Low dimensional approximations: Problems and Algorithms

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

Thanh Trung Ngo

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Doctor of Philosophy

Yousef Saad

May, 2014

Acknowledgements

First, I wish to thank my advisor, Professor Yousef Saad, without whom this thesis would not have been possible. He has guided me throughout the years with great patience while giving me as much freedom as possible to pursue my interests. His great energy has always inspired me to work harder. One simply could not have asked for a better mentor.

I wish to thank Professor Rui Kuang as well for the opportunity to work with him during my first year and for his continuous encouragement over the years. I am also indebted to Professor Gilad Lerman in the School of Mathematics. Every discussion with him was always fascinating and widened my views.

My thanks also goes to my fellow labmates Ruipeng Li, Jie Chen, Haw-ren Fang, Daniel Osei-Kuffuor, Pierre Carrier, Jok Tang, Da Gao, Shashanka Ubaru, and Vasileios Kalantzis for their numerous interesting discussions and helpful suggestions. I particularly value the friendship of Ruipeng, Shashanka, and Mr. Kalantzis. Sinh Nguyen, my dear friend, deserves special thanks for always being available to help. I wish him the best of luck in his future endeavors.

Finally, my deepest gratitude goes to my family: my wife, Thanh Le, for sharing with me all the ups and downs in our journey and beyond, my brother, Trung Ngo, for his constant inquiry about when I would graduate, and my parents, Thuan Ngo and Thao Tran, for their unconditional love and constant support.

Dedication

To my family.

Abstract

High dimensional data usually have intrinsic low rank representations. These low rank representations not only reveal the hidden structure of the data but also reduce the computational cost of data analysis. Therefore, finding low dimensional approximations of the data is an essential task in many data mining applications.

Classical low dimensional approximations rely on two universal tools: the eigenvalue decomposition and the singular value decomposition. These two different but related decompositions are of high importance in a large number of areas in science and engineering. As a result, research in numerical linear algebra has been conducted to derive efficient algorithms for solving eigenvalue and singular value problems. Because available solvers for these problems are so well developed, they are often used as black boxes in data analysis.

This thesis explores numerical linear algebra techniques and extends well-known methods for low rank approximations to solve new problems in data analysis. Specifically, we carefully analyze the trace ratio optimization and argue that solving this problem can be done efficiently. We also propose efficient algorithms for low rank matrix approximations with missing entries. We also reformulate and analyze classical problems from a different perspective. This reveals the connection between the proposed methods and traditional methods in numerical linear algebra. The performance of the proposed algorithms is established both theoretically and through extensive experiments in dimension reduction and collaborative filtering.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation and contribution

In this thesis, we are interested in developing efficient numerical linear algebra techniques for low dimensional approximation problems in data analysis focusing on *dimension reduction* and *collaborative filtering*.

Classical low dimensional approximations rely on two universal tools: *the eigenvalue decomposition* and *the singular value decomposition*. These two different but related decompositions are of high importance in a large number of areas in science and engineering. As a result, research in numerical linear algebra has been conducted to derive efficient algorithms for solving eigenvalue and singular value problems. In the information age, with huge data being constantly generated, there is an increasing need for efficient methods to capture useful information from data and so low dimensional approximation has come into play in the field of data analysis.

High dimensional data usually have intrinsic low rank representations. Therefore, finding low dimensional approximations of the data is an essential task in many data mining applications. Low dimensional approximations not only reveal the hidden structure of the data but also reduce the computational cost of subsequent analysis.

A large field in data analysis involving low dimensional approximation is dimension reduction. Dimension reduction refers to a set of techniques that find mappings from high dimensional data to low dimensional data while capturing important characteristics

of the dataset. Given the data in the reduced space, classification and clustering techniques can be performed efficiently to analyze the data. The eigenvalue decomposition and the generalized eigenvalue decomposition have been applied to this task and yield methods such as Principal Componant Analysis (PCA) [2] and Linear Discriminant Analysis (LDA) [3, Chapter 5].

Not all problems can be directly cast as these classical eigenvalue problems. In fact, because eigensolvers have been so extensively developed, eigenvalue problems tend to be directly used to formulate problems in data analysis. The classical formulation of LDA based on the generalized eigenvalue problem is an example. Indeed, another formulation based on *trace ratio optimization* can be applied to a class of supervised dimension reduction techniques including LDA. The trace ratio optimization problem can be solved more efficiently than the generalized eigenvalue problem and it can also lead to an improved performance of data analysis tasks [4, 5, 6, 7, 8, 9, 10, 11].

An example of the singular value decomposition (SVD) being directly applied is in collaborative filtering for *recommender systems*. In this application, we have a matrix whose rows correspond to users and columns correspond to items. Each entry of the matrix is the rating of the user on the item. The matrix is only partly observed and we need to predict missing ratings. Thereafter, items are recommended to users based on the predicted ratings. Low rank approximations and the SVD have been applied very successfully to collaborative filtering. In [12], the missing entries are filled with average ratings and the SVD of this imputed matrix are used to obtain a low rank approximation of the partially observed matrix. The method outperforms previous approaches to recomender systems.

In recent years, much progress has been made in tackling the issue of missing entries. Matrix factorization algorithms have been devised for *low rank matrix approximation with missing entries* and very good results were obtained (see e.g. [13]). Although the methods are very efficient, it is hard to obtain performance guarantees because they are based on nonconvex optimization formulations. Convex optimization and probabilistic analysis have been employed to derive algorithms with performance guarantees [14, 15, 16, 17]. At the core of these algorithms is the SVD to compute approximate solutions. From the optimization point of view, the SVD itself is a very interesting problem because it is a nonconvex optimization problem whose global solution can be

computed efficiently. It is employed in each iteration of these algorithms and well developed solvers can be used as black boxes. Nevertheless, finding the SVD of a matrix requires an iterative process and it can be a good idea to investigate iterative techniques for SVD exploiting ideas from convex formulations of the problem to derive more efficient algorithms with provable convergence properties. Indeed, some nonconvex optimization algorithms with performance guarantees have been proposed [18, 19]. Similar guarantees for Alternating Least Squares, a widely used matrix factorization algorithm for low rank approximation with missing entries, are also obtained in [20, 21]. The relationship between these algorithms and subspace iteration, a classical and simple method for singular value decomposition and eigenvalue decomposition, was exploited to obtain the results. These have shown possibilities to extend numerical linear algebra techniques to efficiently solve new low rank approximation problems in data analysis.

This thesis contributes to the literature along this direction. Numerical linear algebra techniques are explored to develop efficient algorithms for matrix approximation with missing entries and trace ratio optimization. Reformulations and analyses of classical problems from a different angle are discussed, revealing the connection between the proposed methods and traditional methods in numerical linear algebra. The performance of the algorithms is established both theoretically and through extensive experiments.

1.2 Summary of thesis

In Chapter 2, we will discuss low rank approximation problems and techniques in data analysis. Dimension reduction methods are examined from a matrix perturbation and numerical linear algebra point of view. Collaborative filtering and the related technique, Latent Semantic Indexing, are discussed. At the end of the chapter, we will describe our experiments of using dimension reduction techniques in an emgerging field, material informatics, to study the properties of chemical compounds.

In Chapter 3, we will discuss computational methods for low rank approximation. The two classical techniques, subspace iteration and Krylov subspace methods, are described. These methods are fundamental to algorithms proposed in later chapters. Next, an optimization perspective of low rank approximation is presented. The classical SVD

is reformulated as an optimization problem and algorithms to solve this optimization are shown to be essentially the subspace iteration method. This is the bridge between traditional algorithms and algorithms for new low rank approximation problems. From this viewpoint, nuclear norm minimization and its nonconvex formulation are also discussed. The stationary points of the nonconvex objective function are investigated. The analysis sheds some light on why global convergence can usually be obtained. After this, the optimization framework on matrix manifolds is presented. This framework provides geometric insights into well-known algorithms and helps develop other efficient methods in numerical linear algebra. Finally, randomized matrix approximation is briefly introduced to provide a different view of the low rank matrix approximation problem with missing entries.

In Chapter 4, we present our proposed algorithms for this problem. We start by an adaptation of the subspace iteration for incomplete matrices. Each iteration of the subspace iteration is inexpensive and the algorithm converges very rapidly. A connection to the optimization algorithms on the Grassmann manifold is derived and improvements to the subspace iteration are obtained. Then, based on a similar connection between the subspace iteration and optimization algorithms for low rank approximation, we propose ways to improve the convergence of a widely used gradient descent method for matrix completion. Theoretical analysis and experimental results are provided to demonstrate the efficiency of the methods.

In Chapter 5, the trace ratio optimization problem is analyzed in depth. Based on the analysis, an efficient algorithm based on Newton iteration and Lanczos algorithm is presented. Extensive experiments show the advantage of the trace ratio optimization problem over the generalized eigenvalue problem for dimension reduction and confirm observations obtained in previous work. Some extensive analyses of the trace ratio optimization problem are left to Appendix A to preserve the clarity of the chapter.

Chapter 2

Low rank approximations in data analysis

In this chapter, we describe some low rank approximation problems in data analysis. First, we will talk about dimension reduction techniques from classical methods such as Principal Component Analysis (PCA) and then discuss recent graph-based methods for nonlinear modeling. Low rank approximations in information retrieval are also discussed. Most of the problems can be solved using the classical eigenvalue decomposition and singular value decomposition solvers as black boxes. As newer problems appear, one needs to explore these methods more deeply to find new algorithms. We will briefly introduce one such problem, namely the matrix completion problem, and its application in collaborative filtering. Finally, a case study in an emerging field, material informatics, is presented with some promising results.

2.1 Principal Component Analysis and two views of low rank approximation

We begin by depicting two different views of low dimensional approximation, namely, *data denoising* and *data compression*, through Principal Component Analysis (PCA) [2], the most classical dimension reduction method in data analysis. We will also explain in this section how matrix perturbation theory and eigenvalue problems play roles in

formulating PCA.

In many applications, the data points belong to a low dimensional manifold (linear or nonlinear) but they are observed through measurements in a much higher dimensional space called the ambient space. With the unavoidable noise occurring in the measuring process, the observed data points do not exactly lie on the low dimensional structure. Nevertheless, under some assumptions on the noise, the manifold can be recovered with considerably small error. This is the *denoising* view of low dimensional approximation.

Suppose we have n data points, $a_1, a_2, \ldots, a_n$, each of which is a vector in $\mathbb{R}^m$. In this section, we assume that these points are located on a linear manifold of dimension $r \ll m, n$. Denote by A the data matrix in which each column corresponds to a data point:

$$A = [a_1, \ldots, a_n] \in \mathbb{R}^{m \times n}.$$

Consider the case when the linear manifold passes through the origin. In other words, it corresponds to a linear subspace of dimension r in $\mathbb{R}^m$. Let us call the data matrix in this case A_*. Clearly, A_* is a rank r matrix. The thin Singular Value Decomposition (SVD) of A_* is in the form given below:

$$A_* = U_* \Sigma_* V_*^T \text{ where } U_* \in \mathbb{R}^{m \times r}, V_* \in \mathbb{R}^{n \times r} \text{ and } \Sigma_* \in \mathbb{R}^{r \times r}.$$

The columns of U_* form orthonormal bases of the linear subspace containing the data points. This means that the manifold has been exactly recovered by the thin SVD of A_*.

If the data is measured with additive noise, the data matrix A is perturbed from A_* by a noise matrix E:

$$A = A_* + E. \tag{2.1}$$

For most random noise models, A does not have rank r almost surely. If the magnitude of the noise is reasonably small, we can still recover the subspace quite accurately by computing the top-r singular vectors of the perturbed matrix.

The theorem below, by Hermann Weyl, establishes a bound on the perturbation of the singular values of the perturbed matrix A with respect to A_* [22, Corollary 4.10].

Theorem 2.1 (Weyl's inequality) *Let $p = \min(m, n)$ and assume that the singular values of A and A_* are sorted in decreasing order: $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_p$ and $\sigma_1^* \geq \sigma_2^* \geq \ldots \sigma_p^*$*

respectively. Then,

$$|\sigma_i - \sigma_i^*| \leq \|E\|, \quad \forall i \in \{1, 2, \ldots, p\}.$$

Let $A = U\Sigma V^T$ be the SVD of A ($U \in \mathbb{R}^{m\times m}, \Sigma \in \mathbb{R}^{m\times n}, V \in \mathbb{R}^{n\times n}$). The r-truncated SVD of A is

$$A_r = U_r \Sigma_r V_r^T, \tag{2.2}$$

where $U_r \in \mathbb{R}^{m\times r}$ and $V_r \in \mathbb{R}^{n\times r}$ contain the r columns of U and V corresponding to the r largest singular values, and $\Sigma_r \in \mathbb{R}^{r\times r}$ is the diagonal matrix whose diagonal entries are the associated singular values. Denote by $P_{U_r}^{\perp} = I - U_r U_r^T$ the orthogonal projection onto the orthogonal complement of span (U_r). The sine of the principal angle between the subspaces span (U_r) and span (U_*) is defined as follows:

$$\sin\Theta(U_r, U_*) \triangleq \|P_{U_r}^{\perp} U_*\|_2. \tag{2.3}$$

A perturbation bound of the singular subspaces is presented in the following theorem by Davis-Kahan and Wedin [23, 24]:

Theorem 2.2 (Wedin's bound)

$$\sin\Theta(U_r, U_*) \leq \frac{\|E\|_2}{\sigma_r - \sigma_{r+1}}.$$

If $\|E\|$ is small enough compared to σ_r^*, $\|E\|_F \ll \sigma_r^*$, then combining Weyl's inequality and Wedin's bound we have:

$$\sin\Theta(U_r, U_*) \leq \frac{\|E\|^2}{\sigma_r^*}.$$

The smaller the signal to noise ratio $\|E\|^2/\sigma_r^*$ is, the closer span (U_r) is to span (U_*). We illustrate this phenomenon in Figure 2.1. In both plots, there are 50 points lying on a line and they are perturbed with the same magnitude of noise. The width of the data cluster on the left is shorter than that of the data cluster on the right. This translates to σ_1^* of the data matrix on the left being smaller than that of the data matrix on the right. The black line is the unperturbed manifold and the red line is the one computed from the top singular vector. We can clearly see that the lines on the right are almost identical while there is a slight perturbation between the two on the left.

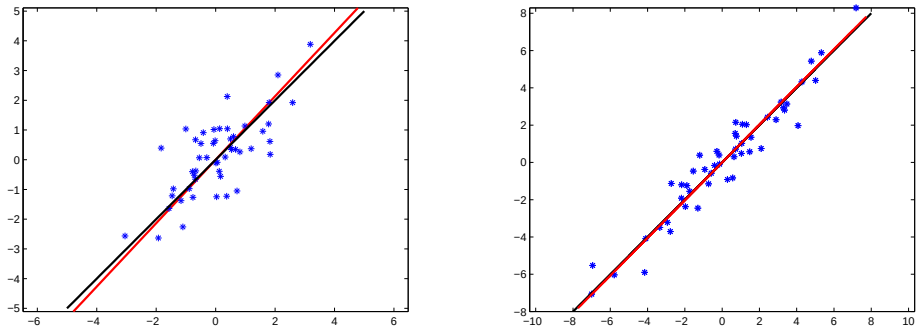


Figure 2.1: 1-D approximation by the first singular vector with different magnitudes of the largest singular value.

From another point of view, consider a general matrix A, which can be of full rank, meaning that all of its singular values are nonzero. Recalling the r-truncated SVD of A in (2.2), the Eckart-Young theorem below provides the optimality characteristic of the truncated SVD.

Theorem 2.3 (Eckart-Young theorem)

$$A_r = \underset{X\in\mathbb{R}^{m\times n},\text{rank}\,(X)=r}{\text{argmin}} \|X - A\|_F = \underset{X\in\mathbb{R}^{m\times n},\ \text{rank}\,(X)=r}{\text{argmin}} \|X - A\|_2$$

The theorem states that A_r is the best rank-r approximation of A in both the Frobenius norm and the spectral norm. Note that a rank-r matrix has much fewer degrees of freedom than a general $m \times n$ matrix. As a result, it also needs less memory to be stored and computations related to a low rank matrix are usually far more efficient. This means A_r is *a compressed version* (or *a concise version*) of A, and A_r is optimal in the sense that its Frobenius distance to A is the smallest among matrices of the same rank.

Now, we relax the assumption that data points lie on linear subspace and let the points be on an affine plane of dimension r. If one can translate the data in a way that the affine plane passes through the origin, we can use the above results to recover the linear manifold. The matrix perturbation in equation 2.1 corresponds to the following additive noise on each data point a_i^*:

$$a_i = a_i^* + \epsilon_i.$$

where ϵ_i is the noise vector. Let $\bar{a}$ be the centroid of the observed data points:

$$\bar{a} = \frac{1}{n}\sum_{i=1}^{n} a_i.$$

If we assume that the noise vectors are independently identically distributed (i.i.d) random vectors with mean zero, i.e. $\mathrm{E}(\epsilon_i) = 0$, then,

$$\mathrm{E}(\bar{a}) = \frac{1}{n}\sum_{i=1}^{n} a_i^* \triangleq \bar{a}^*.$$

Hence, $\mathrm{E}(\bar{a})$ is a point lying on the affine plane. The concentration bound [25] assures that $\bar{a}$ deviates very little from its expectation, $\bar{a}^*$. This means that the empirical mean $\bar{a}$ of the data points is a point which is very close to the original affine plane. A translation of everything by $\bar{a}$ is what we need to obtain a linear subspace.

Let $\bar{A} = [a_1 - \bar{a}, a_2 - \bar{a}, \ldots, a_n - \bar{a}]$ be the centered (translated) data matrix and $U_r\Sigma_r V_r^T$ be its truncated SVD where $U_r \in \mathbb{R}^{m\times r}$, $V_r \in \mathbb{R}^{n\times r}$ and $\Sigma_r \in \mathbb{R}^{r\times r}$. Define the covariance matrix as follows:

$$C = (A - \bar{A})(A - \bar{A})^T.$$

The min-max characteristic (also called the Courant–Fisher characteristic) of U_r tells us that U_r is the solution to the following optimization problem (see e.g. [26, 27]):

$$\max_{U\in\mathcal{U}_r} \mathrm{Tr}\,[U^T C U], \tag{2.4}$$

where $\mathcal{U}_r$ is the set of orthogonal $\mathbb{R}^{n\times r}$ matrices. This matrix U_r is the projection found by *Principal Component Analysis* (PCA) to reduce the dimensionality of a dataset. The quantity $\mathrm{Tr}\,(U_r^T C U_r)$ is the total variance of the dataset after a projection onto $\mathrm{span}\,(U_r)$. From a statistical point of view, PCA finds the projection which maximizes the variance of the dataset.

We will see that this trace optimization appears in many other dimension reduction techniques. In many cases, eigensolvers can be used as black boxes. However, in other cases, we have to investigate eigensolvers in order to obtain efficient methods.

Modern treatments of PCA include analyzing PCA with random perturbation. The perturbation can be abrupt because of privacy settings or missing entries [14, 28]. An

application in collaborative filtering is discussed in Section 2.5. The bound in Theorem 2.2 is optimal for general perturbation matrix E. Under many common models for the random noise, stronger results which hold with high probability, can be obtained. The readers are referred to [29, 30] for such results.

2.2 Linear Discriminant Analysis

Like PCA, Linear Discriminant Analysis (LDA) is arguably one of the most popular supervised dimension reduction techniques. The data points are now labeled by different classes. In this case, the assumption is that in some low dimensional subspace, the classes can be well separated. That is, the data cloud itself may have high dimension, but a low dimensional representation is enough to separate one class from the others.

Figure 2.2 shows a simple example. The 2-D points are not contained in a 1-D linear subspace, but the 1-D subspace represented by the black solid line is sufficient to separate the points of different classes which are represented by different shapes.

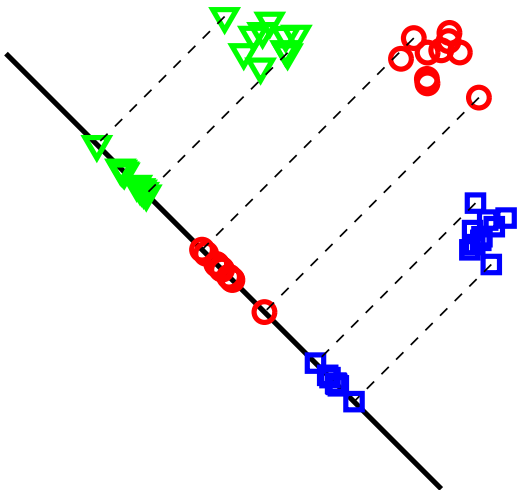


Figure 2.2: Points in 2-D plane of 3 different classes. The black solid line is the 1-D subspace in which different classes are well-separated.

There are obviously infinitely many ways to project the points as depicted in Figure

2.3. The line L_1 is a good 1-D projection such that it is easy to separate squares and circles. The line L_2, in contrast, is not a good projection for classification purpose since it mixes up points of different shapes. Intuitively, a good projection is the one that makes points of the same shape close to each other while keeping points of different shapes far away from the other.

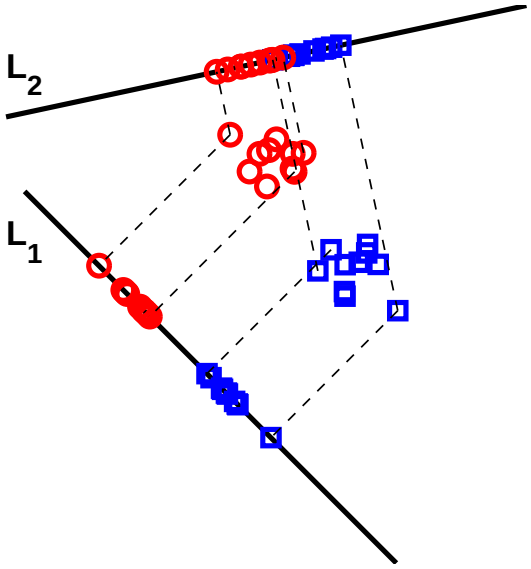


Figure 2.3: Points on a 2-D plane. Projections on the line L_1 have a clear separation between squares and circles. Projections on the line L_2 have mixed shapes in the middle.

