0.0096	0.0323	0.0062
Yaledataset/PAH	1979×32	20	0.0165	0.0308	0.0165
Yaledataset/HOXB	1953×96	36	0.0690	0.1369	0.0272
Yaledataset/17q25	1962×63	35	0.0507	0.0895	0.0197
HapMap/SORCS3	268×307	83	0.0023	0.0624	0.0023
HapMap/PAH	266×88	42	0.0087	0.0130	0.0053
HapMap/HOXB	269×571	57	0.0840	0.1696	0.0211
HapMap/17q25	265×370	80	0.0421	0.1819	0.0162

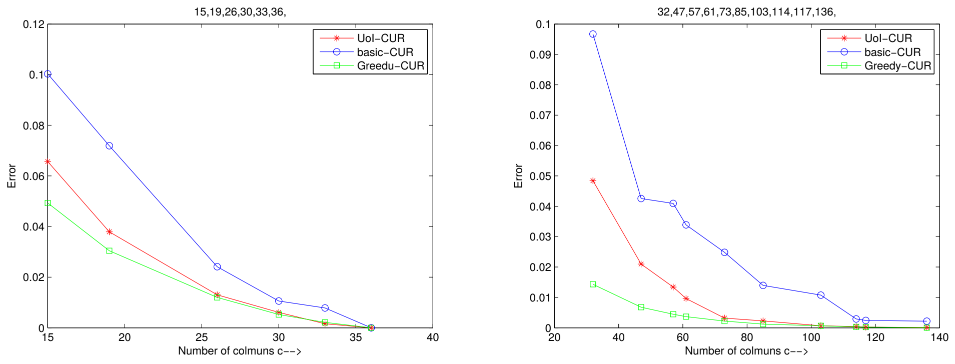


Figure 10.7: $UoI\text{-}CUR$: Error $\|A - P_C A\|_F$ as a function of the number of columns c.

matrix, $\hat{A} = CC^{\dagger}A$, is the projection of A onto C and $nnz(A)$ is the number of elements in A. Recall that the greedy algorithm is very expensive but performs very well in practice. We observe that the UoI-CUR algorithm performs better than basic CUR and the performance is comparable with the greedy algorithm in many cases.

Figure 10.7 plots the results for two of the genetics data sets considered in Table 10.2, as a function of the number of columns c. UoI resulted in reconstruction errors that were consistently lower than the base method (basic-CUR), and quickly approached an unscalable greedy algorithm (Greedy-CUR) as the number of columns c increases. Thus, in both cases, UoI improved the prediction parsimony relative to the base method.

Chapter 11

Material Informatics - Mining material data

11.1 Introduction

In recent years, as a result of the Material Genome Initiative[1], machine learning (ML) techniques have emerged among other 'material informatics' methods, for exploring materials data. Material informatics techniques based on machine learning have been shown to be inexpensive means of exploiting materials data, and can be used to examine a variety of thermodynamics properties. In the final chapter of this thesis, we apply well-known supervised regression techniques to predict properties of compounds that are hard and expensive to compute otherwise, using easily available physical, chemical and structural properties of the compounds, known as *features* in machine learning or *descriptors* in material science. The goal of this work is to help bypass time intensive calculations (some take days of computations), e.g., ab-initio calculations, used in material science. In particular, we present of a machine learning (regression) technique for prediction of the formation enthalpies of new metal alloys using easily available material data.

[1] `https://www.mgi.gov/`

11.2 Formation Enthalpy of metal alloys

The thermodynamic data of alloys such as the standard enthalpy of formation ΔH (also known as standard heat of formation) plays an important role in several applications, e.g., in the calculation of phase diagrams and materials design, in the exploration of new materials having high melting points that can be used in advanced coal-fired plants, building heat-exchangers, filters, and turbines, and many more. The heat of formation of an alloy indicates its stability, *i.e.*, a more negative enthalpy of formation implies a more stable alloy. Also, the sign of ΔH is a fundamental property that can serve as an indicator for the stability of a given alloy. Systems with a positive ΔH are only stabilized by entropy considerations. In addition, the formation enthalpies of compounds are also significant for certain high-throughput density functional theory (DFT) calculations [158]. Unfortunately, it is well known that determining such thermodynamic properties via experiments is difficult, especially for compounds with high melting points.

Since the experimental determination of thermodynamic properties of a vast combinations of elements is inefficient, recent research has focused on developing various computational approaches to predict and estimate these properties of interest. In the case of the enthalpies of formation of compounds, several different approaches have been proposed over the years. For example, we note the Hildebrand formula [159] for enthalpy of solutions, a semi-empirical model of alloy cohesion by Miedema *et al.* [160], and a modified embedded atom model for random alloys [161]. Popular among these, particularly for binary metal alloys, is Miedema's model.

Miedema and his co-authors [160, 162] developed a semi-empirical method for predicting the heat of formation of binary intermetallic compounds that contain at least one transition metal. They showed that the formation enthalpies of such binary alloys can, in general, be described in terms of a simple atomic model, that depends only on two parameters of the constituent atoms. Their model has been very successful in predicting correctly the signs for the heats of formation. However, it is less quantitative for predicting the magnitude of the enthalpy change and requires certain experimental information.

With the advent of density functional theory and its concurrent implementations for realistic computations, using first principles or *ab initio* calculations for predicting and understanding material properties has become popular [163, 164, 165]. One can compute accurate values for the formation enthalpies of compounds using such calculations. However, a major drawback of DFT calculations is the relative high computational cost, especially for a quick screening of a large database, and the need for certain prior information such as a known crystal structure.

In recent years, machine learning (ML) techniques have emerged for exploiting materials data. A popular approach in the literature is to apply tools from machine learning on certain DFT calculations to accelerate prediction of various properties of compounds [166, 167]. Ideas from machine learning have been coupled with databases of *ab initio* calculations to estimate molecular electronic properties in chemical compound space, including the enthalpy of formation of compounds [168, 169]. However, these methods still have the major disadvantage of requiring results from many DFT calculations, which may not be possible for alloys without given crystal structures, *i.e.*, amorphous or noncrystalline alloys. Recently, a machine learning approach to predict the density functional theory total energies has been implemented and these predictions are used to compute the enthalpies of formation of metal-nonmetal compounds [158].

In this chapter, we present an alternative machine learning approach to predict the formation enthalpies of binary metal alloys. The method we propose differs from previous ML techniques in that it uses readily available properties of the constituting elements (elemental properties), complemented by some basic properties of the compounds that are available in popular databases (e.g., Materials Project [2]), to predict the formation enthalpies.

A large set of (publicly available) elemental properties is considered and three different methods are explored to select (a smaller set of) appropriate elemental properties for enthalpy prediction from this large set. The three sets of elemental properties used are: (i) properties selected based on a literature study, (ii) properties obtained through sensitivity analysis. (iii) properties selected by a modified LASSO (Least Absolute Shrinkage and Selection Operator) method [170]. The first set can be viewed as a set selected based on prior physics knowledge, while the latter two are based on machine learning methods (do not take into account any physics knowledge). Our results indicate that features (elemental properties) selected based on the prior physics knowledge

[2]`https://materialsproject.org/`

perform better in predicting enthalpies than those obtained through machine learning techniques.

A well-known method exploited in machine learning and known as “Support Vector Regression” is employed for the formation enthalpy predictions. The approach proposed in this work is fast and *does not require DFT calculations*, since the model takes available properties of elements and compounds as input, and is trained and validated against (or reproduces) the formation enthalpies calculated using Miedema’s model, which are also easily available for many binary alloys. Since the Miedema’s model is itself not very accurate, the proposed machine learning approach cannot give highly accurate formation enthalpies. However, the presented method is an extremely inexpensive technique aimed at predicting formation enthalpies of new compounds (as accurately as Miedema’s model) without any empirical information. Such enthalpy predictions suffices in many applications such as new material discovery, stability analysis and melting point predictions. In applications where accurate formation enthalpies are required, these predictions can be coupled with simple DFT calculations (which are less expensive than full DFT calculations taking elemental properties as an input) to obtain accurate enthalpies.