LDA does this by finding the projection that maximizes the ratio of between-class scatter and within-class scatter where between-class scatter is the distance between the projected centroid of the squares and the projected centroid of the circles and within-class scatter is the sum of the variance of the projected squares and the variance of the projected circles.

Let us first formally describe how to find a subspace in which *the between-class scatter is well-preserved*. Consider c classes and assume that the classes can be well-represented by their centroids $\mu_1, \mu_2, \ldots, \mu_c$. For example, the assumption is true when the data points in each class are normally distributed around its mean and the distances between the classes are bigger than the variance of each class.

PCA can be used to find such subspace with the data matrix $[\mu_1, \mu_2, \ldots, \mu_c]$. Let $\bar{\mu}$

be the mean of the class centroids.

$$\bar{\mu} = \frac{1}{c}\sum_{i=1}^{c}\mu_i.$$

Note that if the number of data points in every class is roughly the same, $\tilde{\mu}$ is very closed to μ, the centroid of all data points:

$$\mu = \frac{1}{m}\sum_{i=1}^{m}a_i.$$

PCA would find the matrix $U_r \in \mathbb{R}^{m\times r}$ which is the solution to the problem:

$$\underset{U\in\mathcal{U}_r}{\text{maximize}}\ \text{Tr}\,[U^T S_B U] \quad \text{where} \quad S_B = \sum_{i=1}^{c}(\mu_i-\mu)(\mu_i-\mu)^T. \tag{2.5}$$

Now, we switch our focus to within each class. The subspace that we are looking for is the one that makes the points in each class tightly concentrated around the centroid. This is opposite to PCA and can be characterized by the optimization problem below:

$$\underset{U\in\mathcal{U}_r}{\text{minimize}}\ \text{Tr}\,[U^T S_{W_i} U] \quad \text{where} \quad S_{W_i} = \sum_{a_j\in \text{ class } i}(a_j-\mu_i)(a_j-\mu_i)^T.$$

Because the trace is a linear function, summing over the classes, the problem becomes:

$$\underset{U\in\mathcal{U}_r}{\text{minimize}}\ \text{Tr}\,[U^T S_W U] \quad \text{where} \quad S_W = \sum_i S_{W_i}. \tag{2.6}$$

If U is a single vector u, i.e. it is a 1-D projection, we can simultaneously achieve the goals in (2.5) and (2.6) by maximizing the Rayleigh quotient:

$$\frac{u^T S_B u}{u^T S_W u}.$$

The solution to this problem is well-known and it is the eigenvector corresponding to the largest eigenvalue of the generalized eigenproblem:

$$S_B u = \lambda S_W u. \tag{2.7}$$

There is no unique way to generalize LDA to the multidimensional case. The natural way, in our opinion, is to find $U \in \mathcal{U}_r$ which maximizes:

$$\frac{\text{Tr}\,[U^T S_B U]}{\text{Tr}\,[U^T S_W U]}. \tag{2.8}$$

This problem has been considered hard to solve and is often overlooked. We will analyze this problem carefully in Chapter 5.

The traditional way to formulate LDA with multidimensional projection is to take the top-r eigenvectors of the generalized eigenproblem (2.7). It is known that the top-r generalized eigenvectors are the solution to the following optimization problem:

$$\max_{U^T S_W U = I} \quad \mathrm{Tr}\,[U^T S_B U]. \tag{2.9}$$

Note that the constraint means that U is S_W-orthogonal. If S_W is close to λI where λ is a scalar, the solutions to Problem 2.8 and Problem 2.9 coincide. One particular case in which this happens is when the data points within each class are of the same dimension and they are isometrically normally distributed around their means, i.e. the covariance matrices are $\lambda_i I$ ($\lambda_i \in \mathbb{R}$, $i \in \{1 \dots c\}$).

LDA has been applied successfully in several applications such as image recognition. Each image is a vector of very high dimension. The curse of dimensionality makes it hard to derive robust models to classify images into different classes. LDA and other dimension reduction techniques can be used to project data points onto a low dimensional space where many classification methods can be efficiently applied to derive good models to classify data.

2.3 Manifold learning and graph-based dimension reduction

The methods we have discussed so far are in the linear domain. These linear methods can be regarded as nonlinear methods when they are applied to the space of nonlinear polynomials or functions of the original data. These techniques correspond to the kernel methods whose goal is to transform a linear problem into a nonlinear one.

In this section, we discuss another approach to low dimensional approximation of nonlinear problems to model data lying on a smooth nonlinear manifold embedded within the measured space. The approach is based on building graphs from local information among data points. The graph provides an approximation of the smooth manifold. Having the graph representation of the manifold, we can use linear algebra methods to approximate it. The two views of low dimensional approximation discussed

in Section 2.1 can also be readily seen in this context for manifolds and graphs. These methods have roots in spectral graph theory [31] and the relationship between a smooth manifold and its graph-based approximation [32].

Central to these methods is the graph Laplacian matrix which is defined as follows. Consider a weighted undirected graph G and let W be the adjacency matrix of the graph. Each vertex of G corresponds to a data point. In its simplest form, the entries of W are:

$$w_{ij} = \begin{cases} 1 & \text{if there is an edge connecting vertex } i \text{ and vertex } j, \\ 0 & \text{otherwise.} \end{cases}$$

The graph Laplacian is:

$$L \triangleq D - W, \tag{2.10}$$

where D is the diagonal matrix whose entries are

$$D_{ii} = \sum_{j=1}^{n} w_{ij}.$$

To understand the intuition, let us begin with a simple example in Figure 2.4. We want to embed the points sampled from the 1-dimensional manifold (the spiral curve) into the real line $\mathbb{R}$. Assume that we can label the sampled points from a_1 to a_m ($m = 21$ in this case) in order from one end-point to another end-point of the curve. Let us have a graph connecting neighbor points as depicted in the figure.

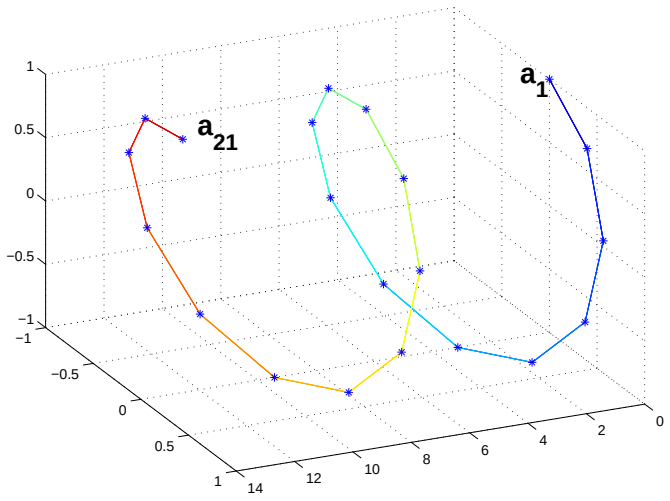


Figure 2.4: A graph connecting points sampled from the 1-D manifold, labeling from a_1 to a_{21}.

In this case, the adjacency matrix of the graph is the following 21×21 tridiagonal matrix:

$$W = \begin{bmatrix} 1 & 1 & 0 & \dots & \dots & 0 \\ 1 & 1 & 1 & & & \\ & 1 & \ddots & & & \\ & & & \ddots & 1 & \\ & & \dots & 1 & 1 & 1 \\ 0 & \dots & \dots & 0 & 1 & 1 \end{bmatrix}.$$

The Laplacian can be readily obtained. It is well-known that the smallest eigenvalue of the Laplacian matrix is 0 and the associated eigenvector is $\mathbf{1}/\|\mathbf{1}\|$. An interesting result from spectral graph theory is that the second smallest eigenvector provides a good embedding of the graph into a 1-dimensional space. In this case, the second eigenvector is:

$$\begin{bmatrix} 0.3077 & 0.3009 & 0.2873 & 0.2673 & 0.2413 & 0.2099 & 0.1738 \\ 0.1339 & 0.0910 & 0.0460 & 0.0000 & -0.0460 & -0.0910 & -0.1339 \\ -0.1738 & -0.2099 & -0.2413 & -0.2673 & -0.2873 & -0.3009 & -0.3077 \end{bmatrix}^T.$$

Each component of the vector is used as the coordinate of the embedded point in 1-D. Figure 2.5 shows the embedded points in 1-D. We can see that the points are perfectly embedded into the 1-D space in the sense that their order on the curve is perfectly maintained.

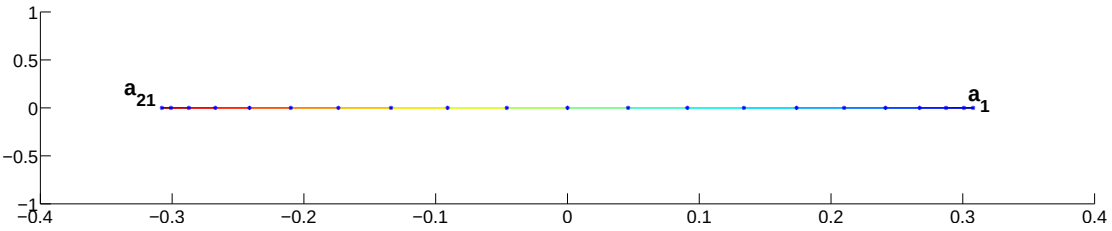


Figure 2.5: 1-D embedding of the curve in Figure 2.4. The embedded points $a_1, \ldots, a_{21}$ are positioned in order from right to left.

This simple example shows that the low dimensional structure of the manifold is revealed by the low rank approximation of the graph Laplacian matrix. Note again that the approximation is in the Frobenius sense. In practice, the data always have noise and it is not trivial to obtain a graph that describes the shape of the manifold faithfully.

Taking a closer look at the plot in Figure 2.5, we can see that although the order of the points is perfectly preserved, the distances between them get smaller at both ends. In some applications, one may need the pairwise distances and/or other criteria to be well-preserved. Embedding into spaces of dimension higher than one is also desired. This gives rise to various methods [33, 34] where one can construct the graphs and the edge weights differently or optimize a different objective function.

The most common method to construct the graph is to use an undirected graph based on the k-nearest neighbor (kNN) graph or the ϵ-graph. With kNN, each point is connected to its k nearest neighbors based on some distance measure, e.g. the Euclidean distance. For ϵ-graph, each point is connected to neighbor points within a ball centered at the point with radius ϵ. We will assume that *the graph is connected*.

Edge weights usually reflect the similarities between data points. One common method used in many manifold learning applications is to use the Gaussian kernel:

$$w_{ij} = e^{-\alpha \mathrm{d}(a_i, a_j)^2}, \tag{2.11}$$

where $\mathrm{d}(a_i, a_j)$ is some distance measure between a_i and a_j, and α is a parameter which

controls the decay of the similarity as the distance increases. Laplacian Eigenmaps (LE) [33] is a well-known method using this construction. If the desired embedded dimension is r, let $Y \in \mathbb{R}^{n \times r}$ be the matrix containing r smallest eigenvectors of L (except the smallest one), LE uses each row of Y as the coordinates for the r dimensional embedding of each data point.

A different view of Laplacian Eigenmaps lies in the optimization of eigenvalue problems. The problem is to find Y that solves the problem:

$$\text{minimize } \mathrm{Tr}\,(Y^T L Y) \text{ subject to } Y^T \mathbf{1} = 0 \text{ and } Y^T Y = I.$$

Let $y_i = (Y^{(i)})^T \in \mathbb{R}^r$, where $(Y^{(i)})$ is the i-th row of Y. y_i is the image of a_i in the embedded space. The problem is equivalent to:

$$\text{minimize } \sum_{ij} w_{ij} \|y_i - y_j\|^2 \text{ subject to } y_i^T 1 = 0 \ \forall i. \tag{2.12}$$

This optimization means that we want points that are close in the original space to be also close in the projected space. To better preserve the relative distances between data points, one of the approaches is to use the normalized Laplacian matrix:

$$\tilde{L} = I - D^{-1} W.$$

Pre-multiplying $\tilde{L}$ with D, we can see that computing the eigenvalue of this matrix is equivalent to the following generalized eigenvalue problem:

$$Ly = \lambda D y.$$

This is in turn equivalent to the following optimization problem:

$$\text{minimize } \mathrm{Tr}\,(Y^T L Y) \text{ subject to } Y^T D Y = I. \tag{2.13}$$

Another well-known method is Locally Linear Embedding (LLE) [34]. As the name of the method indicates, LLE assumes that the manifold is locally linear, meaning each point can be approximated by a linear combination of neighbor points:

$$a_i \approx \sum_{j, a_j \in N_i} w_{ij} a_j$$

where $\sum_{j \neq i} w_{ij} = 1$ and $w_{ij} = 0$ if $a_j \notin N_i$. The weights of the matrix $W = [w_{ij}]$ can be calculated from the set of equations above. Instead of using the Laplacian matrix L, LLE uses the Gram matrix defined below:

$$G = (I - W^T)(I - W).$$

Similarly, the optimization of LLE is:

$$\text{minimize } \text{Tr}\,[Y(I - W^T)(I - W)Y^T] \text{ subject to } Y^T\mathbf{1} = 0 \text{ and } Y^TY = I. \tag{2.14}$$

The objective function can be re-formulated as follows:

$$\begin{aligned} \text{Tr}\,[Y(I - W^T)(I - W)Y^T] &= \|Y - YW^T\|_F^2 \\ &= \sum_i \left| y_i - \sum_j w_{ij} y_j \right|^2 \end{aligned}$$

This shows that LLE encourages the embedded points to obey roughly the same local linear combinations as those in the original space.

A prominent problem with these methods is that if the dataset changes, the eigenvectors need to be recomputed because the mapping is nonlinear and it is not an explicit function. Several methods have been proposed to efficiently update the eigenvectors (see e.g. [35]).

A more computationally efficient approach is to make the mapping linear and explicitly represented by a set of basis vectors V, i.e. $Y = V^TA$. Replace $Y = V^TA$ in (2.13), we obtain the below minimization:

$$\underset{V}{\text{minimize}}\ \text{Tr}\,[V^TALA^TV] \text{ subject to } V^TADA^TV = I \tag{2.15}$$

If we use the Gram matrix as in LLE, we have:

$$\underset{V}{\text{minimize}}\ \text{Tr}\,[V^TA(I - W^T)(I - W)A^TV] \text{ subject to } V^TADA^TV = I \tag{2.16}$$

These methods correspond to Locality Preserving Projection (LPP) [36] and Neighborhood Preserving Projection (NPP) [37]. Orthogonal projections based on these methods and a unified framework based on trace optimization are proposed in [37].

An interesting observation is that PCA can be seen as a graph-based method using a fully connected graph. As a result, *PCA can be regarded as a global method while*

graph-based methods are local methods. Naturally, one can ask whether local versions of LDA can be formulated.

Indeed, a class of supervised dimension reduction algorithms based on graphs can be derived similarly [38, 39]. The central question is how to build graphs with the availability of class labels. One approach is to utilize two graphs: (1) within-class graph where edge (a_i, a_j) exists if a_i and a_j belong to the same class; and (2) between-class graph where edge (a_i, a_j) exists if a_i and a_j belong to different classes. They are also called class graph and repulsion graph in the literature [38]. Figure 2.6 illustrates how these two graphs are extracted from a k-nearest neighbor graph.

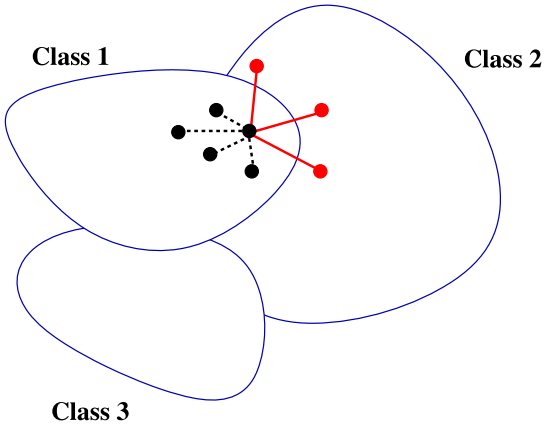


Figure 2.6: Illustration of the class graph and repulsion graph extracted from a kNN graph ($k = 7$). The class graph is illustrated in black, the repulsion graph is illustrated in red.

The key idea of these methods is that the class graph will force points that are in the same class to be close to each other in the embedded space, while the repulsion graph will create repulsion forces between nearby points which are not from the same class and will push them far away from each other in the embedded space.

Let L and $L^{(r)}$ be the Laplacian matrices of the class graph and the repulsion graph. Similar to the traditional way to formulate LDA, one may want to solve the optimization problem:

$$\underset{YLY^T=I}{\text{maximize}} \quad \text{Tr}\,[YL^{(r)}Y^T], \tag{2.17}$$

If a linear mapping is desired, this becomes:

$$\underset{V^T A L A^T V = I}{\text{maximize}} \quad \text{Tr}\,[V^T A L^{(r)} A^T V], \tag{2.18}$$

Again, the trace ratio optimization problem in (2.8) can be used. We will discuss this in more detail in Chapter 5.

2.4 Latent Semantic Indexing

Latent Semantic Indexing [40] is a well-established technique for information retrieval. It is extensively used in text mining for retrieving documents which are relevant to a query. It is also used in other context such as image retrieval [41] and video retrieval [42].

The method is equivalent to PCA except that the data points are assumed to be in a linear subspace and thus centering at zero is not needed. We resort to the classical text mining case to describe the technique. Suppose we have a set of m documents and n terms in the dictionary and a *document-term* matrix of size $m \times n$ where each element is essentially the frequency of the term occurring in the document. There are many different weighting schemes to compute the value of each entry of the document-term matrix. Readers are referred to [43] for an extensive discussion.

In terms of the data matrix A discussed in the previous section, each document is represented in this context as a n dimensional vector which describes the distribution of the words appearing in the document. In the same vein, each item is represented by a m-dimensional vector which reflects how often is it used in each document. Obviously, this representation may lead to a huge dimension and it might contain redundancy. For instance, many words can be highly correlated and tend to appear together in certain documents. Besides, the representation also reveals very little about the semantics of the documents which is what the users are most interested in when searching for documents using a query containing a few keywords.

Instead, let us assume that each document, d_i, can be represented by a small number of latent factors, say, r factors. That is $d_i \in \mathbb{R}^r$ where each element represents the degree of association between this document and the latent factor. We will see in some examples later that these factors, albeit being constructed from a pure computational viewpoint,

reveal some semantic meanings of the documents. Also, assume that each word can also be represented in terms of these r latent factors, $t_j \in \mathbb{R}^r$.

Now, we make a crucial assumption on the document-term matrix. Each entry of the matrix, which is the frequency of occurences of the term in the document, depends on how similar the document and the term are in this latent space. That is:

$$A_{ij} = \langle d_i, t_j \rangle.$$

Let $D = [d_1, d_2, \ldots, d_m]^T \in \mathbb{R}^{m \times r}$ and $T = [t_1, t_2, \ldots, t_n]^T \in \mathbb{R}^{n \times r}$. We have: $A = DT^T$. This means that the matrix A has rank r. Similar to what we discussed in Section 2.1, we use the truncated SVD of A to approximate it by a rank r matrix:

$$A \approx A_r = U_r \Sigma_r V_r^T$$

In practice, only the thin matrix factors U_r, Σ_r, V_r are stored. Given a query $q = [q_1, q_2, \ldots, q_n]$ which is a vector of the keywords in the query, the similarities of the documents in the collection to the query can be computed as:

$$s = A_r q = (U_r \Sigma_r)(V_r^T q).$$

This can be computed efficiently. Having s, one can pick some documents corresponding to the largest entries in s and return them to the user.

Given the huge size of the matrix and the constant stream of new documents in applications such as web search, the need for fast updating of the SVD of the document-term matrix is essential. This problem has been addressed in [44] and still remains active [45, 46].

2.5 Collaborative filtering using low rank approximation

Collaborative filtering, simply speaking, consists of collecting data about past interests of users to predict their future interests. Collaborative filtering has been implemented into several recommender systems at large e-commerce sites such as Amazon and Netflix to provide their users with useful suggestions.

The performances of recommender systems are so vital to e-commerce sites that Netflix announced its Grand Prize of 1 million dollars a few years ago for the first

participant to achieve 10% improvement to its own recommender system, Cinematch. Several teams have participated in the competition and many of them merged with each other to obtain better scores. The final results have revealed that the winning solution and many other top solutions are ensembles of many methods that boost up the results of single ones. Among the methods, low rank matrix approximation is widely used and provides superior results to other methods.

The assumptions are similar to those we discussed in the previous section. In this context, the document-term matrix is replaced by the user-item matrix, A, which consists of scores reflecting how the users like the items. Each user and each item are assumed to be represented by r dimensional vectors, u_i and v_j respectively. The user-item score is assumed to be:

$$A_{ij} = \langle u_i, v_j \rangle.$$

Similar to LSI, the matrix A has a certain rank r. The main difference to LSI is that we can only obtain some user-item scores from the rating system of the e-commerce website. This results in a partially observed matrix A. Accordingly, we call this problem *low rank approximation with missing entries*. It is usually referred to in the literature as *the matrix completion problem*. This partial observation is the chief challenge because classical methods for eigenproblem cannot be directly applied.

Since the end of the Netflix prize, much work has been devoted to analyze the problem and generalize it in many ways. A wide line of research involves using very powerful tools in convex analysis to formulate this problem as a convex problem using the nuclear norm relaxation of the rank:

$$\underset{X}{\text{minimize}}\ f_A(X) + \lambda \|X\|_*, \tag{2.19}$$

where $f_A(X)$ is a loss function representing how far X is from A, $\lambda > 0$ and $\|X\|_*$ is the nuclear norm of X:

$$\|X\|_* = \sum_{i=1}^{\min(m,n)} \sigma_i.$$

With this formulation, we do not know the rank beforehand. When λ gets larger, the solution to this problem tends to have lower rank.

As is always the case in data analysis, the data can be contaminated with noise. Under some assumptions on the matrix A and the noise model, it is proven that with

the right choice of λ, the matrix A can be recovered up to bounded error. If there is no noise, A can be exactly recovered. All of these recovery guarantees hold with high probability for random sets of observed entries.

Nuclear norm minimization for matrix completion can be reformulated as a semidefinite program (SDP) [47] and SDP solvers can be used to solve it. There are also methods which generalize the ℓ_1 minimization for vector problems to the matrix case and they are considerably faster than SDP solvers. However, these methods need to perform the singular value decomposition iteratively which makes them quite slow.