11.3 Machine Learning for Prediction

The performance of machine learning predictions depends primarily on two aspects: the *feature selection* and the *machine learning model* used.

11.3.1 Features Selection

A quintessential step for successful predictions is identifying the key characteristics of the constituting elements (elemental features), that dictate or affect the properties of the compounds that we wish to predict. In this chapter, we consider three different approaches for feature selection.

Literature study: In order to identify a good set of elemental features that influence the formation enthalpies of compounds, let us first look at Miedema’s model [160]. It has been known for a long time that the work function ϕ is correlated to the ionization energy, the electron affinity and the electronegativity of constituting elements. While ionization energy and electron affinity are properties of isolated atoms, the electronegativity provides information about the attraction the given atom has for electrons in an ionic (or partially ionic) bond formed with another atom. For pure metals, the theoretical electron density values n_{ws} depends on bulk modulus B and molar volume V. Thus, Miedema's model suggests that the following features of the constituting elements are crucial for the prediction of the formation enthalpies: *ionization energy*, *electronegativity*, *electron density* and *molar volume*.

Another model which helps to identify the elemental features that affect the formation enthalpies, is the Hildebrand formula for the enthalpy of solution of two liquids [159]. This formula depends on two properties of the constituting liquids, namely, the *enthalpy of vaporization* of the liquids and their *molar volumes*. The formation enthalpies describe the cohesion in the metal alloys [162]. The modified embedded atom method by Ouyang *et al.* [161] uses the *cohesive energy*, *formation energy* and *atomic volumes* of pure elements to describe the work function ϕ in Miedema's model. From the above studies, we expect that the following seven elemental properties are likely to be the most influential features in predicting the formation enthalpy: *ionization energy*, *electron affinity*, *electronegativity*, *electron density*, *enthalpy of vaporization*, *cohesive energy* and *molar volume*.

Sensitivity method: A machine learning approach to identify the elemental features that provide good property predictions is to use the *'sensitivity method'* described by Saad *et al.* [171]. To verify the impact of elemental features on the enthalpy prediction accuracy, we find the sensitivity of each of the available properties of the constituting atoms (we collected $d' = 49$ properties of each element, and hence obtained $d = 2d' = 98$ features in total after concatenation to represent the binary alloys).

The sensitivity method applied to our model can be described as follows: Let $X \in \mathbb{R}^{n \times d}$ be a matrix that contains the known properties (the input features/descriptors) of the individual compounds as columns (since X is a concatenation of the d' properties of the two elements forming a compound, the number of columns is $d = 2d'$). First, for a considered feature k, we perturb the values of this feature for both elements of each compound, i.e., the vectors $X(:, k)$ and $X(:, k + d')$ are perturbed respectively by

$\epsilon \approx c10^{-8}\|[X(:,k); X(:,k+d')]\|$, where c is a random number.

Second, we calculate a new coefficient vector a_ϵ, using the least squares solution $a_\epsilon = (X^\top X)^{-1}X^\top v$, where v is a vector containing the actual formation enthalpies of the compounds. Next, the norm of the difference between the original (obtained without perturbing columns of X) and the perturbed coefficient vector $\|a_\epsilon - a\|$ is computed.

Finally, the ratio $\frac{\|a_\epsilon - a\|}{\epsilon}$ is assigned as the sensitivity measure of the k-th feature. The top seven most sensitive features for the prediction of formation enthalpies are : the *electrochemical equivalent weight*[3], *first oxidation state*, *group number*, *effective nuclear charge* (Slater's rule), *metal radius*, *electronegativity* and *distance core electron*.

LASSO method: Another alternative method used recently in the literature [172] for feature selection is the so called *compressed sensing approach*, which is a LASSO [170] type method. Given a large feature matrix $X \in \mathbb{R}^{n\times d}$, and the output vector v (property to be predicted), the LASSO method yields a sparse relation between X and v by solving the convex optimization problem

$$\arg\min_{\beta\in\mathbb{R}^d} \|v - X\beta\|_2^2 + \lambda\|\beta\|_1, \tag{11.1}$$

where the ℓ_1-norm $\left(\|\beta\|_1 = \sum_i \beta(i)\right)$ promotes the sparsity in vector β. Thus, the sparsity of vector β helps us to select the descriptors (columns of X) that best describe v in the least squares sense. However, recall that the matrix X is formed by simply concatenating the properties of the two constituting elements. Using the LASSO method directly will not guarantee selection of the same set of properties for the two elements. That is, the vector β need not have same nonzero coordinates in the first $d' = 49$ coordinates ($\beta(1:49)$) and last d' coordinates ($\beta(50:98)$). We indeed obtained different sets of features being selected for the two elements when the LASSO method was used directly in our experiments. In order to overcome this issue, we propose the following

[3]The electrochemical equivalent weight of an element is the ratio between its atomic weight and its principal valence number.

modified LASSO problem obtained by splitting vector β as $\beta = [\beta_1; \beta_2]$,

$$\min_{\beta\in\mathbb{R}^d} \|v - X\beta\|_2^2 + \mu\|\beta_1 - \beta_2]\|_2^2 + \lambda\|\beta\|_1$$
$$\text{or} \quad \min_{\beta\in\mathbb{R}^d} \|v - X\beta\|_2^2 + \mu\|J\beta\|_2^2 + \lambda\|\beta\|_1,$$

where $J = [I, -I]$ with the identity matrix I. We include the additional term $\mu\|J\beta\|_2$ to ensure that the two halves of the vector β are close (equal), such that the same set of properties is selected for the two elements (from the first 49 and the last 49 features). This modified LASSO problem is still a convex optimization problem and therefore can be easily solved using any of the available optimization packages, e.g. the CVX package [173]. The parameters λ and μ were adjusted such that the modified LASSO selects exactly seven properties from both elements, i.e., both β_1 and β_2 have exactly seven nonzero entries. The following seven properties were selected by the LASSO method for the two elements: *atomic weight, density, energy ionization first, temperature boiling, temperature melting, electronegativity* and *bulk modulus*. The modified LASSO method for property selection is also robust, i.e., changing slightly the parameters λ and μ does not give different set of features.

11.3.2 Machine Learning Model

We use a supervised learning regression method to predict the formation enthalpies of binary metal alloys.

Given n compounds and d specific features (descriptors) we build a matrix $X \in \mathbb{R}^{n\times d}$ that stores the features of each compound as a column of X. We assume that a certain property being studied, e.g., enthalpy of formation, is known for each of the n compounds. We are now presented with a new compound, which is not among the n ones already studied, and whose same d features, as those of the data, are known and stored in a vector $z \in \mathbb{R}^d$. Regression methods attempt to answer the question: "What is our best guess of the enthalpy of formation for this new compound?" Regression methods use X to build a mapping that will yield the desired property from z. In the simplest case of linear regression, this mapping is just a linear combination of the values of the features, and the coefficients of the linear combination are extracted by solving a least squares problem that involves X and the right-hand side of the properties of the n compounds.

Linear regression is often too simple model, and is rarely used to predict complex physical properties. A common and efficient regression technique used for real world data applications is the *support vector regression* or SVR [174]. SVR is a nonlinear regression technique that employs kernels to implicitly map the inputs into high-dimensional (nonlinear) feature spaces.

Since the relation between the elemental properties and the desired thermodynamic property of the compound is typically highly nonlinear, we consider a nonlinear kernel based regression method. The most suitable SVR variant for our purpose, is the ϵ-SVR method with RBF or Gaussian kernels given by

$$k(x_i, x_j) = \exp\left(-\gamma \|x_i - x_j\|^2\right).$$

For our experiments, we consider the ϵ-SVR method implemented in the **libSVM** MATLAB library [175]. For the optimal γ value in the kernel, we sweep from 0.1 to 1 with increments of 0.1 and choose the value that yields the best results (smallest error).

11.4 Results and Discussion

Here, we present our results for the prediction of formation enthalpy for transition metal alloys using the support vector regression (SVR) model. We found that, SVR outperforms the other regression approaches.

To illustrate the use of machine learning tools for the prediction of the enthalpy of formation for binary metal alloys, we considered 648 transition metal alloys whose formation enthalpies are available [162]. These formation enthalpies are computed using the Miedema *et al.* model.

Previously, we discussed feature selection. Collecting such features/properties of the constituting elements is the first step of the prediction. We acquired 49 different chemical properties of all the elements from the Database on Properties of Chemical Elements[4]. Next, six different physical properties of the 648 compounds (compound

[4]`http://phases.imet-db.ru/elements/main.aspx`

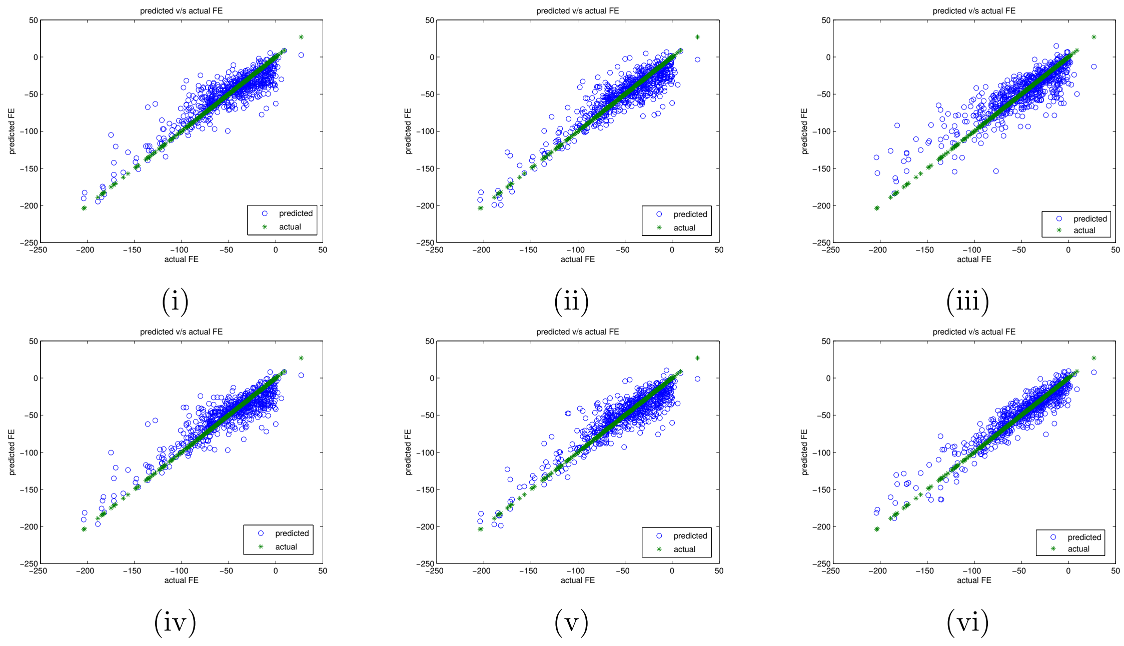


Figure 11.1: Predictions of enthalpies of formation using different feature sets.

features) were collected from the Materials Project database[5] [176].

These six properties were: *band gap, number of atoms per unit cell (nsite), volume, magnetic moment, density* and *energy-per-atom* (energy normalized to per atom in the unit cell), see [176, The Materials API]. We also collected six different crystal properties of these 648 compounds from the same database, namely the three *unit cell dimensions* a, b, c and the three *unit cell angles* α, β, γ. Various experiments were conducted using these data features. Figure 11.1 presents the results obtained from these experiments for the prediction of the formation enthalpies of these 648 transition metal alloys using the support vector regression method and various feature sets.

We considered three approaches to select the appropriate elemental features (feature selection) that affect the formation enthalpies of the metal alloys the most. The first set of features was based on the literature study. The order of concatenation of features is done based on the atomic number. Concatenating the elemental features does not incorporate the stoichiometric information (the ratios of the individual elements in the compound). We feed this information to the regression model as two new features.

[5]`https://materialsproject.org/`

Table 11.1: Relative errors in formation enthalpy predictions for different feature sets.

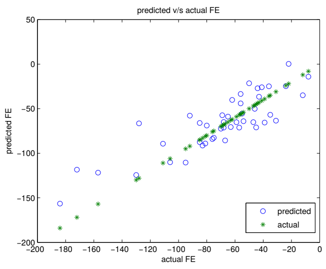
Feature Set	MAE	RMSE	MRRE	NRE	R^2
Literature	1.3809	5.5598	0.0157	0.0286	0.9563
Sensitivity	1.7657	5.7145	0.0195	0.0365	0.9468
LASSO	4.6838	9.0660	0.1049	0.2004	0.6858
Literature+compound	1.3682	5.4965	0.0156	0.0283	0.9556
Sensitivity+compound	1.6422	5.5695	0.0181	0.0340	0.9508
LASSO+compound	2.2960	6.9060	0.0580	0.1096	0.8704

That is, we include two additional features as inputs, whose values are the ratios of the first and the second element of the compound, respectively. For example, for compound `ScGe`, the values of these two features will be $[0.5, 0.5]$, and for $\texttt{ScGe}_2$, their values will be $[0.33, 0.67]$. Thus, we consider 16 features in total.

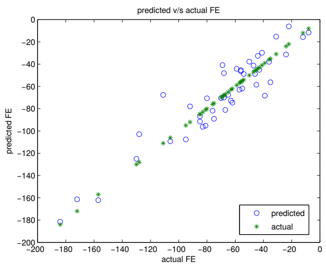
Figure 11.1(i) presents the formation enthalpies predicted by the SVR model against the actual formation enthalpies (obtained from [162]) using the literature set of elemental properties as input features. We used a 10 fold cross-validation method to predict the formation enthalpies of the 648 compounds. That is, we repeated the experiments 10 times with 10% of the dataset (around 65 compounds) chosen at random without replacement from the 648 compounds used as test data. Hence, after the 10 trials we have all 648 alloys' formation enthalpies predicted once by the model. These predicted values of the test data are those presented in the figure.

The second set of features considered was based on the sensitivity method [171] Figure 11.1(ii) presents the formation enthalpies predicted by the SVR model, using the sensitivity set of elemental features, against the actual formation enthalpies. The third set of features was selected based on the modified LASSO method. Figure 11.1(iii) presents the formation enthalpies predicted by the SVR model, using the LASSO set of elemental features (16 in total), against the actual formation enthalpies.

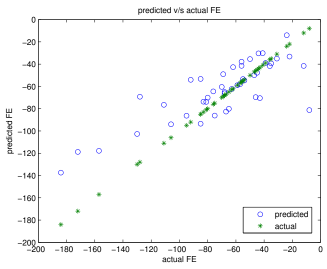
For the following numerical experiments, we considered the compound features (6 physical and 6 crystal properties of the alloys) along with the elemental and stoichiometric information as the input features for the SVR model. Figure 11.1(iv) presents predicted versus actual formation enthalpies obtained using collectively the literature and the compound feature sets ($16+12 = 28$ in total). Similarly, Figure 11.1(v) presents



(i)



(ii)



(iii)

Figure 11.2: Predictions of enthalpies of formation of Sc binary alloys.

Table 11.2: Predicted and actual formation enthalpies (FE) of Sc binary alloys.