Another line of work is based on a non-convex formulation by restricting the search for the solution in the set of rank r matrices. There are some recovery guarantees with this approach [18, 48, 21] and efficient methods are being devised [18, 49, 50, 51, 52].

In chapter 4, we will present one such method which is practically efficient and has some convergence guarantees.

2.6 An example in material informatics

In this section, we describe some applications of low dimension approximation for discovery in materials science. This is one of the initial steps in the emerging field of material informatics.

Recent years have witnessed tremendous advances in bioinformatics by using computational tools to study biological structures such as genes and proteins. Although not being as well developed, material informatics is gaining more attention as a result of large databases of materials and chemical compounds being developed. At the same time, data analysis algorithms have been applied to study these data and they have shown some successful results [53]. We will now briefly sketch our work with dimension reduction methods to study binary compounds based on their constituent atoms.

We consider binary octet crystals whose composition is $A^N B^{8-N}$, where N refers to the number of valence electron. This family of crystals includes technologically important semiconductors such as Si, Ge, GaAs, GaN, and ZnO. There are approximately 80 members of this crystal family, which condense primarily in graphite, diamond (D), zincblende (Z), wurtzite (W), rocksalt (R), and cesium chloride structures.

The separation of these structures into distinct classes is difficult and has existed as

a problem in the literature for over 50 years [54]. Figure 2.7 illustrates one of the most successful structural maps for this family [1]. The separation between structural types is nearly exact. The 2D mapping in this example was performed by a judicious change of coordinates, exploiting physical intuition.

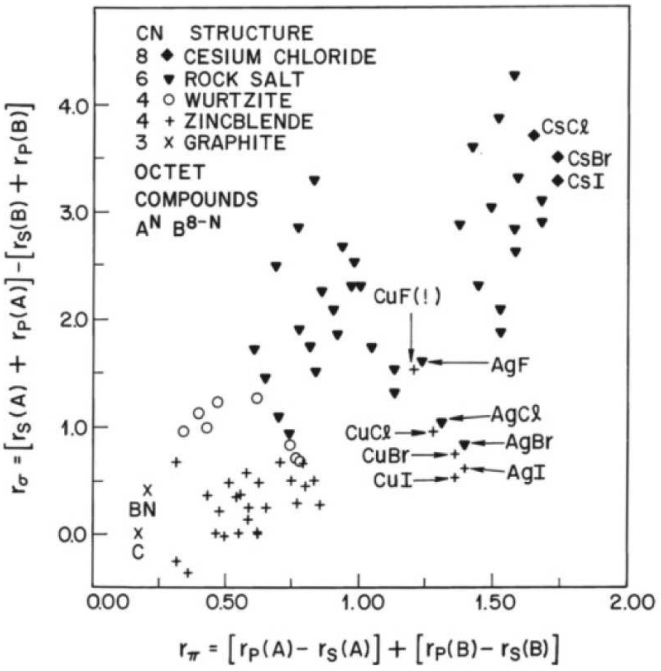


Figure 2.7: Structural map for binary octet crystals. The coordination number (CN) is indicated for each structural grouping. The chemical coordinates (r_σ, r_π) are combinations of orbital radii. This mapping of these compounds in two dimensions with the particular coordinates reveals a good clustering of the structures. Figure courtesy [1].

One question that may be asked is whether or not a similar mapping can be discovered in some systematic way. If we restrict the mapping to be linear, then the answer depends on what "features" are included in the data. In our experiment, we use the following information from each of the two constituent atoms:

- the number of valence electrons;
- the ionization energies of the s and p states of the ion core; and
- the radii for the s and p states.

As a result, each compound is represented by a vector in $\mathbb{R}^{10}$. We used PCA to find a 2-D projection of the dataset. The result depicted in Figure 2.8 shows that PCA obtain very good clusters.

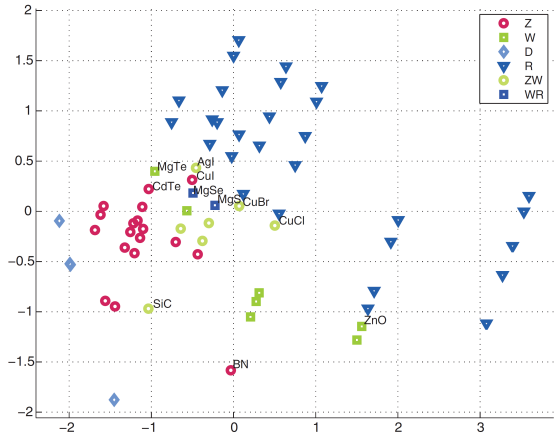


Figure 2.8: PCA Projection of 67 octet compounds.

Some methods for crystal structure classification and melting point prediction are also derived based on a combination of dimension reduction methods, classification techniques and regression analysis. We refer to [55] for more details.

2.7 Summary

Low dimensional approximation for data analysis can be viewed as a means of denoising data as well as a means for data compression. The denoising nature helps unravel hidden structures in the data which is a desired goal in data mining and knowledge discovery. The compression part is meaningful given the huge amount of data in modern applications.

Many low rank approximation problems can be cast as classical eigenvalue problems for which established solvers are available. As new problems emerge and constant streams of data are gathered, it is neccessary to explore numerical linear algebra methods combined with tools from convex optimization and probabilistic analysis to derive new methods.

Chapter 3

Computational methods for low rank approximation

In this chapter, we will discuss computational methods for low rank matrix approximation. Some of these methods are directly used in algorithms proposed in subsequent chapters while others are related techniques. Because complex valued matrices do not often appear in data analysis tasks, only the real case is considered. Nevertheless, most of the methods can be applied to complex matrices.

We will begin with two classical iterative techniques for eigenvalue problems: the subspace iteration and Krylov subspace methods. The first is conceivably the simplest method for eigenvector computation. It is robust to noise and a modification of it is proposed in Chapter 4 for matrix completion. The second is widely considered the most powerful technique for eigenproblems. The algorithm is known to converge fast for eigenvalues located in the outmost part of the spectrum. It will be used in Chapter 5 as well as in many algorithms for the matrix completion problem.

In the second part, an optimization perspective on low rank approximation is presented. This optimization viewpoint is useful because the objective functions can be generalized and adapted to other problems. We will also discuss low rank matrix approximation with nuclear norm minimization and its nonconvex formulation. Then, the framework of optimization on matrix manifolds is introduced. This is a framework developed fairly recently which provides elegent geometric insights into the numerical

problems along with efficient methods to solve these problems.

Finally, we will give a brief introduction to randomized low rank approximation algorithms of which low rank approximation with missing entries (Chapter 4) can be seen as a particular form.

3.1 Iterative methods for eigenvalue problems

3.1.1 Subspace iteration

Subspace iteration is a block generalization of the power method to compute the dominant subspaces of a matrix. It is regarded as a slow technique comparing to the Krylov subspace methods discussed in the next section. However, it is simpler to analyze, especially in the presence of small perturbations. Some perturbation analysis of these methods can be found in [21, 56].

Let $A \in \mathbb{R}^{n\times n}$ be a symmetric matrix and r be the number of eigenvectors needed. Starting from a random r-dimensional subspace represented by the orthogonal matrix U_0, a sequence of approximate eigenspaces spanned by matrices of the form $U_j = A^j U_0$ is generated. The approximate eigenvalues and eigenvectors can be retrieved within the current approximate subspace, $\text{span}\,(U_j)$, using a Rayleigh-Ritz projection. Denote by $\lambda_i(U_j^T A U_j)$ the i-th (largest) eigenvalue of $U_j^T A U_j$ and q_i the corresponding eigenvector. The approximate eigenvalue, $\tilde{\lambda}_i$, and the approximate eigenvector, $\tilde{u}_i$, of A are:

$$\begin{aligned} \tilde{\lambda}_i &= \lambda_i(U_j^T A U_j) \\ \tilde{u}_i &= U_j q_i \end{aligned} \tag{3.1}$$

Algorithm 3.1 shows a sketch of the method. Here, we adopt the Matlab notation $U = \text{qr}(X, 0)$ to state that U consists of an orthonormal basis of $\text{span}\,(X)$ obtained through some orthogonalization process, e.g., the Gram-Schmitd algorithm.

Algorithm 3.1 Subspace iteration for a symmetric matrix $A \in \mathbb{R}^{n\times n}$.

1: Select an initial orthogonal matrix $U_0 \in \mathbb{R}^{n\times r}$.
2: **for** j = 1,2,... **do**
3: $\quad \tilde{U}_j = AU_{j-1}$
4: $\quad U_j = \mathrm{qr}(\tilde{U}_j, 0)$
5: $\quad$ **if** (approximate eigenvalues and eigenvectors are desired) **then**
6: $\quad\quad$ *// Rayleigh-Ritz projection*
7: $\quad\quad [R_j, D_j] = \mathrm{eig}(U_j^T A U_j)$ *// D_j contains approximate eigenvalues*
8: $\quad\quad V_j = U_j R_j$ *// V_j contains approximate eigenvectors*
9: $\quad$ **end if**
10: **end for**

Recall the definition of the sine of the principal angle between two subspaces in equation (2.3). Define the cosine and the tangent between two subspaces as follows:

$$\cos\Theta(U,V) = \sigma_{min}(U^TV) \quad \text{and} \quad \tan\Theta(U,V) = \frac{\sin\Theta(U,V)}{\cos\Theta(U,V)}, \tag{3.2}$$

where $\sigma_{min}(X)$ is the minimum singular value of X. Theorem 3.1 establishes the linear convergence of subspace iteration in terms of the tangent of the principal angle between the subspace spanned by U_j and the dominant invariant subspace of A of dimension r (see e.g. [57, Theorem 1.1, Chapter 6]).

Theorem 3.1 *Let $\lambda_1, \ldots, \lambda_n$ be the eigenvalues of A in decreasing order. Let U be an orthonormal basis of the dominant r-dimensional subspace of A. Then*

$$\tan\Theta(U_j, U) \le \frac{\lambda_{r+1}}{\lambda_r} \tan\Theta(U_{j-1}, U) \tag{3.3}$$

As a result, at the m-th iteration:

$$\tan\Theta(U_m, U) \le \left(\frac{\lambda_{r+1}}{\lambda_r}\right)^m \tan\Theta(U_0, U) \tag{3.4}$$

For a general rectangular matrix A, the dominant r-dimensional subspaces of A refer to the left and right subspaces corresponding to the top-r singular vectors of A. These are in turn the top-r eigenspaces of AA^T and A^TA respectively. Algorithm 3.2 is a two-sided subspace iteration to simultaneously compute the left and right singular

subspaces of A. The algorithm is equivalent to two subspace iteration procedures of AA^T and A^TA.

Algorithm 3.2 Subspace iteration for a $m \times n$ rectangular matrix A.

1: Select initial orthogonal matrices $U_0 \in \mathbb{R}^{m\times r}$ and $V_0 \in \mathbb{R}^{n\times r}$.
2: **for** j = 1,2,... **do**
3: $\quad \tilde{U}_j = AV_{j-1}$
4: $\quad U_j = \mathrm{qr}(\tilde{U}_j, 0)$
5: $\quad \tilde{V}_j = A^TU_j$
6: $\quad V_j = \mathrm{qr}(\tilde{V}_j, 0)$
7: $\quad$ **if** (approximate eigenvalues and eigenvectors are desired) **then**
8: $\quad\quad [R_j^U, D_j, R_j^V] = \mathrm{svd}(U_{j+1}^T AV_{j+1})$
9: $\quad\quad U_j = U_jR_j^U$
10: $\quad\quad V_j = V_jR_j^V$
11: $\quad$ **end if**
12: **end for**

3.1.2 Krylov subspace methods

Krylov subspace methods are among the most important classes of available algorithms for computing eigenvalues and eigenvectors of large matrices. In subspace iteration, approximate subspaces of fixed dimension (dimension r in Algorithm 3.1 and Algorithm 3.2) are generated. In contrast, Krylov subspace methods are based on searching for the approximate eigenvectors in subspaces of increasing dimension.

An appealing aspect of Krylov subspace methods is that they need only a few iterations to approximate the extreme (the smallest and the largest) eigenvalues. Starting with an intial vector v, the Krylov subspace of order m of the matrix $A \in \mathbb{R}^{n\times n}$ is:

$$\mathcal{K}_m \triangleq \{v, Av, A^2v, \ldots, A^{m-1}v\} \tag{3.5}$$

This is exactly the subspace of vectors generated by the power method. The power method only uses the latest vector as the approximate eigenvector. Krylov subspace methods search for the approximate eigenvectors within the entire Krylov subspace that has been built so far.

First, we will describe the Arnoldi method to build an orthornormal basis of a Krylov subspace. The method is a combination of the power iteration and the Gram-Schmidt re-orthogonalization procedure. In practice, the modified Gram-Schmidt is used for better numerical stability.

Algorithm 3.3 Arnoldi algorithm.

1: Select initial vector v_1 of norm 1.
2: **for** j = 1,2,... **do**
3: $h_{ij} = \langle Av_j, v_i \rangle$ for $i = 1, 2, \ldots, j$
4: $w_j = Av_j - \sum_{i=1}^{j} h_{ij} v_i$
5: $h_{j+1,j} = \|w_j\|$
6: **if** $h_{j+1,j} == 0$ **then**
7: break
8: **end if**
9: $v_{j+1} = w_j / h_{j+1,j}$
10: **end for**

The algorithm generates an $m \times m$ Hessenberg matrix, H_m, consisting of h_{ij}'s up to the m-th iteration. The entries of H_m below the first subdiagonal are zeros. Denote by V_m the $n \times m$ matrix with column vectors $v_1, \ldots, v_m$. The following important relations hold:

$$AV_m = V_m H_m + h_{m+1,m} v_{m+1} e_m^T, \tag{3.6}$$

$$V_m^T A V_m = H_m, \tag{3.7}$$

where e_m is the unit vector with the only nonzero entry at the m-th position. It can be seen that if the matrix A is symmetric, the Hessenberg matrix H_m becomes a tridiagonal matrix. Let us denote this tridiagonal matrix by T_m. In this case, letting $\alpha_j = h_{jj}$ and $\beta_j = h_{j-1,j}$, the Arnoldi process amounts to performing the following three-term recurrence:

$$\beta_{j+1} v_{j+1} = Av_j - \alpha_j v_j - \beta_j v_{j-1}, \quad j = 1, 2, \ldots.$$

As a result, we obtain the Lanczos algorithm (Algorithm 3.4) which, among many of its interpretations, can be viewed as the Arnoldi process for symmetric matrices.

Algorithm 3.4 Lanczos algorithm for the symmetric $n \times n$ matrix A.

1: Select initial vector v_1 of norm 1. Set $\beta_0 = 0$ and $v_0 = 0$.
2: **for** j = 0,1,2,... **do**
3: $\quad w_j = Av_j - \beta_j v_{j-1}$
4: $\quad \alpha_j = \langle w_j, v_j \rangle$
5: $\quad w_j = w_j - \alpha_j v_j$
6: $\quad \beta_{j+1} = \|w_j\|_2$
7: $\quad$ **if** $\beta_{j+1} == 0$ **then**
8: $\quad\quad$ Break
9: $\quad$ **end if**
10: $\quad v_{j+1} = w_j / \beta_{j+1}$
11: **end for**

Given the current search subspace spanned by V_m, the Rayleigh-Ritz projection, which involves computing the eigenvalues and eigenvectors of T_m, is inexpensive. Similar to (3.1), let q_i be the i-th eigenvector of T_m, the approximate eigenvalues, $\tilde{\lambda}_i$, and the approximate eigenvectors, $\tilde{u}_i$, of A, are computed as follows:

$$\tilde{\lambda}_i = \lambda_i(T_m) \quad \text{and} \quad \tilde{u}_i = V_m q_i$$

Theorem 3.2, in [58] (see also [27, Theorem 6.3]), describes the rate of convergence of the angle between the eigenvectors of A and the Krylov subspace of order m.

Theorem 3.2 *[27, Theorem 6.3] The angle between the exact eigenvector u_i associated with λ_i and the t-th Krylov subspace $\mathcal{K}_m$ satisfies the inequality:*

$$\tan \Theta(u_i, \mathcal{K}_m) \leq \frac{\kappa_i}{C_{m-i}(1 + 2\gamma_i)} \tan \Theta(v_1, u_i), \tag{3.8}$$

where

$$\kappa_1 = 1, \kappa_i = \prod_{j=1}^{i-1} \frac{\lambda_j - \lambda_n}{\lambda_j - \lambda_i} \text{ for } i > 1$$

and,

$$\gamma_i = \frac{\lambda_i - \lambda_{i+1}}{\lambda_{i+1} - \lambda_n}.$$

The Rayleigh-Ritz projection is known to achieve the min-max optimality condition within the search subspace [27, Section 4.3.2]. Hence, the upper bounds obtained in Theorem 3.2 and Theorem 3.1 reflect the convergences of the approximate eigenvectors and eigenvalues of the corresponding algorithms.

For a very crude comparison between the convergence rates of subspace iteration and Lanczos algorithm for a symmetric matrix, let us consider the simple case where only the top eigenvector is needed. Let $\lambda_1 = (1+c)\lambda_2$ $(c > 0)$ and assume for simplicity that $\lambda_n = 0$. Thus, $\gamma_1 = c$. Then, $C_{t-1}(1 + 2\gamma_1) > \frac{1}{2}(1 + 4c)^{t-1}$.

Hence, for Lanczos iteration with initial vector v_1,

$$\tan\Theta(u_1, \mathcal{K}_m) \leq \frac{2}{(1+4c)^{m-1}} \tan\Theta(v_1, u_1).$$

For subspace iteration, with initial vector v_0, after m iterations, we have:

$$\tan\Theta(u_1, v_m) \leq \frac{1}{(1+c)^m} \tan(v_0, u_1).$$

Clearly, the rate of convergence of Krylov subspace methods is faster than that of subspace iteration. The advantage of Lanczos over subspace iteration is rooted in the fact that the characteristic polynomial $\bar{p}_m$ resulting from the Rayleigh-Ritz projection onto $\mathcal{K}_m$ is the polynomial which minimizes the norm $\|p(A)v_1\|_2$ over all monic polynomials of degree m.

In a similar vein, among iterative methods for solving symmetric positive linear systems, conjugate gradient iteration which is a Krylov subspace method converges faster than gradient descent iteration. To achieve the same error, the number of iterations of conjugate gradient is approximately only the square root of that of gradient descent (see e.g. [59, Theorem 38.5] or [60, Section 6.11.3]). In the optimization context, similar results are also obtained for first order methods to optimize a general function [61, Section 2.2].

For nonsymmetric matrices, two-sided iterations can be derived similarly to Algorithm 3.2. The next section discusses an optimization view of low rank approximation and the connection of gradient methods to subspace iteration (Algorithm 3.2).

3.2 An optimization perspective on low rank approximation

3.2.1 A non-convex optimization problem and its stationary points

Low rank approximation and eigenvalue computation can be looked from many different viewpoints. In classical literature, besides subspace iteration and Krylov subspace methods, a class of techniques is based on finding (U, Λ) that satisfies the nonlinear equation:

$$AU - U\Lambda = 0.$$

Along this line, methods such as Jacobi-Davidson [62], Newton iteration [63] and Newton-Sylvester iteration [64] have been proposed.

In this section, we consider the problem of approximating a general rectangular $m \times n$ matrix A from an optimization perspective. This viewpoint is useful because apart from the standard objective function based on the Frobenius norm of the residual, various problem-dependent objective functions can be used. The analysis of the standard case can provide insights to study related problems.

By the Eckart-Young theorem (Theorem 2.3), we know that computing the top-r singular subspaces of A is equivalent to solving the following optimization problem:

$$\underset{X \in \mathbb{R}^{m \times n}}{\text{minimize}} \ \|X - A\|_F^2 \text{ such that } \operatorname{rank}(X) = r. \tag{3.9}$$

This nonconvex optimization problem can be re-formulated in many different ways. One way is to solve the problem below:

$$\underset{U \in \mathbb{R}^{m \times r}, V \in \mathbb{R}^{n \times r}}{\text{minimize}} \ f(U, V) = \frac{1}{2}\|UV^T - A\|_F^2. \tag{3.10}$$

Hereafter, we will refer to the pair (U, V) as a point in the domain of the function f, $\mathbb{R}^{m \times n} \times \mathbb{R}^{n \times r}$. The global optimum of this nonconvex problem is not unique. However, the solution is unique up to the subspaces spanned by U and V. Specifically, if (U, V) is a global optimum, the set of global optima of the problem is:

$$\left\{(UR, VR^{-T}) \text{ for any invertible matrix } R \in \mathbb{R}^{r \times r}\right\}. \tag{3.11}$$

We will see shortly that equivalent classes of stationary points are also defined in this way. The partial gradients of f with respect to U and V are:

$$\text{grad}_U f(U,V) = (UV^T - A)V \tag{3.12}$$

$$\text{grad}_V f(U,V) = (UV^T - A)^T U \tag{3.13}$$

This shows that when the gradients vanish, U and V span a pair of corresponding left and right singular subspaces of A. Let $A = U_* \Sigma_* V_*^T$ be the SVD of A. The point (U,V) is a stationary point of f iff:

$$UV^T = U_* S V_*^T, \tag{3.14}$$

where S is the diagonal matrix containing at most r nonzero entries of Σ_* while the rest are replaced by zeros. It should be noted that although U and V have r columns, they can be of rank less than r. For example, denote by σ_i, u_i and v_i the i-th singular value, left singular vector and right singular vector of A respectively, then any (U,V) for which $UV^T = \sigma_i u_i v_i^T$ is a stationary point of problem 3.10.

It can be readily seen that if (U,V) is a stationary point, the pairs of the form (3.11) are also stationary points. Moreover, it has been shown that the stationary points are *all saddle points except for the global optima* [65]. In other words, Problem 3.10 has no local minimum. In Section 3.4, we will present an analysis of the objective function in (3.10) with an additional regularization term. The results in that section can be applied here when the regularization term is 0.

3.2.2 Gradient methods and their connections to subspace iteration

Even though $f(U,V)$ has no local minima, in general, a gradient-based algorithm to minimize $f(U,V)$ can be stuck at some saddle point. We will see that with careful treatment, some of these methods can avoid the saddle points. The convergence is obtained by relating the methods to subspace iteration.

Gradient descent with orthogonalization

Consider a gradient descent step to reduce the value of $f(U, V)$ with respect to U while keeping V fixed:

$$\begin{aligned} U_{new} &= U - \alpha(UV^T - A)V \\ &= U(I - \alpha V^T V) + AV. \end{aligned}$$

At any point (U, V), we know that we can use the transformation in (3.11) to obtain a new point on the same level set of f. That said, if we orthogonalize V, we effectively move to a new location on the same level set of f and do not sacrifice what we achieve so far. Having $V^T V = I$, if we choose $\alpha = 1$ (this step size makes sense indeed because the function $\text{grad}_U f$ has a local Lipschitz constant of 1 due to $V^T V = I$), then the descent update with orthogonalization becomes the subspace iteration step:

$$U_{new} = \text{qr}(AV, 0).$$

Similarly, if we fix U and perform a gradient descent step with orthogonalization to update V, we obtain:

$$V_{new} = \text{qr}(A^T U, 0).$$

We therefore have seen that *alternating gradient descent of f with step size 1 and orthogonalization is equivalent to subspace iteration.* As a result, this judicious gradient descent procedure effectively avoids all the saddle points and converges to a global optimum.

Note that if U and V are updated simultaneously instead of being updated in an alternating fashion, the algorithm is similar to the subspace iteration of the matrix:

$$B = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix}. \tag{3.15}$$

Variable metric gradient descent

Taking the derivative of $\text{grad}_U f(U, V)$ in (3.12) with respect to U, we can see that $V^T V$ contains the second order information of $f(U, V)$ with respect to U. Similarly, $U^T U$ is the second order information of $f(U, V)$ with respect to V. Naturally, one would like

to use a metric in the domain of f that reflects this information. Specifically, assuming that V^TV is invertible and employing the following inner product in $\mathbb{R}^{m\times r}$:

$$\langle W, Z\rangle = \text{Tr}\left[(V^TV)^{-1}W^TZ\right], \quad W, Z \in \mathbb{R}^{m\times r},$$

the partial gradient of $f(U, V)$ with respect to U is:

$$\text{grad}_U f(U, V) = (UV^T - A)V(V^TV)^{-1}. \tag{3.16}$$

Similarly, using $(U^TU)^{-1}$ to scale the inner product in $\mathbb{R}^{n\times r}$, we get:

$$\text{grad}_V f(U, V) = (UV^T - A)^TU(U^TU)^{-1}. \tag{3.17}$$

Using these gradients with step size 1, we obtain:

$$\begin{aligned} U_{new} &= U + (A - UV^T)V(V^TV)^{-1} \ = AV(V^TV)^{-1}, \\ V_{new} &= V + (A - UV^T)^TU(U^TU)^{-1} = A^TU(U^TU)^{-1}. \end{aligned} \tag{3.18}$$

When U and V are updated alternatingly, this is equivalent to the Alternating Least Squares algorithm which will be discussed next. Note that, different metrics are used in different iterations because of changing U's and V's.

Alternating Least Squares

Instead of moving along the gradient direction with step size 1, if we find U to minimize f while keeping V fixed, we have:

$$(U_{new}V^T - A)V = 0 \Leftrightarrow U_{new}V^TV = AV \Leftrightarrow U_{new} = AV(V^TV)^{-1}.$$

This is exactly the update we just obtained above. If $V^TV = I$, then $U_{new} = AV$. Similarly, fixing U to find V, we obtain $V_{new} = A^TU$. This shows that the Alternating Least Squares (ALS) algorithm with orthogonalization for minimizing f is also subspace iteration and therefore it converges to a global optimum. Subspace iteration with perturbation has been used to obtain convergence of ALS for the matrix completion problem [20, 21].

Note that the convergence of subspace iteration takes place almost surely (with probability 1) because if the initial point is completely in the nullspace of the dominant

subspace of A, then the algorithm under exact arithmetic will never converge to the global optimum.

We have discussed three different gradient-based algorithms to solve (3.10). For this particular objective function, the three algorithms with orthogonalization are essentially equivalent and they correspond to different interpretations of subspace iteration. If a different objective function is used, these methods may no longer be equivalent. We will discuss one such objective function for the matrix completion problem in Chapter 4.

If we directly use a gradient method, e.g. with a small constant step size and no orthogonalization, local convergence to a global optimum can still be achieved if the initial point is close enough to the global optimum. Specifically, for Problem 3.10, based on the condition of the stationary points (3.14), the following set is guaranteed to contain no saddle points:

$$\left\{(U, V) \text{ such that } \|UV^T - A\|_2 < \sigma_r(A)\right\}$$

Consequently, if the initialization falls into this set, gradient methods with appropriate step sizes will converge to the global optimum. However, the convergence rate can be more complicated to derive because it depends on the choice of the step sizes.

In addition, accelerated gradient descent [61, Section 2.2] and nonlinear conjugate gradient [66, Chapter 5] can be used to achieve faster convergence speed. These methods can be seen as Lanczos equivalents that use subspaces of fixed dimensions.

Other than (3.10), there are formulations based on minimizing some auxiliary functions. Two examples are:

$$f(U) = \min_{V \in \mathbb{R}^{n \times r}} \|UV^T - A\|_F^2 \tag{3.19}$$

$$f(U, V) = \min_{S \in \mathbb{R}^{r \times r}} \|USV^T - A\|_F^2 \tag{3.20}$$

Defined as the minimum of some functionals, these auxiliary functions have some optimality conditions which are extremely useful when deriving convergence bounds for corresponding methods. Examples of these formulations in matrix completion are [18, 67, 51] (see also Chapter 4). We will now turn our focus to low rank approximation with regularization.

3.3 Rank minimization and nuclear norm minimization

So far, we have considered low rank approximation problems when the desired rank is known. Following a general low rank assumption without an explicit rank constraint, many problems can be formulated as finding a matrix X with the minimum rank subject to some constraints. Rank minimization is NP-Hard in general and nuclear norm minimization is a tight convex relaxation of it. As a direct generalization of the ℓ_1 minimization problems in the emerging field, compressed sensing, nuclear norm minimization is attractive due to the very powerful tools in convex analysis to derive algorithms with guaranteed global convergence.

Let $f(X)$ be a convex loss function that we want to keep small, the general problem is:

$$\underset{X\in\mathbb{R}^{m\times n}}{\text{minimize}} \quad f(X) + \lambda\|X\|_* \tag{3.21}$$

The bigger λ is, the lower the rank the solution X tends to have. There are several algorithms to solve this problem. A particular class of algorithms is that of proximal gradient methods. Assume that f is differentiable and let $g(X) = \lambda\|X\|_*$, the proximal gradient iteration to minimize the function $h(X) = f(X) + g(X)$ is:

$$X_{k+1} = \text{prox}_g(X_k - \alpha_k \nabla f(X_k)),$$

where α_k is the step size and

$$\text{prox}_g(X) = \underset{Y\in\mathbb{R}^{m\times n}}{\text{argmin}} \left(\lambda\|Y\|_* + \frac{1}{2}\|Y - X\|_F^2\right). \tag{3.22}$$

The proximal operator minimizes the function $g(Y)$ while keeping Y not too far away from X. Let $X = U\Sigma V^T$ be the SVD of X, then the matrix Y that realizes $\text{prox}_g(X)$ in equation (3.22) is:

$$Y = U(\Sigma - \lambda I)_+ V^T, \tag{3.23}$$

where $(\Sigma - \lambda I)_+$ is the diagonal matrix whose entries are $\max(\sigma_{ii} - \lambda, 0)$. This is called the singular value shrinkage operator in the literature. Because of this thresholding, intermediate iterates and the final solution tend to have low rank. Performing an SVD computation in each iteration is the main reason for the slowness of algorithms of this type. To overcome this, one can try to compute the SVD approximately using a few Lanczos iteration. One can also use some randomized methods discussed in Section 3.6.

This class of algorithms is usually referred to as soft-thresholding iteration and the rank of the approximation can change in every iteration. If a truncated SVD with a fixed rank r is used instead of (3.23), i.e.,

$$X_{k+1} = \text{svd}(X_k - \alpha_k \nabla f(X_k), r),$$

then the method is called iterative hard thresholding (see e.g. [68, 69]). Note that, hard thresholding algorithms do not solve the nuclear norm minimization problem. Instead, it is directly related to the non-convex formulation discussed in Section 3.2 where an upper bound of the rank is placed. From the angle of low rank approximation, the top-r dominant subspaces of $X_k - \alpha_k \nabla f(X_k)$ are used as the new approximate subspaces.

There is also interesting work at the intersection of the two approaches. An upper bound of the rank is used in combination with a regularization term related to the nuclear norm. Let r be the upper bound of the rank, one then solve the problem below:

$$\underset{U \in \mathbb{R}^{m \times r}, V \in \mathbb{R}^{n \times r}}{\text{minimize}} \quad f_1(U, V) = f(UV^T) + \frac{\lambda}{2} \left(\|U\|_F^2 + \|V\|_F^2 \right) \tag{3.24}$$

This formulation is found to be related to nuclear norm minimization based on the following interesting property of the nuclear norm:

$$\|X\|_* = \min_{UV^T = X} \frac{1}{2} \left(\|U\|_F^2 + \|V\|_F^2 \right). \tag{3.25}$$

Restricting the rank, as a trade-off for the non-convexity, the dimension of the problem is much smaller and each iteration is much cheaper. Interesting results on the convergence to the global optimum and the equivalence of Problem 3.21 and Problem 3.24 in some particular cases have been derived (see e.g. [49, 70]). These results are inspired by the SDP formulations of the nuclear norm minimization problems and low rank algorithms for such SDPs [71]. There is also recent work [72] on combining the two approaches where each iteration is reduced to a low dimensional approximation but the dimensions can change among different iterations.

The next section discusses the equivalence of Problem (3.21) and Problem (3.24). We will also investigate the stationary points of the nonconvex formulation extending the discussion given in Section 3.2.

3.4 Regularized low rank approximations

In this section, we analyze two different formulations of the regularized low rank matrix approximation problem:

$$\underset{X\in\mathbb{R}^{m\times n}}{\text{minimize}} \quad f(X)\equiv\frac{1}{2}\|A-X\|_F^2+\lambda\|X\|_*, \tag{3.26}$$

$$\underset{U\in\mathbb{R}^{m\times r},V\in\mathbb{R}^{n\times r}}{\text{minimize}} \quad g(U,V)\equiv\frac{1}{2}\|A-UV^T\|_F^2+\frac{\lambda}{2}(\|U\|_F^2+\|V\|_F^2), \tag{3.27}$$

where $\lambda\geq 0$. The former is non-smooth and convex while the later is smooth and non-convex. A closed form solution to Problem (3.26) can be defined in terms of the SVD of A. It should be noted that iterative algorithms, such as Krylov subspace methods, are needed to compute the SVD of A. Meanwhile, algorithms such as alternating least squares and gradient descent methods can be used to solve (3.27).

3.4.1 The equivalence of the two problems

When the solution to (3.26) has rank r, the two objective functions have the same global minimum, i.e. the global minimum X of (3.26) and a global minimum (U,V) of (3.27) satisfy $X=UV^T$. We present in this section the result for a more general class of objective functions based on (3.21) and (3.24).

First, the following lemma establishes a well-known property of the nuclear norm (see e.g. [73, 74, 75]).

Lemma 3.3 *If $X=UV^T$, then*

$$\|X\|_*\leq\frac{1}{2}\left(\|U\|_F^2+\|V\|_F^2\right).$$

The equality holds iff U and V are two matrices such that $U^TU=V^TV$.

Proof: Let X be $m\times n$ with $m\geq n$ and $X=U_0\Sigma_0V_0^T$ be the thin SVD of X where $U_0\in\mathbb{R}^{m\times n}$, $V_0\in\mathbb{R}^{n\times n}$ and $\Sigma_0\in\mathbb{R}^{n\times n}$. Then, there exist $R\in\mathbb{R}^{n\times n}$ and $Q\in\mathbb{R}^{n\times n}$ such that $U=U_0R$ and $V=V_0Q$, respectively, and $RQ^T=\Sigma_0$. The inequality transforms to:

$$\|\Sigma_0\|_*\leq\frac{1}{2}\left(\|R\|_F^2+\|Q\|_F^2\right).$$

We have:

$$\|\Sigma_0\|_* = \mathrm{Tr}\,(\Sigma_0) = \mathrm{Tr}\,(RQ^T) \leq \|R\|_F \|Q\|_F \leq \frac{1}{2}\left(\|R\|_F^2 + \|Q\|_F^2\right).$$

The equalities hold iff $R = Q$ which implies $U^TU = V^TV$. In the reverse direction, when $U^TU = V^TV$, the squared singular values of UV^T are the eigenvalues of UV^TVU^T, which are the same as the eigenvalues of $V^TVU^TU = (U^TU)^2$. So the singular values of X are simply the eigenvalues of U^TU and so we have

$$\|UV^T\|_* = \sum \sigma_i[UV^T] = \sum \lambda_i[U^TU] = \|U\|_F^2 = \frac{1}{2}[\|U\|_F^2 + \|V\|_F^2]$$

□

The following proposition addresses the equivalence of two generalized forms of (3.26) and (3.27) (see also [74] for a similar result).

Proposition 3.4 *Consider the problems below:*

$$\underset{X\in\mathbb{R}^{m\times n}}{\text{minimize}} \quad F(X) + \lambda\|X\|_*, \quad \textit{and} \tag{3.28}$$

$$\underset{U\in\mathbb{R}^{m\times r}, V\in\mathbb{R}^{n\times r}}{\text{minimize}} \quad G(U,V) + \frac{\lambda}{2}(\|U\|_F^2 + \|V\|_F^2), \tag{3.29}$$

where $F : \mathbb{R}^{m\times n} \to \mathbb{R}$ and $G(U,V) = F(UV^T)$. If the solution to (3.28) has rank r, then the two objective functions have the same global minimum.

It is clear that when $F(X) = (1/2)\|X - A\|_F^2$, (3.28) and (3.29) become (3.26) and (3.27), respectively.

Proof: Writing X in the form $X = UV^T$, according to Lemma (3.3), we can see that

$$\begin{aligned}\min_X F(X) + \lambda\|X\|_* &= \min_{U,V} F(UV^T) + \lambda\|UV^T\|_* \\ &\leq \min_{U,V} G(U,V) + \frac{\lambda}{2}(\|U\|_F^2 + \|V\|_F^2)\end{aligned} \tag{3.30}$$

Consider now the minimizer X_* of the objective function in (3.28) and let $X_* = U_0\Sigma_0V_0^T$ be its thin SVD, where Σ is $k\times k$, with k being the rank of X_*, U_0 is $m\times k$ and V_0 is $n\times k$. Let $U_x \equiv U_0\Sigma^{1/2}$ and $V_x \equiv V_0\Sigma^{1/2}$. Then clearly, $X_* = U_xV_x^T$ and $U_x^TU_x = V_x^TV_x = \Sigma_0$. Again, using Lemma 3.3, we have $\|U_xV_x^T\|_* = \frac{1}{2}(\|U_x\|_F^2 + \|V_x\|_F^2)$. Hence,

$$\begin{aligned}\min_X F(X) + \lambda\|X\|_* &= G(U_x,V_x) + \frac{\lambda}{2}(\|U_x\|_F^2 + \|V_x\|_F^2) \\ &\geq \min_{U\in\mathbb{R}^{m\times k}, V\in\mathbb{R}^{n\times k}} G(U,V) + \frac{\lambda}{2}(\|U\|_F^2 + \|V\|_F^2)\end{aligned} \tag{3.31}$$

If the solution to (3.28) has rank r, inequalities (3.30) and (3.31) together prove the proposition. □

The next section describes the closed form of the global solution to (3.26) in terms of the SVD of A.

3.4.2 The solution to the convex problem

Let $X = U\Sigma V^T$ be the singular value decomposition of X and $\Sigma = \text{diag}(\{\sigma_i\}_{1\leq i\leq p})$ where $p = \min(m, n)$. We define the singular value shrinkage operator $\mathcal{D}_\lambda$ of X as follows [16]:

$$\mathcal{D}_\lambda(X) = U\text{diag}(\{\sigma_i - \lambda\}_+)V^T,$$

where $\{\sigma_i - \lambda\}_+ = \max(0, \sigma_i - \lambda)$.

Lemma 3.5 *[16, Theorem 2.1] $\mathcal{D}_\lambda(A)$ is the unique solution to Problem (3.26).*

Below is a proof of the lemma based on the subdifferential of the nuclear norm.
PROOF: The objective function is strictly convex, therefore the solution to (3.26) is unique. Let $X = U\Sigma V^T$ be the singular value decomposition of X. The subgradient of the nuclear norm, $\partial\|X\|_*$, is [15]:

$$\partial\|X\|_* = \left\{UV^T + W : W \in \mathbb{R}^{m\times n}, U^TW = 0, WV = 0, \|W\|_2 \leq 1\right\} \tag{3.32}$$

Let A have the singular value decomposition:

$$A = U_0\Sigma_0V_0^T + U_1\Sigma_1V_1^T,$$

where U_0 and V_0 (resp. U_1 and V_1) consist of the singular vectors associated with the singular values greater than λ (resp. not greater than λ). Let $\hat{X} = \mathcal{D}_\lambda(A)$. Then,

$$\begin{aligned}\hat{X} &= U_0\mathcal{D}_\lambda(\Sigma_0)V_0^T, \quad \text{and}\\ A - \hat{X} &= \lambda(U_0V_0^T + W), \quad \text{where } W = \lambda^{-1}U_1\Sigma_1V_1^T.\end{aligned}$$

Hence, $A - \hat{X} \in \lambda\partial\|\hat{X}\|_*$, which means $0 \in \hat{X} - A + \lambda\partial\|\hat{X}\|_*$. Therefore, $\hat{X}$ is the unique minimum of Problem (3.26). □

An immeditate implication of this lemma is: if $\sigma_r(A) > \lambda > \sigma_{r+1}(A)$, then the solution to Problem (3.26) has rank r. As we have shown, the nonconvex problem

(3.27) would have the same global solution. In the next section, we will show that this global solution is the unique local optimum of the objective function in (3.27) up to unitary transforms and other stationary points are saddle points.

3.4.3 The stationary points of the nonconvex optimization problem

In this section, we will derive the general form of *all stationary* points of Problem (3.27). First, observe that the partial gradients of g have the forms given below:

$$\begin{aligned}\text{grad}_U\ g(U,V) &= (UV^T - A)V + \lambda U,\\ \text{grad}_V\ g(U,V) &= (VU^T - A^T)U + \lambda V.\end{aligned}$$

Therefore, at any critical point $(\bar{U}, \bar{V})$, we have:

$$(A - \bar{U}\bar{V}^T)\bar{V} = \lambda \bar{U}, \tag{3.33}$$

$$(A - \bar{U}\bar{V}^T)^T\bar{U} = \lambda \bar{V}. \tag{3.34}$$

Denote by E the matrix $A - \bar{U}\bar{V}^T$ and multiply (3.33) to the left by $\bar{U}^T$ and (3.34) to the left by $\bar{V}^T$, we get

$$\bar{U}^T E\bar{V} = \lambda \bar{U}^T\bar{U}, \ \text{ and } \ \bar{V}^T E^T\bar{U} = \lambda \bar{V}^T\bar{V}.$$

This implies the important relation below.

Remark 1 *Any stationary point $(\bar{U}, \bar{V})$ of g satisfies:*

$$\bar{U}^T\bar{U} = \bar{V}^T\bar{V}. \tag{3.35}$$

As it turns out, this implies that for this particular $\bar{X} = \bar{U}\bar{V}^T$ we have: $\|\bar{X}\|_* = (\|\bar{U}\|_F^2 + \|\bar{V}\|_F^2)/2$.

Equations (3.33) and (3.34) are equivalent to:

$$A\bar{V} = \bar{U}(\lambda I + \bar{V}^T\bar{V}), \tag{3.36}$$

$$A^T\bar{U} = \bar{V}(\lambda I + \bar{U}^T\bar{U}). \tag{3.37}$$

This shows that $\bar{U}$ and $\bar{V}$ span corresponding left and right singular subspaces of A, respectively. Therefore, $\bar{U}$ and $\bar{V}$ have the forms:

$$\bar{U} = U_0 R_U \ \text{ and } \ \bar{V} = V_0 R_V \text{ with } R_U^T R_U = R_V^T R_V, \tag{3.38}$$

where $U_0 \in \mathbb{R}^{m\times r}$ and $V_0 \in \mathbb{R}^{n\times r}$ consist of arbitrary sets of r corresponding left and right singular vectors of A, respectively. Denote by Σ_0 the diagonal matrix of the associated singular values, i.e. $U_0^T A V_0 = \Sigma_0$. The constraint $R_U^T R_U = R_V^T R_V$ is to enforce condition (3.35). Plug (3.38) into (3.36) and left multiply both sides by U_0^T, we obtain:

$$\begin{aligned}\Sigma_0 R_V = R_U(\lambda I + R_V^T R_V) \Rightarrow R_V^T \Sigma_0^2 R_V &= (\lambda I + R_V^T R_V) R_U^T R_U (\lambda I + R_V^T R_V) \\ &= (\lambda I + R_V^T R_V) R_V^T R_V (\lambda I + R_V^T R_V).\end{aligned}$$

Let $R_V = PDQ^T$ be the SVD of R_V, the above equality becomes:

$$QDP^T \Sigma_0^2 PDQ^T = Q(D^2 + \lambda I) D^2 (D^2 + \lambda I) Q^T.$$

This implies

$$DP^T \Sigma_0^2 PD = (D^2 + \lambda I) D^2 (D^2 + \lambda I).$$

If $D > 0$, we have

$$P^T \Sigma_0^2 P = (D^2 + \lambda I)^2.$$

If the singular values in Σ_0 are not duplicated, then $|P|$ must be the identity matrix ($|P|$ is the entrywise absolute matrix of P) and

$$D^2 = \Sigma_0 - \lambda I.$$

As a result, we have $\bar{V} = V_0(\Sigma_0 - \lambda I)_+^{1/2} Q^T$, where Q is a unitary matrix. Similarly, we also obtain $\bar{U} = U_0(\Sigma_0 - \lambda I)_+^{1/2} R^T$, where R is a unitary matrix. The condition $\bar{U}^T \bar{U} = \bar{V}^T \bar{V}$ implies that $RD^2R^T = QD^2Q^T$. Thus, under the same assumption that the singular values in Σ_0 are not duplicated, we get $R = QH$ where $|H| = I$. Substitute $\bar{U}$ and $\bar{V}$ of these forms into (3.36) and (3.37), we obtain that R must equal Q.