Formula	Actual FE	Predicted FE	Formula	Actual FE	Predicted FE
	best			worst	
Sc_5Ge_3	-75	-75	$ScBe_5$	-31	-58
Sc_3Ga_2	-63	-63	Sc_3In	-39	-69
ScCd	-55	-56	ScN	-184	-152
Sc_5Sn_3	-76	-78	ScIr	-92	-60
ScAl	-68	-66	Sc_3P_2	-157	-110
ScGe	-85	-88	ScP	-172	-124

the results obtained when the sensitivity and the compound feature sets were used in the SVR model. The results obtained when the LASSO and the compound feature sets were used together are presented in Figure 11.1(vi). The various error measures obtained for each of these six experiments are listed in Table 11.1. In the table, we have the following error metrics; MAE: Mean Absolute Error, RMSE: Root Mean Square Error, MRRE: Mean-Regularized Relative Error, and NRE: Net Relative Error, see [13] for details.

SVR Model's Predictive Ability

One of the primary goals of developing new techniques for predicting properties of compounds is the hope to identify compounds with desired properties or to predict some unknown properties of existing compounds. In this experiment, we examine such predictive ability of our SVR based model by predicting the formation enthalpies of several new compounds. Let us assume that all compounds containing the element Sc (scandium) are unknown to our SVR model, i.e., we set aside all compounds containing `Sc` as a test dataset and put all other compounds into the training set. Element `Sc` was chosen since we have 45 binary alloys containing `Sc` in our initial dataset (which is a good number of instances for testing), and also because the values of the formation enthalpy of these compounds lie across a wide range $[-181, -6]$, making it a good test set. Once the model is trained on the remaining 603 compounds, we predicted the formation enthalpies (FE) of the 45 `Sc` binary alloys.

The corresponding results are presented in Figure 11.2. The plots display predicted FE values for `Sc` binary alloys using for Figure 11.2(i) the elemental properties (literature set). The results obtained using the elemental properties (sensitivity analysis), and the elemental properties (modified LASSO method) are presented in Figure 11.2(ii)–(iii), respectively. Table 11.2 list the compounds' chemical formula, the predicted and the actual formation enthalpy values (in $\left[\frac{\text{kJ}}{\text{mol}}\right]$) of the top six closest (best) predictions (left side) and the bottom six farthest (worst) predictions (right side) for the case of `Sc` binary alloys using the 14(+2) elemental properties (literature set) and the compound properties. We have 73% of the 45 enthalpy predictions within the mean-regularized relative error of 0.1 (10% relative error) and 89% within 0.15 (15% relative error). We observe that the values of formation enthalpies predicted by the SVR model are very close to the values obtained using Miedema's model and the "worst" predictions in Tables 11.2 include some alloys of `Sc` with heavy elements, i.e., `Bi`, `Pd` and `Ir`. This experiment illustrates the ability of our SVR model to predict formation enthalpies of the new compounds. Additional experimental results and details are reported in [13].

Observations: The aforementioned experimental results lead to the following observations. Firstly, we note that the three feature selection methods select three different sets of features with little overlap. This shows that: a) there are multiple sets of elemental features that are likely to influence the formation enthalpy of the alloys; b) the machine learning features are not the same as those selected based on a-priori knowledge of underlying physics; c) the two machine learning feature sets also differ. Our main observation is that predictions based on the literature set (based on prior knowledge) are better than the ones obtained using the machine learning sets. Clearly, the fourth feature set (literature+compound) yields the best results amongst all the experiments.

This shows that coupling actual knowledge of relevant physics (domain knowledge) with machine learning provides improved performance. This is likely because the machine learning methods attempt to find a linear relation between the features and the target property. However, the actual relation between the different properties of a compound will typically be highly nonlinear. Hence, we observe that coupling prior physics (domain) knowledge with machine learning methods tend to give better results than using pure machine learning features. We also observe that the different machine learning methods do not yield same results (do not agree with each other). The features selected by the sensitivity method differs from the ones selected by the modified LASSO.

Chapter 12

Conclusion

This thesis presented a number of methods for various problems in machine learning and other areas involving large dimensional matrices and scientific data.

Matrix spectrum problems: In the first part, we demonstrated how combining ideas from numerical linear algebra and stochastic methods, we can develop computationally inexpensive methods for matrix trace and spectrum related problems that are scalable to very large matrices. These methods are fast and require no matrix decompositions, making them appealing for many large data applications. Another key advantage of these methods is that, they require only matrix-vector products and are easily parallelizable. Hence, these methods have advantages in parallel and distributed settings. We also presented a novel rank estimator which along with the Krylov subspace methods simultaneously estimates the numerical rank of matrices and approximates the associated principal subspace. The method can be used as a stopping criterion for the Krylov subspace methods, and unlike prior methods, it does not require computing the complete eigen-decomposition for rank estimation.

Future Work: In section 1.2, we saw how computing matrix functions is related to computing the function at the eigenvalues of the matrix. Interesting future work includes developing matrix function approximation methods that use certain information about the matrix spectrum to obtain better approximation of the function at points that are closer to the eigenvalues of the matrix. In particular, we can first compute the cumulative spectral density (CDOS) of the matrix (related to DOS discussed in chapter 3) inexpensively. Then, we can use this CDOS to obtain better function approximation by: a) *Barycentric CDOS interpolation*: approximating the function f using Barycentric interpolation [57], where the interpolation points are chosen using the inverse of the CDOS. b) *Discrete spectrum specific orthogonal polynomials*: expanding the function using discrete orthogonal polynomials, where the polynomials are orthogonal wrt. the spectral distribution (CDOS) of the matrix.

Matrix approximation: We illustrated how the mutlilevel coarsening idea can be adapted for matrix approximation problems such as computing the partial SVD, column subset selection, and graph sparsification. This method exploits the structure of the input matrix and yields superior approximations. Coarsening is also inexpensive compared to leverage score sampling, and often yields comparable results. While coarsening has traditionally been exploited in a completely different context to devise multilevel schemes for sparse systems as well as for graph partitioning, it appears that the same principles offer a tremendous potential for solving problems related to data. We also demonstrated how rank shrinkage (a linear algebra related idea) can be used to obtain improved incoherent dictionaries in the dictionary learning problem.

Future Work: Interesting future work here includes adapting the coarsening strategy presented in the thesis for matrix approximations in online and streaming settings. In the streaming model, the columns of the input matrix can be accessed and processed only once and storage is severely limited. The idea is to maintain a small sketch of the streaming (or online) data based on column matching, similar to our coarsening strategy. This method will improve on the existing "frequent directions" method for deterministic sketching for streaming data.

Applications of codes: The second part of the thesis was dedicated to presenting new nontraditional applications of codes, primarily in solving problems related to machine learning and high dimensional data analysis. We demonstrated that certain error correcting codes (which are almost deterministic) mimic the randomness properties desired for sampling data matrices. Using this, we illustrated how codes can be used to obtain low rank approximations and solve least squares regression problems. The proposed approach has several advantages such as (a) reduced randomness, (b) logarithmic factor gain in sampling complexity, (c) efficient implementation in parallel and distributed settings and others. We also demonstrated that codewords from BCH codes, a popular coding scheme, performs exceptionally well in the group testing problem, and outperforms random GT schemes. Lastly, we combined the ideas of group testing and codes to present a new algorithm for large scale multilabel classification. The proposed algorithm has many promising advantages such as (a) simple prediction algorithm that is very inexpensive, (b) operates on the binary alphabet, hence leverages efficient binary classifiers, and (c) exploits the error correction capabilities of codes for the first time to correct prediction errors.

Future Work: Many important problems such as experimental design, Bayesian optimality, column subset selection (CSSP), sparse modeling and others, can be naturally formulated as a *cardinality constrained* maximization problem. Here, a set function $F(S)$ is maximized under a K-cardinality constraint, i.e., we wish to solve,

$$\max_{S\subset V, |S|\leq K} F(S),$$

where $V = \{v_1, \dots, v_n\}$ is the large set of possible inputs. An interesting future work includes developing a new method which extends the group testing idea for solving such *cardinality constrained* optimization problems.