If $D \not> 0$, meaning some of its diagonal entries are 0, it can be verified that $\bar{U}$ and $\bar{V}$ still have the forms: $\bar{U} = U_0 DR$ and $\bar{V} = V_0 DR$, where R is an arbitrary unitary matrix, and d_{ii} is either $\sqrt{(\Sigma_0(i,i) - \lambda)_+}$ or 0.

Lemma 3.6 *Let $p = \min(m, n)$, and $u_1, \ldots, u_p$ and $v_1, \ldots, v_p$ be the sets of left and right singular vectors of A, respectively. Assume that the singular values of A are simple.*

The pair (U, V) is a stationary point of Problem (3.27) if and only if they are of the forms given below:

$$U = U_0 DR \quad \textit{and} \quad V = V_0 DR, \tag{3.39}$$

where,

- $U_0 = [u_{\pi(1)}, \ldots, u_{\pi(r)}]$ *and* $V_0 = [v_{\pi(1)}, \ldots, v_{\pi(r)}]$, *where* π *is a subset of* $\{1, \ldots, p\}$ *and* $|\pi| = r$,
- $D \in \mathbb{R}^{r \times r}$ *is a diagonal matrix where* d_{ii} *is either 0 or* $\sqrt{(\sigma_{\pi(i)} - \lambda)_+}$, *and*
- R *is an arbitrary* $r \times r$ *unitary matrix.*

PROOF: Only the "if" part remains to be shown. We can easily check that all matrices of the form (3.39) satisfy (3.33) and (3.34). Hence, they are stationary points of Problem (3.27). □

3.4.4 Saddle points of the nonconvex optimization problem

We can further show that among the stationary points, there is one unique local optimum (up to unitary transforms), the rest are saddle points. This is an extension to the analysis given in [65].

Proposition 3.7 *The objective function g in Problem (3.27) has a unique local minimum up to unitary transforms. Let $u_1, \ldots, u_r$ and $v_1, \ldots, v_r$ be the left and right singular vectors of A, respectively, associated with the top-r singular values $\sigma_1 > \cdots > \sigma_r$. If $\sigma_r > \lambda$ and $\sigma_r > \sigma_{r+1}$, then the unique local minimum up to unitary transforms has the form below:*

$$\begin{aligned} U_* &= [u_1, \ldots, u_r]\mathrm{diag}([\sigma_1 - \lambda, \ldots, \sigma_r - \lambda])^{1/2} \\ V_* &= [v_1, \ldots, v_r]\mathrm{diag}([\sigma_1 - \lambda, \ldots, \sigma_r - \lambda])^{1/2} \end{aligned} \tag{3.40}$$

PROOF: Recall that the stationary points of g have the form (3.39). We will now show that if U_0 and V_0 have at least one column that is not a top r singular vector of A, the corresponding stationary point is a saddle point. This will be proved for the particular case below and all other cases can be proved similarly.

Consider the following stationary point:

$$\begin{aligned} U &= [u_1, \ldots, u_{r-1}, u_{r+1}]\text{diag}([\sigma_1 - \lambda, \ldots, \sigma_{r-1} - \lambda, (\sigma_{r+1} - \lambda)_+])^{1/2} \\ V &= [v_1, \ldots, u_{r-1}, v_{r+1}]\text{diag}([\sigma_1 - \lambda, \ldots, \sigma_{r-1} - \lambda, (\sigma_{r+1} - \lambda)_+])^{1/2} \end{aligned} \tag{3.41}$$

First, assume that $\sigma_{r+1} > \lambda$. Let $\Sigma_0 = \text{diag}([\sigma_1, \ldots, \sigma_{r-1}, \sigma_{r+1}])$. Consider the matrices:

$$\begin{aligned} W &= \left[u_1, \ldots, u_{r-1}, \frac{1}{\sqrt{\epsilon^2+1}}(u_{r+1} + \epsilon u_r)\right](\Sigma_0 - \lambda I)_+^{1/2} \\ Z &= \left[v_1, \ldots, v_{r-1}, \frac{1}{\sqrt{\epsilon^2+1}}(v_{r+1} + \epsilon v_r)\right](\Sigma_0 - \lambda I)_+^{1/2}. \end{aligned}$$

where $\epsilon > 0$. Clearly, $\|U\|_F = \|W\|_F$ and $\|V\|_F = \|Z\|_F$. Now, let $\alpha_i = (\sigma_i - \lambda)_+$, it can be shown that:

$$\|A - WZ^T\|_F^2 = \|A - UV^T\|_F^2 + 2(\sigma_{r+1} - \sigma_r)\frac{\epsilon^2 \alpha_{r+1}}{1+\epsilon^2}.$$

Because $\alpha_{r+1} > 0$ and $\sigma_{r+1} < \sigma_r$, the second term is negative. Thus,

$$\|A - WZ^T\|_F^2 < \|A - UV^T\|_F^2.$$

This is true for arbitrary $\epsilon > 0$. Therefore, in any neighborhood of (U, V), there exists a pair (W, Z) such that $g(U, V) > g(W, Z)$. As a result, (U, V) is a saddle point.

Now, if $\sigma_{r+1} \leq \lambda$, the matrix U and V span singular subspaces of dimensions less than r. In this case (3.41) becomes:

$$\begin{aligned} U &= [u_1, \ldots, u_{r-1}, u_{r+1}]\text{diag}([\alpha_1, \ldots, \alpha_{r-1}, 0])^{1/2}, \quad \text{and} \\ V &= [v_1, \ldots, v_{r-1}, v_{r+1}]\text{diag}([\alpha_1, \ldots, \alpha_{r-1}, 0])^{1/2} \end{aligned}$$

Consider the matrices:

$$\begin{aligned} W &= [u_1, \ldots, u_{r-1}, u_r]\text{diag}([\alpha_1, \ldots, \alpha_{r-1}, \epsilon])^{1/2}, \quad \text{and} \\ Z &= [v_1, \ldots, v_{r-1}, v_r]\text{diag}([\alpha_1, \ldots, \alpha_{r-1}, \epsilon])^{1/2} \end{aligned}$$

We have:

$$\|W\|_F^2 = \|U\|_F^2 + \epsilon \quad \text{and} \quad \|Z\|_F^2 = \|V\|_F^2 + \epsilon.$$

Also,

$$\begin{aligned} \|A - WZ^T\|_F^2 &= \sum_{i=1}^{r-1}(\sigma_i - \alpha_i)^2 + (\sigma_r - \epsilon)^2 + \sum_{i=r+1}^{\min(m,n)} \sigma_i^2 \\ &= \|A - UV^T\| + (\sigma_r - \epsilon)^2 - \sigma_r^2 \end{aligned}$$

Therefore,

$$2g(W,Z) = 2g(U,V) + (\sigma_r - \epsilon)^2 - \sigma_r^2 + 2\lambda\epsilon$$

Hence, $g(W,Z) < g(U,V)$ iff:

$$(\sigma_r - \epsilon)^2 - \sigma_r^2 + 2\lambda\epsilon < 0 \Leftrightarrow \lambda < \sigma_r - \epsilon/2.$$

Because $\lambda < \sigma_r$, when $0 < \epsilon/2 < \sigma_r - \lambda$, we have $g(W,Z) < g(U,V)$. Therefore, (U,V) is a saddle point of g. $\square$

A similar proof can be obtained by using the gradient and the Hessian of g.

In this section and in Section 3.2, we have seen that the global optima and stationary points are defined as classes of matrices spanning the same subspaces. To avoid the non-uniqueness of the solutions, one approach is to restrict the variable to some matrix manifolds. This brings us to the next section.

3.5 Optimization on matrix manifolds

In this section, we describe a framework for optimization algorithms with low rank constraints by exploiting the differentiable structures of the constraint sets. Combining the geometric abstraction and algebraic computation of matrix manifolds, this framework connects seemingly unrelated ideas in numerical linear algebra, provides insights into the convergence of scientific computational methods, as well as furnishes new and efficient algorithms for low rank constrained problems.

Instead of rigorously deriving the differential goemetry of matrix manifolds, we will give a brief intuition of the basic elements through a simple manifold, the sphere. Then, the computation of basic ingredients for a first order method on a commonly used matrix manifolds will be described. Extensive analysis on the differential geometry of these manifolds can be found in two excellent references [76, 77]. In this chapter, we will occasionally use the notation $\mathcal{M}$ for a general smooth manifold.

3.5.1 Basic geometry of a smooth manifold

The main purpose of the section is to illustrate basic elements of a smooth manifold through a sphere centered at $(0,0,0)$ in $\mathbb{R}^3$ denoted by $\mathbb{S}^2$:

$$\mathbb{S}^2 = \{x \in \mathbb{R}^3 : x^T x = 1\}.$$

A differentiable structure can be defined on this set and when endowed with this differentiable structure, $\mathbb{S}^2$ is a differentiable manifold. This manifold is one of the simplest forms of matrix manifold where the vectors x's can be seen as $\mathbb{R}^{3\times 1}$ matrices.

At a point x, we have a set of *tangent vectors* to the manifold at x. This set defines a vector space called *the tangent space* of the manifold $\mathbb{S}^2$ at x, denoted by $T_x\mathbb{S}^2$. Algebraically, the set is:

$$\{\xi \in \mathbb{R}^3 \mid \xi^T x = 0\}.$$

We can see that for every $x \in \mathbb{S}^2$, the tangent space at every point x is a 2-dimensional plane. This means the sphere $\mathbb{S}^2$ is a *2-dimensional manifold* because it can be locally embedded to a 2-D plane everywhere. Figure 3.1 illustrates the sphere with 3 different tangent planes at 3 different points.

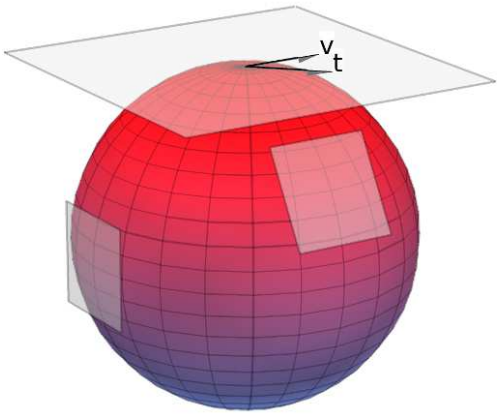


Figure 3.1: $\mathbb{S}^2$ and three tangent planes. The tangent spaces are 2-D Euclidean space. t and v are two tangent vectors.

Each tangent vector defines a *geodesic* starting from x. A geodesic is a generalization of a straight line in a linear space to a nonlinear manifold. When the manifold is endowed with a Riemannian metric, which will be introduced shortly, the geodesic is the shortest path between two points on the manifold. For $\mathbb{S}^2$, we can see that the geodesic starting at a point x in the direction of the tangent vector $\xi \in T_x\mathbb{S}^2$ is the circle on the sphere passing through x and tangent to ξ. Note that the geodesic needs not be closed for

general manifolds.

A geodesic can be approximated by a *retraction mapping*. In this case, a commonly used retraction for the geodesic starting at x along the direction $\xi \in T_x\mathbb{S}^2$ with step size t is:

$$\mathcal{R}_x(t\xi) = \frac{x + t\xi}{\|x + t\xi\|}. \tag{3.42}$$

Hence, along at

The tangent space $T_x\mathbb{S}^2$ is a linear space ($\mathbb{R}^2$) hence some *metric* (inner product) can be defined on it. A manifold whose tangent spaces are endowed with a smoothly varying inner product is a *Riemannian manifold* and the inner product is called a *Riemannian metric*. A linear space with its Euclidean metric is an example of a Riemannian manifold. In the case of $\mathbb{S}^2$, viewed as an embedded manifold within $\mathbb{R}^3$, the canonical metric used on every tangent space is the Euclidean metric inherited from $\mathbb{R}^3$:

$$\langle \xi_1, \xi_2 \rangle = \xi_1^T \xi_2.$$

Equipped with the Riemannian metric, we are now ready to introduce the notion of *gradient vector* of a function on the manifold. For a general manifold $\mathcal{M}$, the gradient of a smooth function $f : \mathcal{M} \to \mathbb{R}$ at $x \in \mathcal{M}$ is the unique tangent vector $\text{grad}_{\mathcal{M}} f(x) \in T_x\mathcal{M}$, such that:

$$\langle \text{grad}_{\mathcal{M}} f(x), \xi \rangle = Df(x)[\xi], \quad \forall \xi \in T_x\mathcal{M},$$

where $Df(x)[\xi]$ is the Frechet directional differentiation of f at x along the direction ξ. Note that the inner product is the Riemannian metric endowed with the manifold. For $\mathbb{S}^2$, the gradient vector is simply the projection of the gradient of the function f, viewed as a function in $\mathbb{R}^3$, onto the tangent space $T_x\mathbb{S}^2$. This intuition is also true for general manifolds embedded in a Euclidean space.

One last piece in the picture which is necessary to derive conjugate gradient methods on manifolds is *vector transport*. As we know, conjugate gradient methods, or other accelerated first order methods, use a combination of a few recent gradient vectors to compute the next search direction. In a linear manifold, the tangent space at every point is the manifold itself, the search direction is simply a linear combination of the gradient vectors. In a nonlinear manifold, gradient vectors are generally lying on different tangent planes. Therefore, we need a way to transport a gradient vector from one

tangent plane to another so that they can be linearly combined. Vector transport is the approximation of the concept of parallel transport in differential geometry similar to the way a retraction approximates a geodesic.

Indeed, vector transport can be defined based on retraction. For $\mathbb{S}^2$ endowed with the retraction defined in (3.42), let $x_1 \in \mathbb{S}^2$ and $x_2 = R_{x_1}(\eta) \in \mathbb{S}^2$, i.e. x_2 is a point on a curve starting from x_1 with direction $\eta \in T_{x_1}\mathbb{S}^2$. The vector transport $T_{x_1 \to x_2}(\xi_1)$ which transports the tangent vector $\xi_1 \in T_{x_1}\mathbb{S}^2$ to the tangent vector $\xi_2 \in T_{x_2}\mathbb{S}^2$ is defined as follows:

$$\xi_2 = T_{x_1 \to x_2}(\xi_1) = \frac{1}{\|x_1 + \eta\|} \left(I - \frac{1}{\|x_1 + \eta\|^2} (x_1 + \eta)(x_1 + \eta)^T \right) \xi.$$

The matrix $I - \frac{1}{\|x_1+\eta\|^2}(x_1 + \eta)(x_1 + \eta)^T$ in the above formula is simply the orthogonal projection onto the direction of $x_1 + \eta$ which is also the direction of x_2.

We just introduced some basic geometric elements of a manifold which are necessary to derive first order methods on matrix manifolds with some illustrations using the sphere $\mathbb{S}^2$. A direct generalization of the sphere is the set of matrices $V \in \mathbb{R}^{n \times r}$ satisfying $V^T V = I$. This set is called the Stiefel manifold. We will not discuss Stiefel manifold but a related manifold called the Grassmann manifold will be discussed in Section 3.5.3.

3.5.2 Gradient methods with manifold constraints

Consider the following constrained optimization problem:

$$\text{minimize } f(x) \text{ subject to } x \in \mathcal{C}$$

where f is a differentiable function and $\mathcal{C}$ is a general constraint set. Assuming we can compute a projection $\mathcal{P}_{\mathcal{C}}(x)$ of any point x onto $\mathcal{C}$, a typical algorithm to solve this problem is the projected gradient descent algorithm depicted in Algorithm 3.5.

Algorithm 3.5 PROJECTED GRADIENT DESCENT WITH CONSTRAINT SET $\mathcal{C}$.

1: Initialize $\hat{x}_0$, $x_0 = \mathcal{P}_{\mathcal{C}}(\hat{x}_0)$
2: **for** $t = 1, 2, ...$ until convergence **do**
3: $\quad \hat{x}_{t+1} = x_t - \alpha_t \text{grad} f(x_t)$ $\quad$ where α_t is an appropriate step size
4: $\quad x_{t+1} = \mathcal{P}_{\mathcal{C}}(\hat{x}_{t+1})$
5: **end for**

If f is a convex function and $\mathcal{C}$ is a convex set, it is known that the projected gradient descent converges to the global optimum [78]. The new iterate $\hat{x}$ may fall outside of $\mathcal{C}$ and the projection on line 4 projects the next iterate back onto $\mathcal{C}$. algorithm is the classical gradient method.

When $\mathcal{C}$ is a nonlinear smooth manifold $\mathcal{M}$, the constrained optimization can be cast as an unconstrained optimization where the function f is defined on the nonlinear domain $\mathcal{M}$. A crucial idea of gradient methods on manifolds is the generalization of *translating a point along a straight line* to *moving along a geodesic on the manifold*.

As mentioned in the previous section, each tangent vector in $T_x\mathcal{M}$ defines a geodesic starting from x. We need to choose a tangent vector corresponding to a descent direction of the function and move the point along the corresponding geodesic. Instead of strictly following the geodesic, one can follow an approximate curve using a retraction mapping. Under some conditions of the retraction, several convergence results can be derived similarly to gradient methods on linear manifolds. The prototypical gradient descent procedure on the manifold $\mathcal{M}$ is depicted in Algorithm 3.6.

Algorithm 3.6 Gradient Descent on the manifold $\mathcal{M}$.

1: Initialize $x_0 \in \mathcal{M}$
2: **for** $t = 1, 2, ...$ until convergence **do**
3: Compute the gradient of $f(x_t)$ on $T_x\mathcal{M}$: $\text{grad}_{\mathcal{M}} f(x_t)$
4: Find an appropriate step size α_t and compute: $x_{t+1} = \mathcal{R}_{x_t}(-\alpha_t \text{grad}_{\mathcal{M}} f(x_t))$
5: **end for**

For a manifold embedded in a linear space, the retraction on line 4 can be seen as a projection back to the manifold $\mathcal{M}$:

$$x_{t+1} = P_{\mathcal{M}}(x_t - \alpha_t \text{grad}_{\mathcal{M}} f(x_t)).$$

The gradient descent iteration on $\mathcal{M}$ in this case is very similar to the projected gradient descent with the only difference being that the search direction is $\text{grad}_{\mathcal{M}} f(x)$, the projection of the gradient $\text{grad} f(x)$ onto the current tangent space $T_x\mathcal{M}$. This is a form of the gradient projection method introduced by Rosen [79].

For conjugate gradient and other accelerated first order methods, the update on line 4 of Algorithm 3.6 is replaced by a combination of the current gradient with the vector transport of the previous search direction. A sketch can be found in Algorithm 3.7.

Algorithm 3.7 CONJUGATE GRADIENT METHOD ON THE MANIFOLD $\mathcal{M}$.

1: Initialize $x_0 \in \mathcal{M}$, $\eta_0 = 0$.
2: **for** $t = 1, 2, ...$ until convergence **do**
3: Compute the gradient of $f(x_t)$ on $T_x\mathcal{M}$: $\text{grad}_{\mathcal{M}} f(x_t)$
4: Compute an appropriate β_t and the search direction:

$$\eta_t = -\text{grad}_{\mathcal{M}} f(x_t) + \beta_t T_{x_{t-1} \to x_t}(\eta_{t-1}).$$

5: Find an appropriate step size α_t and compute: $x_{t+1} = \mathcal{R}_{x_t}(\alpha_t \eta_t)$
6: **end for**

matrix

3.5.3 The Grassmann manifold

The Grassmann manifold $\text{Gr}(m, r)$ is the set of subspaces of dimension r within $\mathbb{R}^m$. Recalling the notation $\text{span}\,(U)$, the column space of the matrix U, $\text{Gr}(m, r)$ is the following set:

$$\text{Gr}(m, r) = \{\text{span}\,(U) \mid U \in \mathbb{R}_*^{m \times r}\},$$

where $\mathbb{R}_*^{m \times r}$ is the set of $m \times n$ full-rank matrices. Each point on the Grassmann manifold can be represented in matrix form by any $\mathbb{R}^{m \times r}$ matrix whose columns form an orthonormal basis of the subspace.

In what follows, we will introduce the necessary elements of the Grassmann manifold for first order methods. We slightly abuse the notation when mentioning a matrix, say U, as a point on the Grassmann manifold. It needs to be understood that the point is the column space of U. The tangent space of the Grassmann manifold at the point U is the set:

$$\{\text{span}\,(\xi) \mid \xi^T U = 0, \xi \in \mathbb{R}^{m \times r}\}.$$

From this, we can see that the dimension of $\text{Gr}(m, r)$ is $(m - r)r$. The canonical retraction along ξ on the Grassmann manifold is:

$$\mathcal{R}_U(\xi) = span(U + \xi) \tag{3.43}$$

Note that $U + \xi$ is not an orthonormal matrix in general. Thus, $\mathcal{R}_U(\xi)$ should be represented by the matrix $\text{qr}(U + \xi, 0)$. The canonical inner product between any two

tangent vectors $\xi_1, \xi_2 \in T_U\mathrm{Gr}(m,r)$ is defined as follows:

$$\langle \xi_1, \xi_2 \rangle = \mathrm{Tr}\,(\xi_1^T \xi_2). \tag{3.44}$$

Using this metric, the orthogonal projection of a pencil Z onto the tangent space $T_U\mathrm{Gr}(m,r)$ is:

$$\mathcal{P}_U(Z) = (I - UU^T)Z. \tag{3.45}$$

The above formula will be used to compute the gradient vectors. Finally, transporting a vector $\xi \in T_U\mathrm{Gr}(m,r)$ to $T_V\mathrm{Gr}(m,r)$ is achieved by:

$$T_{U\to V}(\xi) = (I - VV^T)\xi. \tag{3.46}$$

In Chapter 4, we will discuss an application of optimization on the Grassmann manifold for the matrix completion problem. Non-canonical metrics on the Grassmann manifold are introduced to accelerate the gradient methods.

3.6 Randomized matrix approximation

The PCA method discussed in Section 2.1 and related problems have resurfaced in recent years in the context of "big data". With the enormous amount of available data, and the limited computational power, computing the SVD of a big matrix, or even performing a simple matrix-matrix multiplication can be costly. Also, data is often imperfect with noise and missing values. In many cases, one has to work with a random sample of the data matrix. This is where randomized algorithms emerge and promote a whole new area: randomized numerical linear algebra.

Algorithms for solving the low rank approximation problem with randomly missing entries, which will be addressed in Chapter 4, can be seen as a randomized matrix approximation approach. The matrix can be seen as being randomly sampled entry by entry and the obtained sparse matrix is used to compute the dominant subspace of the original matrix. To obtain recovery guarantees, some assumptions on the spikiness of the matrix (the incoherence properties, see Section 4.4) are needed.

In this section, we briefly touch upon a class of randomized methods for low rank matrix approximation. The only main assumption is a big gap in the singular spectrum. A typical example of a method of this type is the CUR decomposition [80].

A random subset of columns and rows of the big matrix $A \in \mathbb{R}^{m \times n}$ is selected based on some probability distribution depending on the norms of the columns and the rows. The sampled columns and rows form the two matrices $C \in \mathbb{R}^{m \times k}$ and $R \in \mathbb{R}^{n \times l}$ respectively.

Then, let $U = C^{\dagger} A R^{\dagger}$, where $X^{\dagger}$ denotes the Moore-Penrose pseudoinverse of X. Based on this, a low rank approximation of A can be obtained which is only slightly suboptimal comparing to the result achieved by a truncated SVD of A, $A_r = U_r \Sigma_r V_r^T$. If $k = O(r \log r / \epsilon^2)$ and $l = O(r \log r / \epsilon^2)$, then with high probability:

$$\|A - CUR\|_F \leq (2 + \epsilon) \|A - A_r\|_F.$$

The sampling process in this case is a Bernouli process. Other sampling operators can be found in [81]. Excellent references on the topic include but are not limited to [81, 82, 83, 84, 85].

Chapter 4

Low rank approximation with missing values

In this chapter, we discuss low rank approximations in the situation of matrices with missing values. This is the low rank matrix completion problem which we briely introduced in Section 2.5. It has attracted much attention recently [14, 18, 19, 86, 87] because of its broad applications, e.g., in recommender systems, structure from motion, and multitask learning (see e.g. [88, 89, 90]).

We will present algorithms to solve this problem by using data-dependent metrics on the Grassmann manifold to improve the method proposed in [18] while maintaining its appealing properties. The data-dependent metric introduces a scaling factor to the gradient of the objective function which can be interpreted as an adaptive preconditioner for the matrix completion problem. The gradient descent procedure using the scaled gradient is related to a form of subspace iteration for matrices with missing entries. Each iteration of the subspace iteration is inexpensive and the procedure converges very rapidly. The connection between the two methods leads to some improvements and to efficient implementations for both of them. Inspired by the scaled gradients on the Grassmann manifold, we also propose scaled metrics for gradient methods in linear domains to solve the regularized least squares problem for matrix completion. Acceleration methods to improve the convergence speed are also discussed. The first five sections of this chapter are based on publication [52].

4.1 Problem formulation and related work

Let $A \in \mathbb{R}^{m \times n}$ be a rank-r matrix, where $r \ll m, n$. The matrix completion problem is reconstructing A given a subset of entries of A. We say an entry is observed if its value is given. Let $\Omega = \{(i,j)|A_{ij} \text{ is observed}\}$. We define $P_\Omega(A) \in \mathbb{R}^{m \times n}$ to be the projection of A onto the set of observed entries Ω:

$$P_\Omega(A)_{ij} = \begin{cases} A_{ij} & \text{if } (i,j) \in \Omega \\ 0 & \text{otherwise.} \end{cases} \tag{4.1}$$

If the rank is unknown and there is no noise, the problem can be formulated as:

$$\text{Minimize rank}\,(X) \text{ subject to } P_\Omega(X) = P_\Omega(A). \tag{4.2}$$

As mentioned earlier in Section 3.3, rank minimization is NP-hard and so work has been done to solve a convex relaxation of it by approximating the rank by the nuclear norm. Under some conditions, the solution to the relaxed problem can be shown to be the exact solution to the rank minimization problem with high probability [14, 86]. Usually, algorithms to minimize the nuclear norm iteratively use the Singular Value Decomposition (SVD), specifically the singular value thresholding operator [16, 91, 92], which makes them expensive.

In this chapter, we focus on the case where the rank r is known. In fact, even when the rank is unknown, the singular spectrum of the sparse matrix $P_\Omega(A)$ can give us a very good approximation of the rank of A [18]. Also, a few values of the rank can be used and the best one is selected. Moreover, the singular spectrum is revealed during the iterations, so many fixed rank methods can also be adapted to find the rank of the matrix. In this case, we can formulate the matrix completion problem as follows:

$$\text{Find matrix } X \text{ to minimize } ||P_\Omega(X) - P_\Omega(A)||_F \text{ subject to rank}\,(X) = r. \tag{4.3}$$

A number of algorithms based on fixed-rank constraints use the framework of optimization on matrix manifolds [18, 50, 51]. In particular, Keshavan et al. [18] propose a steepest descent procedure on Grassmann manifolds with recovery guarantees under the incoherence conditions. Vandereycken [50] discusses a conjugate gradient algorithm on the Riemann manifold of rank-r matrices. Boumal and Absil [51] consider a trust region method on the Grassmann manifold. Although they do not solve an optimization

problem on the matrix manifold, Wei et al. [93] perform a low rank matrix factorization based on a successive over-relaxation iteration. Also, Srebro and Jaakkola [87] discuss SVD-EM, one of the early fixed-rank methods using truncated singular value decomposition iteratively. Dai et al. [67] recently proposed an interesting approach that does not use the Frobenius norm of the residual as the objective function but instead uses the consistency between the current estimate of the column space (or row space) and the observed entries. Guaranteed performance for this method has been established for rank-1 matrices.

Throughout this chapter, A_Ω will be a shorthand for $P_\Omega(A)$ and qf(U) is the Q factor in the QR factorization of U which gives an orthonormal basis for span (U). Also, $P_{\bar{\Omega}}(.)$ denotes the projection onto the complement of Ω.

4.2 Subspace iteration for incomplete matrices

We begin with a form of subspace iteration for matrix completion depicted in Algorithm 4.1. The algorithm is a modified version of the classical two-sided subspace iteration for singular value decomposition described in Chapter 3 (Algorithm 3.2).

Algorithm 4.1 SUBSPACE ITERATION FOR INCOMPLETE MATRICES.

Input: Matrix A_Ω, Ω, and the rank r.

Output: Left and right dominant subspaces U and V and associated singular values.

1: $[U_0, \Sigma_0, V_0] = \text{svd}(A_\Omega, r),\ S_0 = \Sigma_0$; // *Initialize* U, V *and* Σ
2: **for** i = 0,1,2,... **do**
3: $X_{i+1} = P_{\bar{\Omega}}(U_i S_i V_i^T) + A_\Omega$ // *Obtain new estimate of* A
4: $U_{i+1} = X_{i+1} V_i$; $V_{i+1} = X_{i+1}^T U_{i+1}$ // *Update subspaces*
5: $U_{i+1} = \text{qf}(U_{i+1})$; $V_{i+1} = \text{qf}(V_{i+1})$ // *Re-orthogonalize bases*
6: $S_{i+1} = U_{i+1}^T X_{i+1} V_{i+1}$ // *Compute new* S *for next estimate of* A
7: **if** condition **then**
8: // *Diagonalize* S *to obtain current estimate of singular vectors and values*
9: $[R_U, \Sigma_{i+1}, R_V] = \text{svd}(S_{i+1})$
10: $U_{i+1} = U_{i+1} R_U$
11: $V_{i+1} = V_{i+1} R_V$;
12: $S_{i+1} = \Sigma_{i+1}$.
13: **end if**
14: **end for**

The basic idea of Algorithm 4.1 is to use an approximation of A in each iteration to update the subspaces U and V and then from the new U and V, we can obtain a better approximation of A for the next iteration. Line 3 is to compute a new estimate of A by replacing all entries of $U_i S_i V_i^T$ at the known positions by the true values in A. The update in line 6 is to get the new S_{i+1} based on recently computed subspaces. Recall that lines 6-9 correspond to a Rayleigh-Ritz projection to obtain current approximations of singular vectors and singular values. This step is optional for matrix completion. Nevertheless, it could be useful for several purposes such as in regularization or for convergence test. This comes with very little additional overhead, since S_{i+1} is a small $r \times r$ matrix.

Each iteration of Algorithm 4.1 can be seen as an approximation of an iteration of the hard iterative thresholding algorithm described in Section 3.3 where only two matrix multiplications are used to update U and V instead of using a truncated SVD to compute the dominant subspaces of X_{i+1}. Recall that computing an SVD, e.g. by a Lanczos type procedure, requires several, possibly a large number of, matrix multiplications of

this type.

We now discuss efficient implementations of Algorithm 4.1 and modifications to speed-up its convergence. First, the explicit computation of X_{i+1} in line 3 is not needed. Let $\hat{X}_i = U_i S_i V_i^T$. Then

$$X_{i+1} = P_{\bar{\Omega}}(U_i S_i V_i^T) + A_\Omega = \hat{X}_i + E_i,$$

where $E_i = P_\Omega(A - \hat{X}_i)$ is a sparse matrix of errors at known entries which can be computed efficiently by exploiting the structure of $\hat{X}_i$. Assume that each S_i is not singular (the non-singularity of S_i will be discussed in Section 4.4). Then if we post-multiply the update of U in line 4 by S_i^{-1}, the subspace remains the same, and the update becomes:

$$U_{i+1} = X_{i+1} V_i S_i^{-1} = (\hat{X}_i + E_i) V_i S_i^{-1} = U_i + E_i V_i S_i^{-1}. \tag{4.4}$$

The update of V can also be efficiently implemented. Here, we make a slight change, namely $V_{i+1} = X_{i+1}^T U_i$ (U_i instead of U_{i+1}). We observe that the convergence speed remains roughly the same (when A is fully observed, the algorithm is a slower version of subspace iteration where the convergence rate is halved). With this change, we can derive an update to V that is similar to (4.4),

$$V_{i+1} = V_i + E_i^T U_i S_i^{-T}. \tag{4.5}$$

We will point out in Section 4.3 that the updating terms $E_i V_i S_i^{-1}$ and $E_i^T U_i S_i^{-T}$ are related to the gradients of a matrix completion objective function on the Grassmann manifold. As a result, to improve the convergence speed, we can add an adaptive step size α_i to the process, as follows:

$$\begin{aligned} U_{i+1} &= U_i + \alpha_i E_i V_i S_i^{-1} \\ V_{i+1} &= V_i + \alpha_i E_i^T U_i S_i^{-T}. \end{aligned}$$

This is equivalent to using $\hat{X}_i + \alpha_i E_i$ as the estimate of A in each iteration. The step size can be computed using a heuristic adapted from [93]. Initially, α is set to some initial value α_0 ($\alpha_0 = 1$ in our experiments). If the error $\|E_i\|_F$ decreases compared to the previous step, α is increased by a factor $\tau > 1$. Conversely, if the error increases, indicating that the step is too big, α is reset to $\alpha = \alpha_0$.

The matrix S_{i+1} can be computed efficiently by exploiting low-rank structures and the sparsity.

$$S_{i+1} = (U_{i+1}^T U_i) S_i (V_i^T V_{i+1}) + \alpha_i U_{i+1}^T E_i V_{i+1} \tag{4.6}$$

There are also other ways to obtain S_{i+1} once U_{i+1} and V_{i+1} are determined to improve the current approximation of A . For example we can solve the following quadratic program [18]:

$$S_{i+1} = \text{argmin}_S \|P_\Omega(A - U_{i+1} S V_{i+1}^T)\|_F^2 \tag{4.7}$$

We summarize the discussion in Algorithm 4.2.

Algorithm 4.2 Generic Subspace Iteration for incomplete matrices.

Input: Matrix A_Ω, Ω, and number r.

Output: Left and right dominant subspaces U and V and associated singular values.

1: Initialize orthonormal matrices $U_0 \in \mathbb{R}^{m\times r}$ and $V_0 \in \mathbb{R}^{n\times r}$.
2: **for** i = 0,1,2,... **do**
3: Compute E_i and appropriate step size α_i
4: $U_{i+1} = U_i + \alpha_i E_i V_i S_i^{-1}$ and $V_{i+1} = V_i + \alpha_i E_i^T U_i S_i^{-T}$
5: Orthonormalize U_{i+1} and V_{i+1}
6: Find S_{i+1} such that $P_\Omega(U_{i+1} S_{i+1} V_{i+1}^T)$ is close to A_Ω (e.g. via (4.6), (4.7)).
7: **end for**

A sufficiently small error $\|E_i\|_F$ can be used as a stoppping criterion. Algorithm 4.1 can be shown to be very similar to LMaFit algorithm proposed in [93]. The authors in [93] also obtain results on local convergence of LMaFit. We will pursue a different approach here. The updates (4.4) and (4.5) are reminiscent of the gradient descent steps for minimizing matrix completion error on the Grassmann manifold that is introduced in [18] and the next section discusses the connection to optimization on the Grassmann manifold.

4.3 Optimization on the Grassmann manifold for matrix completion

In this section, we show that by using a non-canonical Riemann metric on the Grassmann manifold, the gradient of the same objective function in [18] is of a form similar to (4.4)

and (4.5). Based on this, improvements to the gradient descent algorithms can be made and exact recovery results similar to those of [18] can be maintained. A brief introduction to the framework of optimization on matrix manifolds has been described in Section 3.5.

4.3.1 Scaled gradients on the Grassmann manifold

Let $\mathrm{Gr}(m, r)$ be the Grassmann manifold in which each point corresponds to a subspace of dimension r in $\mathbb{R}^m$. One of the results of [18], is that under a few assumptions (to be addressed in Section 4.4), one can obtain with high probability the exact matrix A by minimizing a regularized version of the function F: $\mathrm{Gr}(m, r) \times \mathrm{Gr}(n, r) \to \mathbb{R}$ defined below:

$$F(U, V) = \min_{S \in \mathbb{R}^{r \times r}} \mathcal{F}(U, S, V), \tag{4.8}$$

where

$$\mathcal{F}(U, S, V) = \frac{1}{2} \| P_\Omega(A - USV^T) \|_F^2,$$

$U \in \mathbb{R}^{m \times r}$ and $V \in \mathbb{R}^{n \times r}$ are orthonormal matrices. This function F is an extension of the objective function (3.20) for low rank approximation mentioned in Section 3.2. Here, we abuse the notation by denoting by U and V both othorthogonal matrices and the points on the Grassmann manifold which they span. Note that F only depends on the subspaces spanned by matrices U and V. The function $F(U, V)$ can be easily evaluated by solving the quadratic minimization problem of the form (4.7).

If $\mathrm{Gr}(m, r)$ is endowed with the canonical inner product $\langle W, W' \rangle = \mathrm{Tr}\,(W^T W')$, where W and W' are tangent vectors of $\mathrm{Gr}(m, r)$ at U, i.e. $W, W' \in \mathbb{R}^{m \times r}$ such that $W^T U = 0$ and $W'^T U = 0$ (see also Section 3.5.3); and similarly for $\mathrm{Gr}(n, r)$, the gradients of $F(U, V)$ on the product manifold $\mathrm{Gr}(m, r) \times \mathrm{Gr}(n, r)$ are:

$$\mathrm{grad} F_U(U, V) = (I - UU^T) P_\Omega(USV^T - A) VS^T \tag{4.9}$$

$$\mathrm{grad} F_V(U, V) = (I - VV^T) P_\Omega(USV^T - A)^T US. \tag{4.10}$$

In the above formulas, $(I - UU^T)$ and $(I - VV^T)$ are the projections of the derivatives $P_\Omega(USV^T - A)VS^T$ and $P_\Omega(USV^T - A)^T US$ onto the tangent space of the manifold at (U, V). Notice that the derivative terms are very similar to the updates in (4.4) and

(4.5). The difference is in the scaling factors where gradF_U and gradF_V use S^T and S while those in Algorithm 4.2 use S^{-1} and S^{-T}.

Assume that S is a diagonal matrix which can always be obtained by rotating U and V appropriately. $F(U, V)$ would change more rapidly when the columns of U and V corresponding to larger entries of S are changed. The rate of change of F would be approximately proportional to S_{ii}^2 when the i-th columns of U and V are changed, or in other words, S^2 gives us an approximate second order information of F at the current point (U, V). This suggests that the level set of F should be similar to an "ellipse" with the shorter axes corresponding to the larger values of S. It is therefore compelling to use a scaled metric on the Grassmann manifold.

Consider the inner product:

$$\langle W, W' \rangle_D = \mathrm{Tr}\,(DW^T W'),$$

where $D \in \mathbb{R}^{r \times r}$ is a symmetric positive definite matrix. We will derive the partial gradients of F on the Grassmann manifold endowed with this scaled inner product. According to [76], gradF_U is the tangent vector of $\mathrm{Gr}(m, r)$ at U such that

$$\mathrm{Tr}\,(F_U^T W) = \langle (\mathrm{grad} F_U)^T, W \rangle_D, \tag{4.11}$$

for all tangent vectors W at U, where F_U is the partial derivative of F with respect to U. Recall that the tangent vectors at U are those W's such that $W^T U = 0$. The solution to (4.11) with the constraints that $W^T U = 0$ and $(\mathrm{grad} F_U)^T U = 0$ gives us the gradient based on the scaled metric, which we will denote by $\mathrm{grad}_s F_U$ and $\mathrm{grad}_s F_V$.

$$\mathrm{grad}_s F_U(U, V) = (I - UU^T) F_U D^{-1} = (I - UU^T) P_\Omega (USV^T - A) V S D^{-1}, \tag{4.12}$$

$$\mathrm{grad}_s F_V(U, V) = (I - VV^T) F_V D^{-1} = (I - VV^T) P_\Omega (USV^T - A)^T U S D^{-1}. \tag{4.13}$$

Notice the additional scaling D appearing in these scaled gradients. Now if we use $D = S^2$ (still with the assumption that S is diagonal) as suggested by the arguments above on the approximate shape of the level set of F, we will have

$$\begin{aligned} \mathrm{grad}_s F_U(U, V) &= (I - UU^T) P_\Omega (USV^T - A) V S^{-1} \\ \mathrm{grad}_s F_V(U, V) &= (I - VV^T) P_\Omega (USV^T - A)^T U S^{-1} \end{aligned}$$

(Note that S depends on U and V).

If S is not diagonalized, we use SS^T and S^TS to derive $\text{grad}_s F_U$ and $\text{grad}_s F_V$ respectively, and the scalings appear exactly as in (4.4) and (4.5).

$$\text{grad}_s F_U(U,V) = (I - UU^T)P_\Omega(USV^T - A)VS^{-1} \quad (4.14)$$

$$\text{grad}_s F_V(U,V) = (I - VV^T)P_\Omega(USV^T - A)^T US^{-T} \quad (4.15)$$

This scaling can be interpreted as an adaptive preconditioning step similar to those that are popular in the scientific computing literature [94]. As will be shown in our experiments, this scaled gradient direction outperforms canonical gradient directions especially for ill-conditioned matrices.

The optimization framework on matrix manifolds allows several elements of the manifold to be defined in a flexible way. Here, we use the scaled-metric to obtain a good descent direction, while other operations on the manifold can be based on the canonical metric which has simple and efficient computational forms. The next two sections describe algorithms using scaled-gradients.

4.3.2 Scaled gradient descent on the Grassmann manifold

As introduced in Section 3.5, gradient descent algorithms on matrix manifolds are based on the update:

$$U_{i+1} = \mathcal{R}(U_i + \alpha_i W_i) \quad (4.16)$$

where W_i is the gradient-related search direction, α_i is the step size and $\mathcal{R}(U)$ is a retraction on the manifold. Here, we use the canonical retraction $\mathcal{R}(U) = \text{span}\,(U)$ on the Grassmann manifold where $\text{span}\,(U)$ is represented by $\text{qf}(U)$, which is the Q factor in the QR factorization of U. Optimization on the product of two Grassmann manifolds can be done by treating each component as a coordinate component.

The step size t can be computed in several ways, e.g., by a simple back-tracking method to find the point satisfying the Armijo condition [95]. Algorithm 4.3 is an outline of our gradient descent method for matrix completion. We let

$$\text{grad}_s F_U^{(i)} \equiv \text{grad}_s F_U(U_i, V_i), \quad \text{and}$$
$$\text{grad}_s F_V^{(i)} \equiv \text{grad}_s F_V(U_i, V_i).$$

In line 5, the exact S_{i+1} which realizes $F(U_{i+1}, V_{i+1})$ can be computed according to (4.7). A direct method to solve (4.7) costs $\mathcal{O}(|\Omega|r^4)$. Alternatively, S_{i+1} can be computed

approximately and we found that (4.6) is fast, $\mathcal{O}(|\Omega|r + (m+n)r^2)$, and gives the same convergence speed. If (4.6) fails to yield good enough progress, we can always switch back to (4.7) and compute S_{i+1} exactly. The subspace iteration and LMaFit can be seen as relaxed versions of this gradient descent procedure. We will also see in Section 4.6 that LMaFit is a form of gradient descent with variable metrics in the Euclidean space $\mathbb{R}^{m\times r} \times \mathbb{R}^{n\times r}$. The next section will describe the conjugate gradient iteration on the Grassmann manifold for matrix completion.

Algorithm 4.3 Scaled gradient descent on the Grassmann manifold.

Input: Matrix A_Ω, Ω, and number r.

Output: U and V which minimize $F(U,V)$, and S which realizes $F(U,V)$.