ML for science: The third part of the thesis focussed on developing algorithms for interpretable learning from scientific data. We presented a new statistical machine learning framework called Union of Intersections (UoI) for interpretable learning and prediction. The advantages of the framework were demonstrated in regression and CUR decompositions, and also its applicability to scientific applications. We also developed a new nonnegative matrix factorization (NMF) algorithm, $UoI\text{-}NMF_{cluster}$, that a) yields more interpretable results, b) gives sparser solutions, and c) is robust to highly noisy data. The proposed algorithm obtained interpretable decompositions, capturing the known characteristics embedded within the noisy neurophysiological data. Finally, we also illustrated how machine learning (regression) techniques can be employed for the prediction of formation enthalpies of new metal alloys using easily available material data. Such work helps us bypass time intensive calculations, e.g., ab-initio calculations, used in material science.

Future Work: As future work, we are exploring tools such as dictionary learning and neural networks for machine learning in Neuroscience and other scientific applications. In particular, we use a constrained (nonnegative and incoherent) dictionary learning algorithm to learn overcomplete representations of neurophysiological datasets. The constraints help us learn dictionary atoms that are unique and likely more interpretable from Neurosience viewpoint. Next, these overcomplete dictionaries can be coupled with neural networks (as auto-encoders) to do classifications and predictions.

References

[1] Shashanka Ubaru, Jie Chen, and Yousef Saad. Fast estimation of $tr(f(a))$ via stochastic lanczos quadrature. *SIAM Journal on Matrix Analysis and Applications*, 38(4):1075–1099, 2017, https://doi.org/10.1137/16M1104974.

[2] Shashanka Ubaru and Yousef Saad. Fast methods for estimating the numerical rank of large matrices. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 468–477, 2016.

[3] Shashanka Ubaru, Yousef Saad, and Abd-Krim Seghouane. Fast estimation of approximate matrix ranks using spectral densities. *Neural Computation*, 29(5):1317–1351, 2017.

[4] Shashanka Ubaru, Abd-Krim Seghouane, and Yousef Saad. Dimension estimation for Krylov subspace approximation of covariance matrices. *In submission*, 2018.

[5] Shashanka Ubaru and Yousef Saad. Sampling and multilevel coarsening algorithms for fast matrix approximations. *arXiv preprint arXiv:1711.00439*, 2017. Submitted.

[6] Shashanka Ubaru, Abd-Krim Seghouane, and Yousef Saad. Improving the incoherence of a learned dictionary via rank shrinkage. *Neural Computation*, 29(1):263–285, 2017.

[7] Shashanka Ubaru, Arya Mazumdar, and Yousef Saad. Low rank approximation using error correcting coding matrices. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 702–710, 2015.

[8] Shashanka Ubaru, Arya Mazumdar, and Yousef Saad. Low rank approximation and decomposition of large matrices using error correcting codes. *IEEE Transactions on Information Theory*, 63(9):5544–5558, 2017.

[9] Shashanka Ubaru, Arya Mazumdar, and Alexander Barg. Group testing schemes from low-weight codewords of BCH codes. In *Information Theory (ISIT), 2016 IEEE International Symposium on*, pages 2863–2867. IEEE, 2016.

[10] Shashanka Ubaru and Arya Mazumdar. Multilabel classification with group testing and codes. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3492–3501, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[11] Kristofer E. Bouchard, Alejandro F. Bujan, Farbod Roosta-Khorasani, Shashanka Ubaru, Prabhat, Antoine M. Snijders, Jian-Hua Mao, Edward F. Chang, Michael W. Mahoney, and Sharmodeep Bhattacharyya. Union of Intersections (UoI) Method for Interpretable Data Driven Discovery and Prediction. In *Advances in neural information processing systems (NIPS)*, 2017.

[12] Shashanka Ubaru, Kesheng Wu, and Kristofer E. Bouchard. *UoI-$NMF_{cluster}$*: A Robust Nonnegative Matrix Factorization Algorithm for Improved Parts-Based Decomposition and Reconstruction of Noisy Data. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 241–248, Dec 2017.

[13] Shashanka Ubaru, Agnieszka Miedlar, Yousef Saad, and James R Chelikowsky. Formation enthalpies for transition metal alloys using machine learning. *Physical Review B*, 95(21):214102, 2017.

[14] Zhaojun Bai, Gark Fahey, and Gene Golub. Some large-scale matrix computation problems. *Journal of Computational and Applied Mathematics*, 74(1):71–89, 1996.

[15] Gene H. Golub and Gerard Meurant. *Matrices, moments and quadrature with applications.* Princeton University Press, 2009.

[16] V Kalantzis, Constantine Bekas, Alessandro Curioni, and Efstratios Gallopoulos. Accelerating data uncertainty quantification by solving linear systems with multiple right-hand sides. *Numerical Algorithms*, 62(4):637–653, 2013.

[17] Insu Han, Dmitry Malioutov, and Jinwoo Shin. Large-scale log-determinant computation through stochastic chebyshev expansions. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 908–917, 2015.

[18] Nicholas J Higham. *Functions of matrices: theory and computation.* SIAM, 2008.

[19] Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 19(2):433–450, 1990.

[20] H. Avron and S. Toledo. Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix. *Journal of the ACM*, 58(2):8, 2011.

[21] Farbod Roosta-Khorasani and Uri Ascher. Improved bounds on sample size for implicit matrix trace estimators. *Foundations of Computational Mathematics*, pages 1–26, 2014.

[22] I. Markovsky. *Low Rank Approximation: Algorithms, Implementation, Applications.* Springer, 2012.

[23] Ian Jolliffe. *Principal component analysis.* Wiley Online Library.

[24] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[25] Christos Boutsidis, Michael W Mahoney, and Petros Drineas. An improved approximation algorithm for the column subset selection problem. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 968–977. Society for Industrial and Applied Mathematics, 2009.

[26] Michael W Mahoney and Petros Drineas. Cur matrix decompositions for improved data analysis. *Proceedings of the National Academy of Sciences*, 106(3):697–702, 2009.

[27] Petros Drineas, Michael W Mahoney, and S Muthukrishnan. Relative-error cur matrix decompositions. *SIAM Journal on Matrix Analysis and Applications*, 30(2):844–881, 2008.

[28] David P Woodruff. Sketching as a tool for numerical linear algebra. *Theoretical Computer Science*, 10(1-2):1–157, 2014.

[29] V. Krishnamurthy, M. Faloutsos, M. Chrobak, L. Lao, J-H Cui, and A. G. Percus. Reducing large internet topologies for faster simulations. In *NETWORKING 2005, LNCS 3462*, page 328341, 2005.

[30] Venu Satuluri, Srinivasan Parthasarathy, and Yiye Ruan. Local graph sparsification for scalable clustering. In *Proceedings of SGMOD'11, Athens, Greece*, June 11-16, 2011.

[31] Daniel A Spielman and Shang-Hua Teng. Spectral sparsification of graphs. *SIAM Journal on Computing*, 40(4):981–1025, 2011.

[32] Michael Kapralov, Yin Tat Lee, Cameron Musco, Christopher Musco, and Aaron Sidford. Single pass spectral sparsification in dynamic streams. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 561–570. IEEE, 2014.

[33] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[34] Patrik O Hoyer. Non-negative matrix factorization with sparseness constraints. *The Journal of Machine Learning Research*, 5:1457–1469, 2004.

[35] Hyunsoo Kim and Haesun Park. Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis. *Bioinformatics*, 23(12):1495–1502, 2007.

[36] Paul D O'grady and Barak A Pearlmutter. Convolutive non-negative matrix factorisation with a sparseness constraint. In *2006 16th IEEE Signal Processing Society Workshop on Machine Learning for Signal Processing*, pages 427–432. IEEE, 2006.

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

[38] Ron Rubinstein, Alfred M Bruckstein, and Michael Elad. Dictionaries for sparse representation modeling. *Proceedings of the IEEE*, 98(6):1045–1057, 2010.