1: Initialize orthonormal matrices U_0 and V_0.
2: **for** i = 0,1,2,... **do**
3: Compute $\text{grad}_s F_U^{(i)}$ and $\text{grad}_s F_V^{(i)}$ according to (4.14) and (4.15).
4: Find an appropriate step size α_i and compute

$$(U_{i+1}, V_{i+1}) = (\text{qf}(U_i - \alpha_i \text{grad}_s F_U^{(i)}), \text{qf}(V_i - \alpha_i \text{grad}_s F_V^{(i)}))$$

5: Compute S_{i+1} according to (4.7) (exact) or (4.6) (approximate).
6: **end for**

4.3.3 Scaled conjugate gradient method on the Grassmann manifold

The scaled gradient can be used with a conjugate gradient (CG) iteration on the Grassmann manifold to increase the convergence speed. The method is a reminiscent of the classical preconditioned conjugate gradient method on linear manifold. Recall that we need to use vector transport to transport the old search direction to the current tangent plane on the manifold (see Section 3.5). Vector transport can be defined using the Riemann connection [77], which in turn is defined based on the Riemann metric. As mentioned at the end of Section 4.3.1, we will use the canonical metric to derive vector transport on the Grassmann manifold. Recall the canonical vector transport defined in (3.46):

$$T_{U\to V}(\xi) = (I - VV^T)\xi,$$

where ξ is the tangent vector at U. Here, we combine the transported search direction with the scaled gradient at the current point to derive the new search direction using the Polak-Ribiere formula defined below (see [76]):

$$\beta_{i+1} = \frac{\langle \Delta_U, \text{grad}_s F_U^{(i+1)} \rangle + \langle \Delta_V, \text{grad}_s F_V^{(i+1)} \rangle}{(\|\text{grad}_s F_U\|_F^2 + \|\text{grad}_s F_V\|_F^2)},$$

where

$$\Delta_U = \text{grad}_s F_U^{(i+1)} - T_{U_i \to U_{i+1}}(\text{grad}_s F_U^{(i)}), \quad \text{and}$$
$$\Delta_V = \text{grad}_s F_V^{(i+1)} - T_{V_i \to V_{i+1}}(\text{grad}_s F_V^{(i)}).$$

After this, a line search procedure is performed to find the appropriate step size along this search direction. Algorithm 4.4 is a sketch of the resulting conjugate gradient procedure.

Algorithm 4.4 CG WITH SCALED-GRADIENT ON THE GRASSMANN MANIFOLD.

Input: Matrix A_Ω, Ω, and number r.

Output: U and V which minimize $F(U, V)$, and S which realizes $F(U, V)$.

1: Initialize orthonormal matrices U_0 and V_0.
2: Compute $(\eta_0, \xi_0) = (\text{grad}_s F_U^{(0)}, \text{grad}_s F_V^{(0)})$.
3: **for** $i = 0,1,2,...$ **do**
4: Compute a step size α_i and compute $(U_{i+1}, V_{i+1}) = (\text{qf}(U_i + \alpha_i \eta_i), \text{qf}(V_i + \alpha_i \xi_i))$
5: Compute β_{i+1} (Polak-Ribiere) and set

$$(\eta_{i+1}, \xi_{i+1}) = (-\text{grad}_s F_U^{(i)} + \beta_{i+1} T_{U_i \to U_{i+1}}(\eta_i), -\text{grad}_s F_V^{(i)} + \beta_{i+1} T_{V_i \to V_{i+1}}(\xi_i))$$

6: Compute S_{i+1} according to (4.7) or (4.6).
7: **end for**

4.4 Convergence and recovery guarantees of scaled-gradient descent methods

4.4.1 Convergence and exact recovery

Let $A = U_* \Sigma_* V_*^T$ be the singular value decomposition of A, where $U_* \in \mathbb{R}^{m \times r}$, $V_* \in \mathbb{R}^{n \times r}$ and $\Sigma_* \in \mathbb{R}^{r \times r}$. Denote by $z = (U, V)$ a point on $\text{Gr}(m, r) \times \text{Gr}(n, r)$. Clearly,

$z_* = (U_*, V_*)$ is a minimum of F.

Let $\sigma_1^*, \ldots, \sigma_r^*$ be the singular values of A in decreasing order. Assume that A satisfies the following incoherence properties proposed in [18]:

- $\|U_*^{(i)}\|^2 \leq \mu(r/m)$, $\|V_*^{(j)}\|^2 \leq \mu(r/n)$, for all $i \in \{1..m\}$ and $j \in \{1..n\}$, and
- $|\sum_{k=1}^{r} U_*(i,k)(\sigma_k^*/\sigma_1^*)V_*(j,k)| \leq \mu\sqrt{r/(mn)}$, for all $i \in \{1..m\}$ and $j \in \{1..n\}$,

where μ is a constant, and $U_*^{(i)}$ and $V_*^{(j)}$ are the i-th row of U and j-th row of U_* and V_* respectively. In addition, assume that A has bounded entries and σ_r^* is bounded away from 0. Let $\kappa(A) = \sigma_1^*/\sigma_r^*$ be the condition number of A. It is shown that, if the set Ω is uniformly random and the number of observed entries is of order $O(\max\{\kappa(A)^2\mu nr\log n, \kappa(A)^6\mu^2 nr^2\})$ then, with high probability, F is well approximated by a parabola and z_* is the unique stationary point of F in a sufficiently small neighborhood of z_* ([18, Lemma 6.4&6.5]).

From these observations, given an initial point that is sufficiently close to z_*, a gradient descent procedure on F (with an additional regularization term to keep the intermediate points incoherent) converges to z_* and exact recovery is obtained. The singular value decomposition of a trimmed version of the observerd matrix A_Ω can give us an initial point that ensures convergence. The readers are referred to [18] for details.

In [18], a regularized version of F with the same minimum (U_*, V_*) is considered so that U and V remain incoherent during the execution of the algorithm. Here, we describe our results based on F for simplicity. These results also hold for the regularized function.

We will now show that the scaled-gradients of F are well-defined during the iterations. Moreover, they are indeed descent directions of F and only vanish at z_*. As a result, the scaled-gradient-based methods can inherit all the convergence results in [18].

First, S must be non-singular during the iterations for the scaled-gradients to be well-defined. As a corollary of Lemma 6.4 in [18], the extreme singular values, σ_{min} and σ_{max}, of any intermediate S are bounded by extreme singular values, $\sigma_{min}^* = \sigma_r^*$ and $\sigma_{max}^* = \sigma_1^*$, of A as follows:

$$\begin{aligned} \sigma_{max} &\leq 2\sigma_{max}^* \quad \text{and} \\ \sigma_{min} &\geq \frac{1}{2}\sigma_{min}^*. \end{aligned} \tag{4.17}$$

The second inequality implies that S is well-conditioned during the iterations. The scaled-gradient is the descent direction of F as a direct result from the fact that it is indeed the gradient of F based on a non-canonical metric.

The bounds in (4.17) also ensure that the scaled gradient vanishes if and only if the canonical gradient vanishes. Indeed, we have:

$$\|\mathrm{grad}_s F(z)\|^2 = \|\mathrm{grad} F_U(z)(SS^T)^{-1}\|_F^2 + \|\mathrm{grad} F_V(z)(S^TS)^{-1}\|_F^2.$$

Note that U and V can always be rotated so that S is a diagonal matrix. Having that, $D = S^TS = SS^T$ is a diagonal matrix whose minimum and maximum diagonal entries are σ_{min}^2 and σ_{max}^2, respectively. Hence,

$$\begin{aligned}
&\|\mathrm{grad}_s F(z)\|^2 \leq \sigma_{min}^{-2}(\|\mathrm{grad} F_U(z)\|_F^2 + \|\mathrm{grad} F_V(z)\|_F^2), \quad \text{and} \\
&\|\mathrm{grad}_s F(z)\|^2 \geq \sigma_{max}^{-2}(\|\mathrm{grad} F_U(z)\|_F^2 + \|\mathrm{grad} F_V(z)\|_F^2).
\end{aligned}$$

These imply

$$4(\sigma_{\min}^*)^{-2}\|\mathrm{grad} F(z)\|^2 \geq \|\mathrm{grad}_s F(z)\|^2 \geq (2\sigma_{max}^*)^{-2}\|\mathrm{grad} F(z)\|^2, \tag{4.18}$$

which confirms the equivalence between the scaled gradient and the canonical gradient.

By Lemma 6.5 in [18],

$$\|\mathrm{grad} F(z)\|^2 \geq Cn\epsilon^2(\sigma_{min}^*)^4 \mathrm{d}(z, z_*)^2 \tag{4.19}$$

for some constant C, where $\epsilon = |\Omega|/\sqrt{mn}$, and $\|.\|$ and $\mathrm{d}(.,.)$ are the canonical norm and distance on the Grassmann manifold respectively. Based on this, a similar lower bound of $\|\mathrm{grad}_s F\|$ can be derived:

$$\begin{aligned}
\|\mathrm{grad}_s F(z)\|^2 &\geq (2\sigma_{max}^*)^{-2}\|\mathrm{grad} F(z)\|^2 \\
&\geq (2\sigma_{max}^*)^{-2} Cn\epsilon^2(\sigma_{min}^*)^4 \mathrm{d}(z, z_*)^2 \\
&= C(\sigma_{min}^*)^4(2\sigma_{max}^*)^{-2} n\epsilon^2 \mathrm{d}(z, z_*)^2.
\end{aligned}$$

Therefore, the scaled gradients only vanish at z_*, which means the scaled-gradient descent procedure must converge to z_*, which is the exact solution [95].

4.4.2 Noisy matrix completion

When the rank-r matrix A is contaminated with the additive noise matrix N and what can be observed is $A_\Omega + N_\Omega$, where $N_\Omega = P_\Omega(N)$, a result similar to (4.19) can be derived [19, Lemma 5]:

$$\|\mathrm{grad}F(z)\|^2 \geq C_1 n\epsilon^2(\sigma^*_{min})^4 \left[\mathrm{d}(z, z_*) - C_2\frac{\sqrt{r}\sigma^*_{max}}{\epsilon\sigma^*_{min}}\frac{\|N_\Omega\|_2}{\sigma^*_{min}}\right]_+^2,$$

for some constants C_1 and C_2. As a result, any stationary point z of F (in the sufficiently small neighborhood of z_*), in the senses of both the scaled gradients and the canonical gradients (because of their equivalence (4.18)), is close to z_*:

$$\mathrm{d}(z, z_*) \leq C_3\frac{\sqrt{r}(\sigma^*_{max})^2\|N_\Omega\|_2}{\epsilon(\sigma^*_{min})^2},$$

for some constant C_3. Consequently, when the scaled gradient methods converge, the Frobenius distance between the matrix $X = USV^T$ to A is also upper bounded by a quantity proportional to $\|N_\Omega\|_2$. More details can be found in [19].

On the other hand, we can perform (scaled) gradient descent of the following regularized version of F [96]:

$$\tilde{F}(U, V) = \min_{S\in\mathbb{R}^{r\times r}} \left(\|P_\Omega(USV^T - A)\|_F^2 + \lambda\|S\|_F^2\right). \tag{4.20}$$

With an appropriate choice of λ, an error bound on the recovered matrix is also obtained in [96]. This regularization term has been shown to be very useful when the noise is large. This is also confirmed by our experiments with real datasets in the next section.

4.5 Experiments with scaled gradient methods on Grassmann manifolds

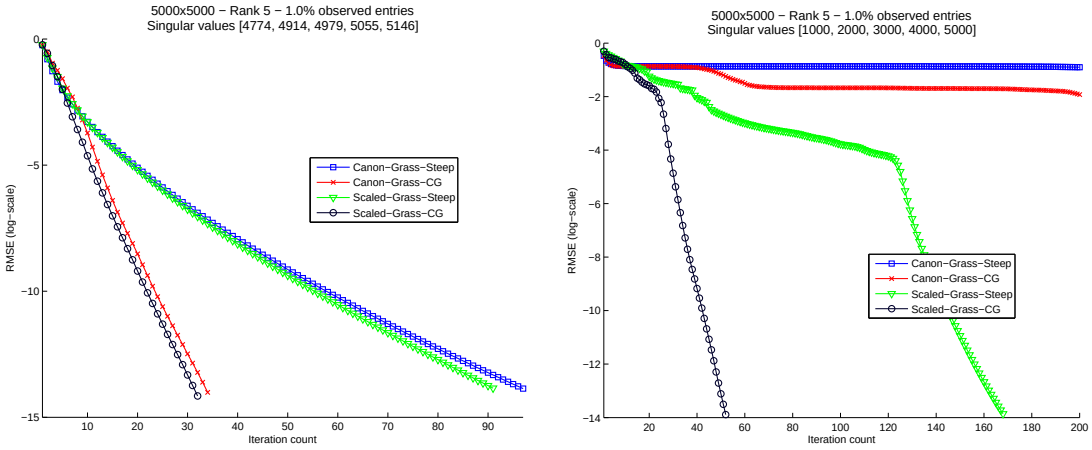
The proposed algorithms were implemented in Matlab with some mex-routines to perform matrix multiplications with sparse masks. For synthesis data, we consider two cases:

1. *Fully random low-rank matrices:* $A = \mathrm{randn}(m, r) * \mathrm{randn}(r, n)$ (in Matlab notations) whose singular values tend to be roughly the same; and

2. *Random low-rank matrices with chosen singular values:* $U = \mathrm{qf}(\mathrm{randn}(m, r))$, $V = \mathrm{qf}(\mathrm{randn}(n, r))$ and $A = USV^T$ where $S \in \mathbb{R}^{r \times r}$ is a diagonal matrix with chosen singular values.

The initializations of all methods are based on the SVD of A_Ω.

First, we illustrate the improvement of scaled gradients over canonical gradients for steepest descent and conjugate gradient methods on 5000×5000 matrices with rank 5 (Figure 4.1). Note that Canon-Grass-Steep is OptSpace with our implementation. In this experiment, S_i is obtained exactly using (4.7). The time needed for each iteration is roughly the same for all methods so we only present the results in terms of iteration counts. We can see that there are some small improvements for the fully random case (Figure 4.1a) since the singular values are roughly the same. The improvement is more substantial for matrices with larger condition numbers (Figure 4.1b).



(a) (b)

Figure 4.1: Log-RMSE for fully random matrix (a) and random matrix with chosen spectrum (b).

Now, we compare the relaxed version of the scaled conjugate gradient which uses (4.6) to compute S_i (ScGrass-CG) to LMaFit [93], Riemann-CG [50], RTRMC2 [51] (trust region method with second order information), SVP [68] and GROUSE [97] (Figure 4.2). These methods are also implemented in Matlab with mex-routines similar to ours except for GROUSE which is entirely in Matlab (Indeed GROUSE does not use

sparse matrix multiplication as other methods do). The subspace iteration method and the relaxed version of scaled steepest descent converge similarly to LMaFit, so we omit them in the graph. Note that each iteration of GROUSE in the graph corresponds to one pass over the matrix. It does not have exactly the same meaning as one iteration of other methods and is much slower with its current implementation. We use the best step sizes that we found for SVP and GROUSE.

In terms of iteration counts, we can see that for the fully random case (upper row), RTRMC2 is the best while ScGrass-CG and Riemann-CG converge reasonably fast. However, each iteraton of RTRMC2 is slower, so in terms of time, ScGrass-CG and Riemann-CG are the fastest in our experiments. When the condition number of the matrix is higher, ScGrass-CG converges fastest both in terms of iteration counts and execution time.

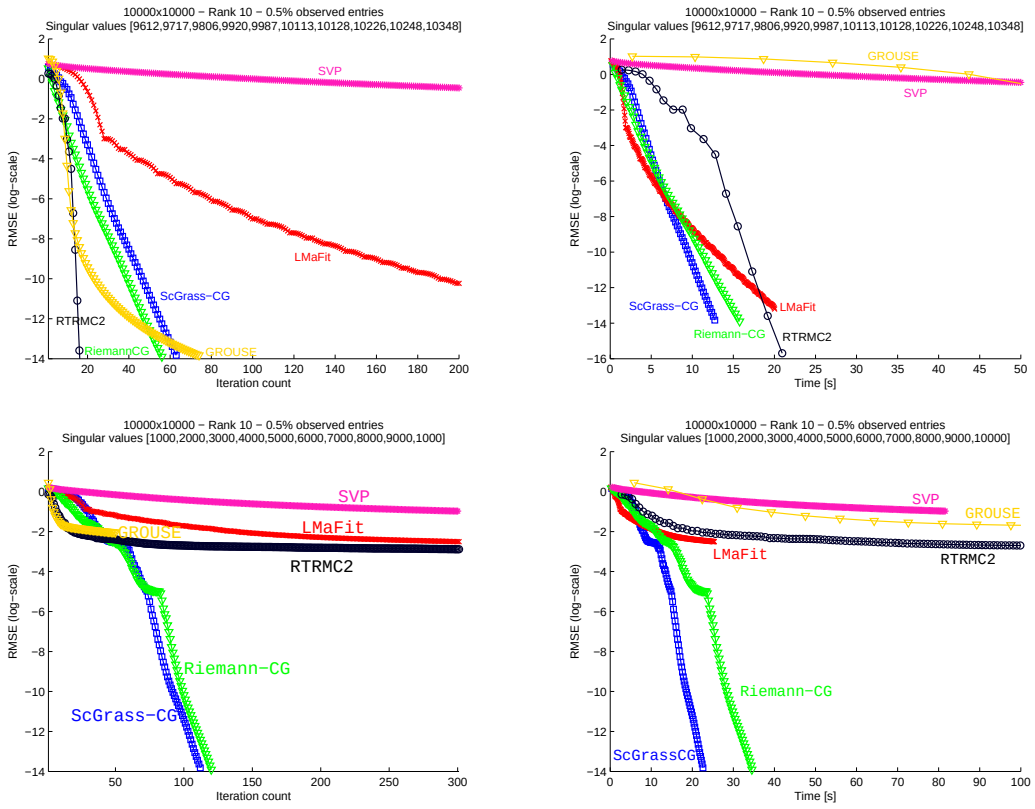


Figure 4.2: Log-RMSE. Upper row is fully random, lower row is random with chosen singular values.

Finally, we test the algorithms on Jester-1 and MovieLens-100K datasets which are assumed to be low-rank matrices with noise (SVP and GROUSE are not tested because their step sizes need to be appropriately chosen). Similarly to previous work, for the Jester dataset we randomly select 4000 users and randomly withhold 2 ratings for each user for testing. For the MovieLens dataset, we use the common dataset prepared by [98], and keep 50% for training and 50% for testing. We run 100 different randomizations of Jester and 10 randomizations of MovieLens and average the results. We stop all methods early, when the change of RMSE is less than 10^{-4}, to avoid overfitting. All methods stop well before one minute. The Normalized Mean Absolute Errors (NMAEs) [19] are reported in Table 4.1. ScGrass-CG is the relaxed scaled CG method and ScGrass-CG-Reg is the exact scaled CG method using the spectral-regularization

version of F in (4.20).

All methods perform similarly and demonstrate overfitting when $k = 7$ for MovieLens. We observe that ScGrass-CG-Reg suffers the least from overfitting thanks to its regularization. This shows the importance of regularization for noisy matrices and motivates future work in this direction.

Rank	ScGrass-CG	ScGrass-CG-Reg	LMaFit	Riemann-CG	RTRMC2
5	0.1588	0.1588	0.1588	0.1591	0.1588
7	0.1584	0.1584	0.1581	0.1584	0.1583
5	0.1808	0.1758	0.1828	0.1781	0.1884
7	0.1832	0.1787	0.1836	0.1817	0.2298

Table 4.1: NMAE on Jester dataset (first 2 rows) and MovieLens 100K. NMAEs for a random guesser are 0.33 on Jester and 0.37 on MovieLens 100K.

4.6 Scaled gradient methods in linear domain for regularized matrix approximation

4.6.1 Regularized least squares for matrix completion

In this section, we address one of the earliest regularized least squares formulations for matrix completion (see e.g. [13]):

$$\underset{U\in\mathbb{R}^{m\times r},V\in\mathbb{R}^{n\times r}}{\text{minimize}}\ g(U,V) \equiv \frac{1}{2}\|P_\Omega(UV^T - A)\|_F^2 + \frac{\lambda}{2}(\|U\|_F^2 + \|V\|_F^2). \tag{4.21}$$

As we discussed in Section 3.4.1, if the global solution to the problem:

$$\underset{X\in\mathbb{R}^{m\times n}}{\text{minimize}}\ \frac{1}{2}\|P_\Omega(X - A)\|_F^2 + \lambda\|X\|_* \tag{4.22}$$

is of rank r, then the two problems have the same global solution. Moreover, given a stationary point of $g(U,V)$ in (4.21), we can check if it is a global optimum based on the following proposition, which is inspired by Proposition 1 in [70].

Proposition 4.1 *At any nonzero stationary point of the objective function in (4.21), we have* $\|P_\Omega(A-UV^T)\|_2 \geq \lambda$. *If* $(\hat{U},\hat{V})$ *is a stationary point and* $\|P_\Omega(A-\hat{U}\hat{V}^T)\|_2 = \lambda$, *then* $(\hat{U},\hat{V})$ *is a global solution of (4.21) .*

PROOF: All stationary points of $g(U, V)$ satisfy:

$$P_\Omega(A - UV^T)V = \lambda U \quad \text{and} \quad P_\Omega(A - UV^T)^T U = \lambda V. \tag{4.23}$$

This implies

$$\lambda\|U\|_2 = \|P_\Omega(A - UV^T)V\|_2 \leq \|P_\Omega(A - UV^T)\|_2\|V\|_2$$
$$\lambda\|V\|_2 = \|P_\Omega(A - UV^T)^T U\|_2 \leq \|P_\Omega(A - UV^T)^T\|_2\|U\|_2$$

Adding the inequalities shows the first part of the proposition.