[39] Meisam Razaviyayn, Hung-Wei Tseng, and Zhi-Quan Luo. Computational intractability of dictionary learning for sparse representation. *arXiv preprint arXiv:1511.01776*, 2015.

[40] Florence Jessie MacWilliams and Neil James Alexander Sloane. *The theory of error-correcting codes*, volume 16. Elsevier, 1977.

[41] Raj Chandra Bose and Dwijendra K Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and control*, 3(1):68–79, 1960.

[42] Arya Mazumdar. *Combinatorial methods in coding theory*. PhD thesis, University of Maryland, 2011.

[43] Philippe Delsarte and Vladimir I. Levenshtein. Association schemes and coding theory. *Information Theory, IEEE Transactions on*, 44(6):2477–2504, 1998.

[44] Ernesto Estrada. Characterization of 3d molecular structure. *Chemical Physics Letters*, 319(5):713–718, 2000.

[45] C.E. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[46] Jason V Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning*, pages 209–216. ACM, 2007.

[47] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[48] Erlend Aune, Daniel P Simpson, and Jo Eidsvik. Parameter estimation in high dimensional Gaussian distributions. *Statistics and Computing*, 24(2):247–263, 2014.

[49] Christos Boutsidis, Petros Drineas, Prabhanjan Kambadur, Eugenia-Maria Kontopoulou, and Anastasios Zouzias. A randomized algorithm for approximating the log determinant of a symmetric positive definite matrix. *Linear Algebra and its Applications*, 533:95–119, 2017.

[50] Emmanuel J Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717–772, 2009.

[51] Moritz Hardt, Katrina Ligett, and Frank McSherry. A simple and practical algorithm for differentially private data release. In *Advances in Neural Information Processing Systems*, pages 2339–2347, 2012.

[52] Yi Li, Huy L Nguyên, and David P Woodruff. On sketching matrix norms and the top singular vector. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1562–1581. SIAM, 2014.

[53] Lingfei Wu, Jesse Laeuchli, Vassilis Kalantzis, Andreas Stathopoulos, and Efstratios Gallopoulos. Estimating the trace of the matrix inverse by interpolating from the diagonal of an approximate inverse. *Journal of Computational Physics*, 326:828–844, 2016.

[54] Ernesto Estrada and Naomichi Hatano. Statistical-mechanical approach to subgraph centrality in complex networks. *Chemical Physics Letters*, 439(1):247–251, 2007.

[55] Ramon Carbó-Dorca. Smooth function topological structure descriptors based on graph-spectra. *Journal of Mathematical Chemistry*, 44(2):373–378, 2008.

[56] Gene H. Golub and Charles F Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.

[57] Lloyd N Trefethen. *Approximation theory and approximation practice*. Siam, 2013.

[58] Edoardo Di Napoli, Eric Polizzi, and Yousef Saad. Efficient estimation of eigenvalue counts in an interval. *ArXiv preprint ArXiv:1308.4275*, 2013.

[59] Yunong Zhang and William E Leithead. Approximate implementation of the logarithm of the matrix determinant in Gaussian process regression. *journal of Statistical Computation and Simulation*, 77(4):329–348, 2007.

[60] Nicholas Hale, Nicholas J Higham, and Lloyd N Trefethen. Computing a$\hat{}\alpha$,\log(a), and related matrix functions by contour integrals. *SIAM Journal on Numerical Analysis*, 46(5):2505–2523, 2008.

[61] Cameron Musco, Praneeth Netrapalli, Aaron Sidford, Shashanka Ubaru, and David P Woodruff. Spectrum approximation beyond fast matrix multiplication: Algorithms and hardness. *arXiv preprint arXiv:1704.04163*, 2017.

[62] Insu Han, Dmitry Malioutov, Haim Avron, and Jinwoo Shin. Approximating spectral sums of large-scale matrices using stochastic chebyshev approximations. *SIAM Journal on Scientific Computing*, 39(4):A1558–A1585, 2017.

[63] Lin Lin, Yousef Saad, and Chao Yang. Approximating spectral densities of large matrices. *SIAM Review*, 58(1):34–65, 2016.

[64] Lin-Wang Wang. Calculating the density of states and optical-absorption spectra of large quantum systems by the plane-wave moments method. *Physical Review B*, 49(15):10154, 1994.

[65] Philip Rabinowitz. Rough and ready error estimates in Gaussian integration of analytic functions. *Communications of the ACM*, 12(5):268–270, 1969.

[66] Timothy A Davis and Yifan Hu. The University of Florida sparse matrix collection. *ACM Transactions on Mathematical Software (TOMS)*, 38(1):1, 2011.

[67] N. Halko, P. Martinsson, and J. Tropp. Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions. *SIAM Review*, 53(2):217–288, 2011, http://dx.doi.org/10.1137/090771806.

[68] X.G. Doukopoulos and G.V. Moustakides. Fast and stable subspace tracking. *IEEE Transactions on Signal Processing*, 56(4):1452–1465, April 2008.

[69] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *The Journal of Machine Learning Research*, 7:551–585, 2006.

[70] Rajkumar Arora, A Cotter, Karen Livescu, and Nathan Srebro. Stochastic optimization for PCA and PLS. In *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on*, pages 861–868. IEEE, 2012.

[71] Justin P Haldar and Diego Hernando. Rank-constrained solutions to linear matrix equations using power factorization. *Signal Processing Letters, IEEE*, 16(7):584–587, 2009.

[72] P. Hansen. *Rank-Deficient and Discrete Ill-Posed Problems.* Society for Industrial and Applied Mathematics, 1998, http://epubs.siam.org/doi/pdf/10.1137/1.9780898719697.

[73] Tony F Chan. Rank revealing QR factorizations. *Linear algebra and its applications*, 88:67–82, 1987.

[74] Gonzalo Camba-Méndez and George Kapetanios. Statistical tests and estimators of the rank of a matrix and their applications in econometric modelling. 2008.

[75] Patrick O Perry and Patrick J Wolfe. Minimax rank estimation for subspace tracking. *Selected Topics in Signal Processing, IEEE Journal of*, 4(3):504–513, 2010.

[76] S. Kritchman and B. Nadler. Non-Parametric Detection of the Number of Signals: Hypothesis Testing and Random Matrix Theory. *IEEE Transactions on Signal Processing*, 57(10):3930–3941, Oct 2009.

[77] Efstathia Bura and R Dennis Cook. Rank estimation in reduced-rank regression. *Journal of Multivariate Analysis*, 87(1):159–176, 2003.

[78] Edward Hannan. Estimating the dimension of a linear system. *Journal of Multivariate analysis*, 11(4):459–473, 1981.

[79] Gene Golub, Virginia Klema, and Gilbert W Stewart. Rank degeneracy and least squares problems. Technical report, DTIC Document, 1976.

[80] R. McWeeny. Some recent advances in density matrix theory. *Rev. Mod. Phys.*, 32:335–369, 1960.

[81] Yousef Saad. Filtered conjugate residual-type algorithms with applications. *SIAM Journal on Matrix Analysis and Applications*, 28(3):845–870, 2006.

[82] C Lanczos. *Applied analysis.* Prentice-Hall, Englewood Cliffs, NJ, 1956.

[83] Sergey Viktorovich Alyukov. Approximation of step functions in problems of mathematical modeling. *Mathematical Models and Computer Simulations*, 3(5):661–669, 2011.

[84] Mati Wax and Thomas Kailath. Detection of signals by information theoretic criteria. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 33(2):387–392, 1985.

[85] Shira Kritchman and Boaz Nadler. Non-parametric detection of the number of signals: Hypothesis testing and random matrix theory. *IEEE Transactions on Signal Processing*, 57(10):3930–3941, 2009.

[86] Milan Meloun, Jindřich Čapek, Petr Mikšık, and Richard G Brereton. Critical comparison of methods predicting the number of components in spectroscopic data. *Analytica Chimica Acta*, 423(1):51–68, 2000.

[87] Nick Patterson, Alkes L Price, and David Reich. Population structure and eigenanalysis. *PLoS genetics*, 2(12):e190, 2006.

[88] Yousef Saad. *Numerical Methods for Large Eigenvalue Problems- classics edition.* SIAM, Philadelpha, PA, 2011.

[89] Guanghan Xu and Thomas Kailath. Fast estimation of principal eigenspace using lanczos algorithm. *SIAM Journal on Matrix Analysis and Applications*, 15(3):974–994, 1994.

[90] Guanghan Xu and Thomas Kailath. Fast subspace decomposition. *IEEE Transactions on Signal Processing*, 42(3):539–551, 1994.

[91] Robb J Muirhead. *Aspects of multivariate statistical theory*, volume 197. John Wiley & Sons, 2009.

[92] Kurt Johansson. Shape fluctuations and random matrices. *Communications in mathematical physics*, 209(2):437–476, 2000.

[93] Cameron Musco and Christopher Musco. Randomized block krylov methods for stronger and faster approximate singular value decomposition. In *Advances in Neural Information Processing Systems*, pages 1396–1404, 2015.

[94] Theodore Wilbur Anderson. Asymptotic theory for principal component analysis. *The Annals of Mathematical Statistics*, 34(1):122–148, 1963.

[95] Shira Kritchman and Boaz Nadler. Determining the number of components in a factor model from limited noisy data. *Chemometrics and Intelligent Laboratory Systems*, 94(1):19–32, 2008.

[96] Petros Drineas, Ravi Kannan, and Michael W Mahoney. Fast Monte Carlo algorithms for matrices II: Computing a low-rank approximation to a matrix. *SIAM Journal on Computing*, 36(1):158–183, 2006.

[97] Petros Drineas, Alan Frieze, Ravi Kannan, Santosh Vempala, and V Vinay. Clustering large graphs via the singular value decomposition. *Machine learning*, 56(1-3):9–33, 2004.

[98] Bruce Hendrickson and Robert Leland. A multilevel algorithm for partitioning graphs. In *Proceedings of the 1995 ACM/IEEE conference on Supercomputing*, page 28. ACM, 1995.

[99] Michael W Berry, Susan T Dumais, and Gavin W O'Brien. Using linear algebra for intelligent information retrieval. *SIAM review*, 37(4):573–595, 1995.

[100] Effrosini Kokiopoulou and Yousef Saad. Polynomial filtering in latent semantic indexing for information retrieval. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 104–111. ACM, 2004.

[101] S. Sakellaridi, H. R. Fang, and Y. Saad. Graph-based multilevel dimensionality reduction with applications to eigenfaces and latent semantic indexing. In M. Arif Wani, editor, *Proceedings of Int. Conf. Mach. Learn. Appls. (ICMLA), 2008*, pages 194–200. IEEE comp. Soc., 2008.

[102] Haw-ren Fang, Sophia Sakellaridi, and Yousef Saad. Multilevel manifold learning with application to spectral clustering. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 419–428. ACM, 2010.

[103] Suely Oliveira and Sang-Cheol Seok. A multi-level approach for document clustering. *Computational Science–ICCS 2005*, pages 23–32, 2005.

[104] Orly Alter, Patrick O Brown, and David Botstein. Singular value decomposition for genome-wide expression data processing and modeling. *Proceedings of the National Academy of Sciences*, 97(18):10101–10106, 2000.

[105] Peristera Paschou, Michael W Mahoney, Asif Javed, Judith R Kidd, Andrew J Pakstis, Sheng Gu, Kenneth K Kidd, and Petros Drineas. Intra-and interpopulation genotype reconstruction from tagging snps. *Genome Research*, 17(1):96–107, 2007.

[106] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. In *Data Warehousing and Mining: Concepts, Methodologies, Tools, and Applications*, pages 64–74. IGI Global, 2008.

[107] Farbound Tai and Hsuan-Tien Lin. Multilabel classification with principal label space transformation. *Neural Computation*, 24(9):2508–2542, 2012.

[108] Wei Bi and James Tin Yau Kwok. Efficient multi-label classification with many labels. In *30th International Conference on Machine Learning, ICML 2013*, pages 405–413, 2013.

[109] U. V. Catalyurek and C. Aykanat. Hypergraph-partitioning based decomposition for parallel sparse-matrix vector multiplication. *IEEE Transaction on Parallel and Distributed Systems*, 10(7):673–693, 1999.

[110] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning with hypergraphs: clustering, classification, and embedding. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, pages 1601–1608. MIT Press, 2006.

[111] Yousef Saad. Finding exact and approximate block structures for ilu preconditioning. *SIAM Journal on Scientific Computing*, 24(4):1107–1123, 2003.

[112] K. Engan, S. O. Aase, and J. Hakon-Husoy. Method of optimal directions for frame design. *IEEE Int. Conference on Acoustics, Speech, and Signal Processing*, pages 2443–2446, 1999.

[113] I. Ramirez, F. Lecumberry, and G. Sapiro. Universal priors for sparse modeling. *In proceedings of IEEE CAMSAP, 2009*, pages 197–200, 2009.

[114] B. Mailhe, D. Barchiesi, and M. D. Plumbley. INK-SVD: learning incoherent dictionaries for sparse representations. *In proceedings of IEEE International Conference on Acoustic Speech and signal Processing, ICASSP*, pages 3573–3576, 2012.

[115] Scott Shaobing Chen, David L Donoho, and Michael A Saunders. Atomic decomposition by basis pursuit. *SIAM review*, 43(1):129–159, 2001.

[116] Joel A Tropp and Anna C Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *Information Theory, IEEE Transactions on*, 53(12):4655–4666, 2007.

[117] D. Barchiesi and M. D. Plumbley. Learning incoherent dictionaries for sparse approximation using iterative projections and rotations. *IEEE Transactions on Signal Processing*, 61:2055–2065, 2013.

[118] L. Breiman. Best subset repression using the nonnegative garrotte. *Technometrics*, 37:373–384, 1995.

[119] Kenneth L Clarkson and David P Woodruff. Numerical linear algebra in the streaming model. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 205–214. ACM, 2009.

[120] Nam H Nguyen, Thong T Do, and Trac D Tran. A fast and efficient algorithm for low-rank approximation of a matrix. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 215–224. ACM, 2009.

[121] Franco Woolfe, Edo Liberty, Vladimir Rokhlin, and Mark Tygert. A fast randomized algorithm for the approximation of matrices. *Applied and Computational Harmonic Analysis*, 25(3):335–366, 2008.

[122] Jiyan Yang, Xiangrui Meng, and Michael W Mahoney. Implementing randomized matrix algorithms in parallel and distributed environments. *Proceedings of the IEEE*, 104(1):58–92, 2016.

[123] Joel A Tropp. Improved analysis of the subsampled randomized hadamard transform. *Advances in Adaptive Data Analysis*, 3(01):115–126, 2011.

[124] Nir Ailon and Edo Liberty. Fast dimension reduction using rademacher series on dual bch codes. *Discrete & Computational Geometry*, 42(4):615–630, 2009.

[125] William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

[126] Tamas Sarlos. Improved approximation algorithms for large matrices via random projections. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 143–152. IEEE, 2006.

[127] M. Gu. Subspace iteration randomization and singular value problems. *SIAM Journal on Scientific Computing*, 37(3):A1139–A1173, 2015, http://dx.doi.org/10.1137/130938700.

[128] D. Z. Du and F.K. Hwang. *Combinatorial group testing and its applications.* World Scientific, 2nd edition, 2000.

[129] W Kautz and R. Singleton. Nonrandom binary superimposed codes. *IEEE Trans. Inform. Theory*, 10(4):363–377, 1964.

[130] A.G. D'yachkov, A. J. Macula, and V.V. Rykov. New applications and results of superimposed code theory arising from the potentialities of molecular biology. In *Numbers, information and complexity (Bielefeld, 1998)*, pages 265–282. Kluwer Acad. Publ., Boston, MA, 2000.

[131] V. M. Sidelnikov and O. Yu. Prikhodov. On the construction of (w, r) cover-free codes. *Probl. Inform. Trans.*, 45(1):36–40, 2009.

[132] M. B. Malyutov. Mathematical models and results in the theory of screening experiments. In *Problems of Cybernetics*, number 35, pages 5–69. USSR Academy of Sciences, Moscow, 1977. in Russian.

[133] A. Mazumdar. Nonadaptive group testing with random set of defectives via constant-weight codes. arXiv:1503:03597.

[134] Arya Mazumdar. On almost disjunct matrices for group testing. In *Algorithms and Computation*, pages 649–658. Springer, 2012.

[135] A. Barg and A. Mazumdar. Almost disjunct matrices from codes and designs. Online at arXiv:1510.02873.

[136] Changhu Wang, Shuicheng Yan, Lei Zhang, and Hong-Jiang Zhang. Multi-label sparse coding for automatic image annotation. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1643–1650. IEEE, 2009.

[137] Rahul Agrawal, Archit Gupta, Yashoteja Prabhu, and Manik Varma. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In *Proceedings of the 22nd international conference on World Wide Web*, pages 13–24. ACM, 2013.

[138] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Effective and efficient multilabel classification in domains with large number of labels. In *ECML/PKDD 2008 Workshop on Mining Multidimensional Data (MMD08)*, 2008.

[139] Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. Sparse local embeddings for extreme multi-label classification. In *Advances in Neural Information Processing Systems*, pages 730–738, 2015.

[140] Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of artificial intelligence research*, 2:263–286, 1995.

[141] Daniel Hsu, Sham M Kakade, John Langford, and Tong Zhang. Multi-label prediction via compressed sensing. *NIPS*, 22:772–780, 2009.

[142] Rohit Babbar and Bernhard Schölkopf. Dismec: Distributed sparse machines for extreme multi-label classification. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 721–729. ACM, 2017.

[143] A. Mazumdar. Nonadaptive group testing with random set of defectives. *IEEE Transactions on Information Theory*, 62(12):7522–7531, Dec 2016.

[144] Mahdi Cheraghchi. Derandomization and group testing. In *Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on*, pages 991–997. IEEE, 2010.

[145] Michael Capalbo, Omer Reingold, Salil Vadhan, and Avi Wigderson. Randomness conductors and constant-degree lossless expanders. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 659–668. ACM, 2002.

[146] Siqi Wu, Antony Joseph, Ann S. Hammonds, Susan E. Celniker, Bin Yu, and Erwin Frise. Stability-driven nonnegative matrix factorization to interpret spatial gene expression and build local gene networks. *Proceedings of the National Academy of Sciences*, 113(16):4290–4295, 2016, http://www.pnas.org/content/113/16/4290.full.pdf.

[147] Jiyan Yang, Oliver Rubel, Michael W Mahoney, and Benjamin P Bowen. Identifying important ions and positions in mass spectrometry imaging data using cur matrix decompositions. *Analytical chemistry*, 87(9):4658–4666, 2015.

[148] Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

[149] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562, 2001.

[150] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 226–231. AAAI Press, 1996.

[151] David Donoho and Victoria Stodden. When does non-negative matrix factorization give a correct decomposition into parts? In *Advances in neural information processing systems*, 2003.

[152] Rong Ge and James Zou. Intersecting faces: Non-negative matrix factorization with new guarantees. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 2295–2303, 2015.

[153] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[154] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[155] Chiranjib Bhattacharyya, Navin Goyal, Ravi Kannan, and Jagdeep Pani. Nonnegative matrix factorization under heavy noise. In *Proceedings of the 33nd International Conference on Machine Learning*, pages 1426–1434, 2016.

[156] Liam A McDonnell and Ron Heeren. Imaging mass spectrometry. *Mass spectrometry reviews*, 26(4):606–643, 2007.

[157] Oliver Rübel, Annette Greiner, Shreyas Cholia, Katherine Louie, E. Wes Bethel, Trent R. Northen, and Benjamin P. Bowen. OpenMSI: A high-performance web-based platform for mass spectrometry imaging. *Analytical Chemistry*, 85(21):10354–10361, 2013.

[158] A.M. Deml, R. O'Hayre, C. Wolverton, and V. Stevanović. Predicting density functional theory total energies and enthalpies of formation of metal-nonmetal compounds by linear regression. *Phys. Rev. B*, 93:085142, Feb 2016.

[159] R.L. Scott and R.L. Hildebrand. The Solubility of Nonelectrolytes. *New York: Reinhold Publ. Corp*, 1950.

[160] A.R. Miedema, F.R. de Boer, and P.F. de Chatel. Empirical description of the role of electronegativity in alloy formation. *J. Phys. F: Metal Phys.*, 3:1558–1576, 1973.

[161] Y. Ouyang, B. Zhang, S. Liao, Z. Jin, and H. Chen. Formation enthalpies for fcc metal based binary alloys by embedded atom method. *Trans. Nonferrous Met. Soc. China*, 8(1):60–63, 1998.

[162] F.R. De Boer, W.C.M. Mattens, R. Boom, A.R. Miedema, and A.K. Niessen. *Cohesion in metals: Transition Metal Alloys (Cohesion and Strucure)*. North-Holland, Amsterdam, 1988.

[163] J. Ihm, A. Zunger, and M.L. Cohen. Momentum-space formalism for the total energy of solids. *J. Phys. C*, 12, 4409, 1979. Erratum: J. Phys. C 13, 3095 (1980).

[164] M.T. Yin and M.L. Cohen. Microscopic Theory of the Phase Transformation and Lattice Dynamics of Si. *Phys. Rev. Lett.*, 45, 1004, 1980.

[165] A.D. Becker. Density-functional exchange-energy approximation with correct asymptotic behavior. *Phys. Rev. A*, 38, 3098, 1988.

[166] A.R. Oganov and C.W. Glass. Crystal structure prediction using ab initio evolutionary techniques: Principles and applications. *J. Chem. Phys.*, 124(24):244704, 2006.

[167] J. Lee, A. Seko, K. Shitara, and I. Tanaka. Prediction model of band-gap for ax binary compounds by combination of density functional theory calculations and machine learning techniques. 2015.

[168] G. Montavon, M. Rupp, V. Gobre, A. Vazquez-Mayagoitia, K. Hansen, A. Tkatchenko, K.-R. Müller, and O.A. von Lilienfeld. Machine learning of molecular electronic properties in chemical compound space. *New J. Phys.*, 15(9):095003, 2013.

[169] A.L. Teixeira, J.P. Leal, and A.O. Falcao. Random forests for feature selection in qspr models-an application for predicting standard enthalpy of formation of hydrocarbons. *J. Cheminform.*, 5(9), 2013.

[170] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[171] Y. Saad, D. Gao, T. Ngo, S. Bobbitt, J.R. Chelikowsky, and W. Andreoni. Data mining for materials: Computational experiments with AB compounds. *Phys. Rev. B*, 85(10):104104, 2012.

[172] Luca M Ghiringhelli, Jan Vybiral, Sergey V Levchenko, Claudia Draxl, and Matthias Scheffler. Big data of materials science: Critical role of the descriptor. *Physical review letters*, 114(10):105503, 2015.

[173] Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. `http://cvxr.com/cvx`, Mar. 2014.

[174] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer Science & Business Media, New York, 2013.

[175] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[176] A. Jain, S.P. Ong, G. Hautier, W. Chen, W.D. Richards, S. Dacek, S. Cholia, D. Gunter, D. Skinner, G. Ceder, and K.A. Persson. The Materials Project: A materials genome approach to accelerating materials innovation. *APL Materials*, 1(1):011002, 2013.