If $\|P_\Omega(A - \hat{U}\hat{V}^T)\|_2 = \lambda$, let $\hat{X} = \hat{U}\hat{V}^T$, from (4.23), it can be shown that:

$$P_\Omega(A - \hat{X}) = \lambda U_0 V_0^T + \lambda W,$$

where U_0 and V_0 are obtained from $\hat{U}$ and $\hat{V}$, respectively, by normalizing each of their columns to unit norm; and W is a matrix such that $U_0^T W = 0$, $WV_0 = 0$ and $\|W\|_2 \leq \lambda$. This means $\hat{X}$ satisfies the optimality condition of (4.22) based on the subgradient of the objective function. Therefore, $\hat{X}$ is a global solution of (4.22) and $(\hat{U}, \hat{V})$ is a global solution of (4.21). □

4.6.2 Scaled gradient descent in linear domain

Gradient descent is among the most widely used methods to solve (4.21) due to its simplicity and its efficient computation per iteration. The descent update is:

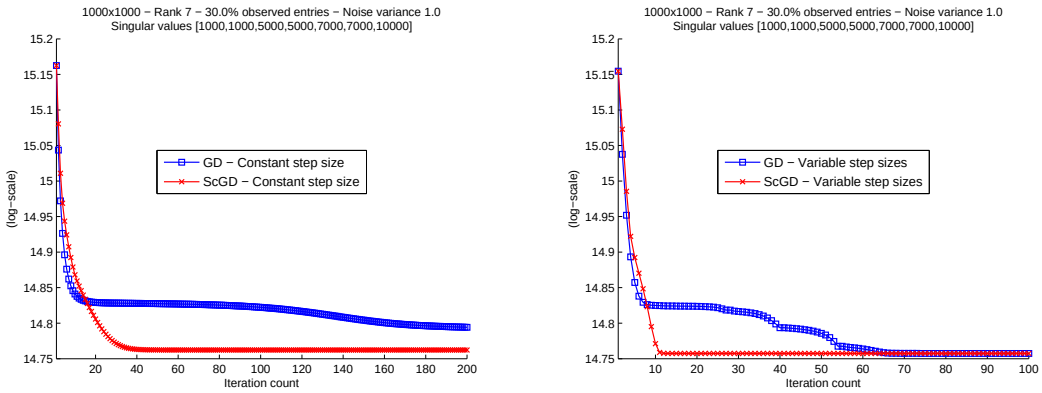
$$U_{i+1} = U_i - \alpha_i \left[P_\Omega(U_i V_i^T - A)V_i + \lambda U_i\right], \text{ and}$$
$$V_{i+1} = V_i - \alpha_i \left[P_\Omega(U_i V_i^T - A)^T U_i + \lambda V_i\right],$$

where α_i is the step size. Note that U_i and V_i are not orthonormalized because their norms contribute to the objective function. We will call this iteration the canonical gradient descent procedure. Indeed, scaled inner products can be employed in this situation. The scaled metrics would be similar to those of the variable-metric gradient descent procedure (3.18) discussed in Section 3.2.2. In this case, because of the regularization term, the scaling matrices are $(V^T V + \lambda I)^{-1}$ and $(U^T U + \lambda I)^{-1}$. The descent updates become:

$$U_{i+1} = U_i - \alpha_i \left[P_\Omega(U_i V_i^T - A)V_i + \lambda U_i\right](V_i^T V_i + \lambda I)^{-1}, \text{ and}$$
$$V_{i+1} = V_i - \alpha_i \left[P_\Omega(U_i V_i^T - A)^T U_i + \lambda V_i\right](U_i^T U_i + \lambda I)^{-1}.$$

When $\lambda = 0$, the algorithm is exactly the LMaFit algorithm [93]. This can be seen as an extension of LMaFit to deal with regularized matrix approximations. More importantly, other objective functions can be used in this context and acceleration schemes can be employed to improve the convergence speed. The new descent updates are also simple to compute and the additional cost is $\mathcal{O}(nr^2)$. Compared to the cost of the original updates, which is $\mathcal{O}(|\Omega|r)$, it is negligible when r is small. Moreover, $|\Omega|$ should be at least $\mathcal{O}(nr\log(n))$ for any recovery guarantees, and so the original cost would be at least $\mathcal{O}(nr^2\log(n))$.

We will now illustrate the advantage of the scaled gradient descent on synthesis data. We generate a 1000×1000 matrix with rank 7 and singular values 1000, 1000, 5000, 5000, 7000, 7000 and 10000. 30% of the matrix are uniformly observed and each observed entry is perturbed by Gaussian noise with mean 0 and variance 1.0 (we obtain similar results for other variances). We set $\lambda = 2\|P_\Omega(N)\|_2$ for this experiment where $P_\Omega(N)$ is the noise matrix. We observe that when λ is at this value or larger, gradient methods to solve (4.21) seem to always converge to the global solutions of the problems. By Proposition 4.1, we can check how close the obtained solution is to the global solution by computing the gap $\|P_\Omega(A - UV^T)\|_2 - \lambda$. A small gap means that the obtained solution is close to the global solution. Hereafter, we will refer to GD as the canonical gradient descent iteration and to ScGD as the scaled gradient descent iteration.



(A) (B)

Figure 4.3: The objective function in log-scale versus the number of iterations. The noise variance is 1.0. On the left are gradient methods with a fixed step size, on the right are gradient methods with variable step sizes.

Figure 4.3 shows the objective function in log-scale versus the number of iterations. The graph on the left depicts the convergence of gradient methods with a constant step size. Here, we choose $\alpha_i = 10^{-4}$ which is small enough for the methods to converge but still big enough to obtain fast convergence. Because the scaled gradient has an additional scaling matrix, for a fair comparison, we normalize it so that its Frobenius norm is equal to that of the canonical gradient in each iteration. The graph on the right is the result of gradient methods with variable step sizes using the heuristic mentioned in Section 4.2 (the initial α_0 is also 10^{-4}). We can see that ScGD is slightly slower than GD in the first few steps. After that, ScGD keeps converging while GD stagnates for a while and makes slow progress after that. Using variable step sizes does help improve the convergence speed and the improvement is about four times in this case. In both cases, ScGD seems to converge to the global solutions. The gap $\|P_\Omega(A - UV^T)\| - \lambda$ is 7.0e-06 after 100 iterations of ScGD.

To see the effect of the scaled gradients, we measure the distances between the subspace spanned by U_i to the 1 dimensional subspaces spanned by the true singular vectors. For this experiment, we illustrate the result in the noiseless case to make sure that we have the true solution. Figure 4.4 shows that GD initially moves very fast to the largest singular subspaces, corresponding to singular values 10000, 7000, and 5000. This

explains its slightly better convergence in the first few iterations. Meanwhile, it makes almost no progress toward the smallest singular vector. For ScGD, the convergence speed is almost the same for all singular vectors. This shows that GD is likely to be attracted to a non-global stationary point and makes slow progress when passing by the stationary point. In contrast, ScGD seems to avoid the non-global stationary points and is attracted toward the global optimum.

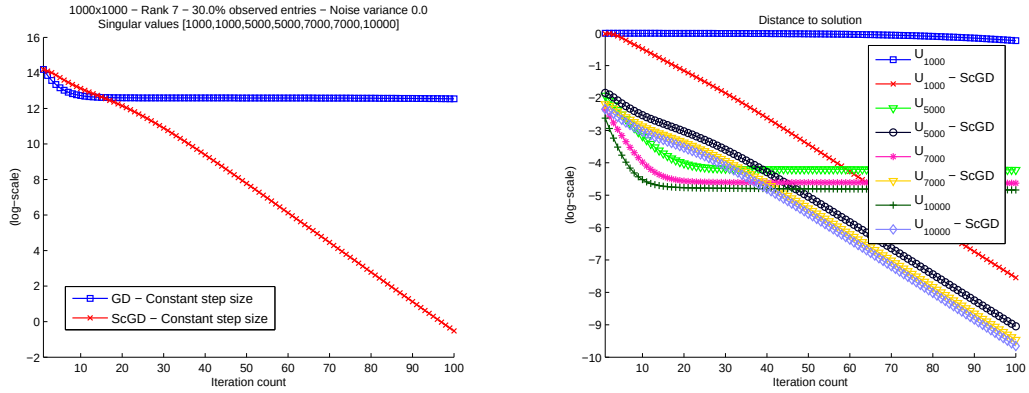


Figure 4.4: Low rank matrix completion without noise. On the left is the objective function and on the right are the distances to the true singular vectors.

We also test the methods for image inpainting task. We uniformly sample 50% of the pixels of the Lena image and use a rank 40 approximation to recover the whole image. The regularization term is set to $\lambda = 5$ and the constant step size is $\alpha_i = 0.0005$. Figure 4.5 illustrates the convergence of GD and ScGD. It can be seen that the scaled metric does help improve the convergence speed for both constant step size methods and variable step size methods in this case. The first stagnation point of GD with variable step sizes is at around iteration 20. ScGD seems to be very close to the global minimum at the same iteration (the gap $\|P_\Omega(A - UV^T)\|_2 - \lambda = 0.0278$ at iteration 100). We show in Figure 4.6 the original image, the image with 50% observed pixels and the images obtained by GD and ScGD at iteration 25. It is clear that ScGD achieves a much better result.

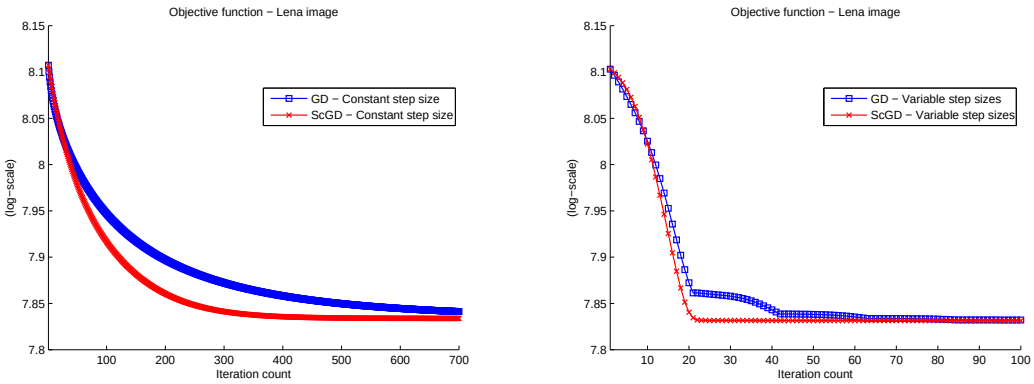
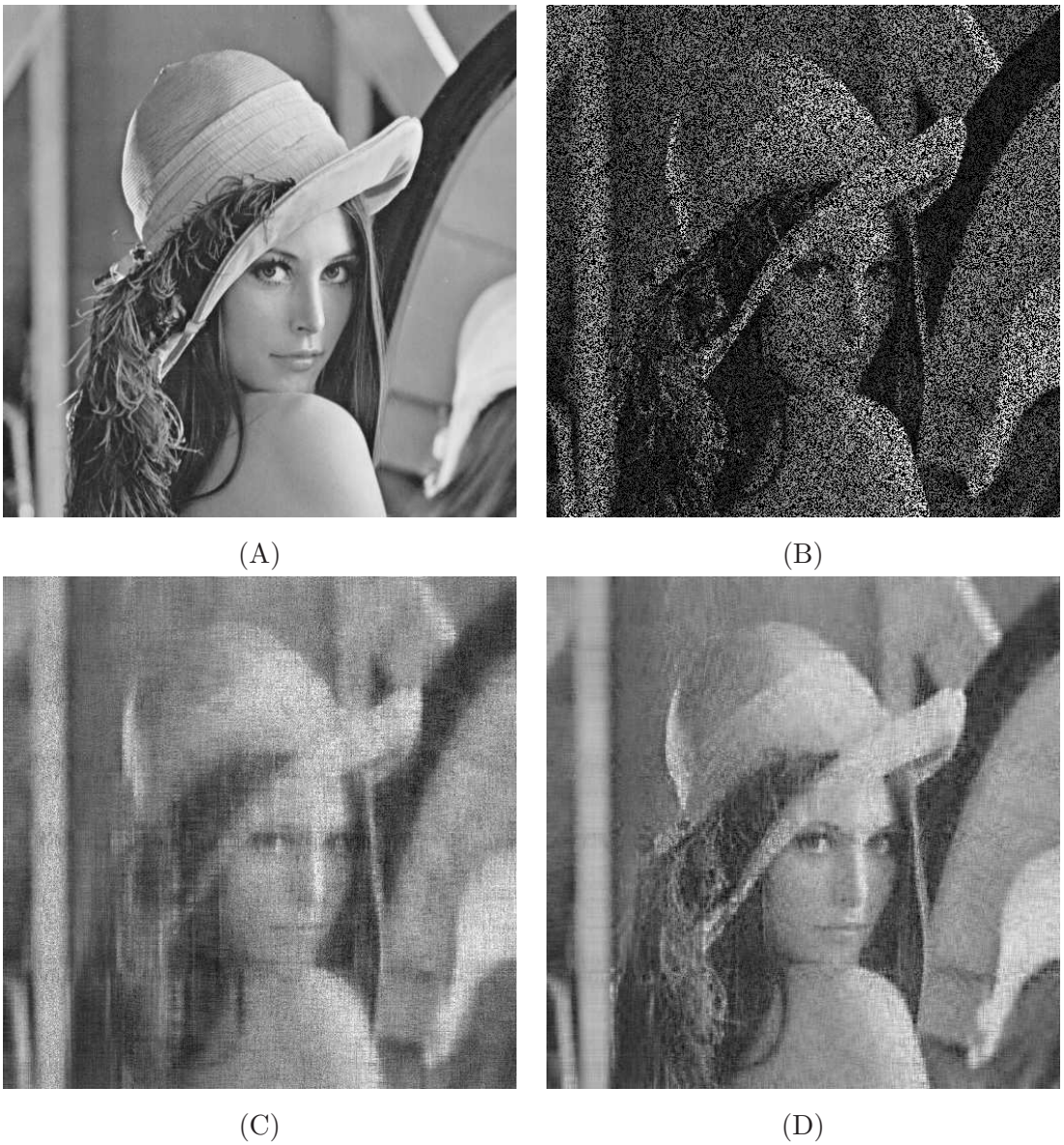


Figure 4.5: The objective function (log-scale) versus the number of iterations for image inpainting. On the left are gradient methods with a fixed step size, on the right are gradient methods with variable step sizes.



(A) (B)

(C) (D)

Figure 4.6: (A) Original image. (B) 50% observed image. (C) GD at iteration 25. (D) ScGD at iteration 25.

4.6.3 Acceleration techniques

It is possible to use acceleration techniques to speed up the convergence speed of gradient descent. The variable step size heuristic can be seen as a way to speed up the convergence

of gradient descent without using the (possibly) costly line search. Nonlinear conjugate gradient heuristics such as Polak-Ribiere can also be used here. In this section, we will demonstrate two other methods to accelerate gradient descent for matrix completion. Similar to conjugate gradient algorithms, these methods improve the convergence speed by combining two or more recent iterations.

The first method is the well-known Nesterov's acceleration scheme [61] which is popular for both smooth and non-smooth convex optimization. For convex and smooth objective function, Nesterov's method has been shown to achieve the optimal convergence rate for first order methods in terms of oracle complexity [61]. Note that, however, the objective function in this case is not convex. Denote by $x = (U, V)$ a point in $\mathbb{R}^{m\times r} \times \mathbb{R}^{n\times r}$, the Nesterov iteration is as follows:

$$y_{i+1} = x_i - \alpha_i \nabla g(x_i),$$
$$x_{i+1} = (1-\gamma_i) y_{i+1} + \gamma_i y_i,$$

where $\gamma_i = (1-\lambda_i)/(\lambda_{i+1})$ with $\lambda_0 = 0$ and $\lambda_{i+1} = (1+\sqrt{1+4\lambda_i^2})/2$. The step sizes α_i can be fixed or varied according to the heuristic.

The second method is a version of Aitken's delta-squared acceleration with relaxation [99]. Aitken acceleration belongs to a class of extrapolation methods for vector sequence acceleration. It is most useful for accelerating linearly converging vector sequences. Examples of other acceleration methods are Wynn's ϵ-algorithm, Anderson's mixing and minimal polynomial extrapolation (see e.g. [100, 101]). In the original form of Aitken acceleration for scalar series, given a convergent series $\{x_i\}$, the accelerated series $\{y_i\}$ is:

$$y_{i+1} = \frac{x_{i+1}x_{i-1} - x_i^2}{x_{i+1} - 2x_i + x_{i-1}}.$$

For vector sequences and to avoid numerical instability at the limit, the accelerated sequence is:

$$y_{i+1} = x_{i+1} + \rho_{i+1}(x_i - x_{i+1}),$$

where ρ_{i+1} is chosen differently for different versions of the method. In our experiments, we use the one proposed in [99]:

$$\rho_{i+1} = \gamma \left[\rho_i + (\rho_i - 1) \frac{(x_{i+1} - 2x_i + x_{i-1})^T (x_{i+1} - x_i)}{\|x_{i+1} - 2x_i + x_{i-1}\|^2} \right],$$

where $\gamma \in (0,1)$ is a parameter. When γ is larger, the convergence speed is faster although the convergence tends to be more fluctuating. Nevertheless, we observe that, in our experiments, the convergence speed is not very sensitive to γ and we choose $\gamma = 0.8$ in all experiments. The computation of ρ_i is $\mathcal{O}(nr)$, which is also negligible.

Aitken's delta-squared process and other acceleration methods are usually used to accelerate convergence of the fixed point iteration: $x_{i+1} = \phi(x_i)$. It has been shown that for scalar series, under some conditions, the convergence rate of Aitken's process is $[\phi'(x_*)]^2$ while that of the original fixed point iteration is $\phi'(x_*)$, where x_* is the unique fixed point of ϕ [102, Chapter 15]. When ϕ is a linear function, i.e. $\phi(x) = Ax$, the fixed point iteration is the power method (subspace iteration). Indeed, Aitken method has been used to accelerate the convergence of subspace iteration (see e.g. [103]). Based on the relationship between scaled gradient descent and subspace iteration, it is expected that Aitken method is also useful for scaled gradient descent. In this situation, we use Aitken's method in each iteration by viewing a descent update with a constant step size as an (approximate) fixed point iteration step. We have:

$$x_{i+1} = x_i - \alpha \nabla g(x_i).$$

Therefore, the Aitken iteration becomes:

$$y_{i+1} = x_i - \alpha(1 - \rho_{i+1})\nabla g(x_i),$$

where

$$\rho_{i+1} = \gamma \left[\rho_i + (\rho_i - 1) \frac{\langle \nabla g(x_{i-1}) - \nabla g(x_i), \nabla g(x_i) \rangle}{\|\nabla g(x_{i-1}) - \nabla g(x_i)\|^2} \right].$$

This shows that Aitken acceleration for gradient descent with a fixed step size is indeed a steepest descent iteration with step sizes computed according to an extrapolation process. In addition, it has recently come to our attention that another version of Aitken acceleration for gradient descent, which yields a similar iteration, is equivalent to the Barzilai-Borwein steepest descent method [104], which is a popular and practical method for unconstraint optimization. This connection was not addressed in Barzilai and Borwein's paper and deserves further study.

Figure 4.7 depicts the results for different values of the constant step size (the initial value α_0 of variable step sizes heuristic is the same as the constant step size) on the 1000×1000 random matrix generated the same way as earlier. We can see that for

$\alpha_i = 10^{-4}$ (Figure 4.7(A), the accelerated versions of ScGD perform similarly to ScGD with variable step size. If we look at the change in the objective function (Figure 4.7(B), we will see that Aitken acceleration converges fastest. We can also see this by checking the gap $\|P_\Omega(A - UV^T)\| - \lambda$, which indicates how close the solution is to the global optimum. After 70 iterations, this gap for ScGD - Aitken is 1.7644e-09 while that of ScGD - Nesterov with variable step size is 6.7676e-04. Note that 10^{-4} is the largest value of α_i for ScGD with constant step size to converge in this case. In practice, this step size should be carefully chosen and usually, smaller values are used to guarantee convergence. We can see in figures 4.7(C)-(D), when α_i is smaller, 10^{-5} and 10^{-6} respectively, ScGD Aitken is significantly faster than other methods. What is fascinating about ScGD Aitken is that the convergence speeds are almost the same for different step sizes. As we expected, the accelerated versions of GD also stagnate for a while as depicted in Figure 4.8.

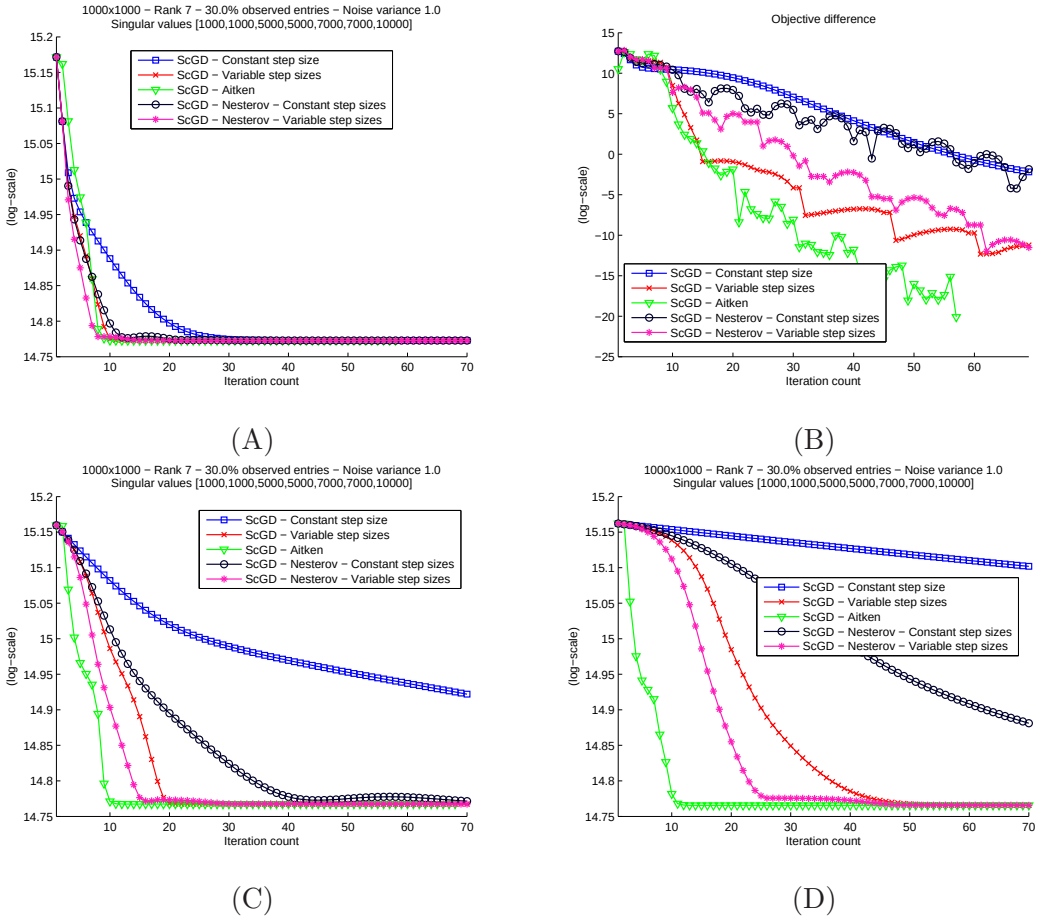


Figure 4.7: (A) Objective function, $\alpha_i = 10^{-4}$. (B) Changes in objective function, $\alpha_i = 10^{-4}$. (C) Objective function, $\alpha_i = 10^{-5}$ (D) Objective function, $\alpha_i = 10^{-6}$.

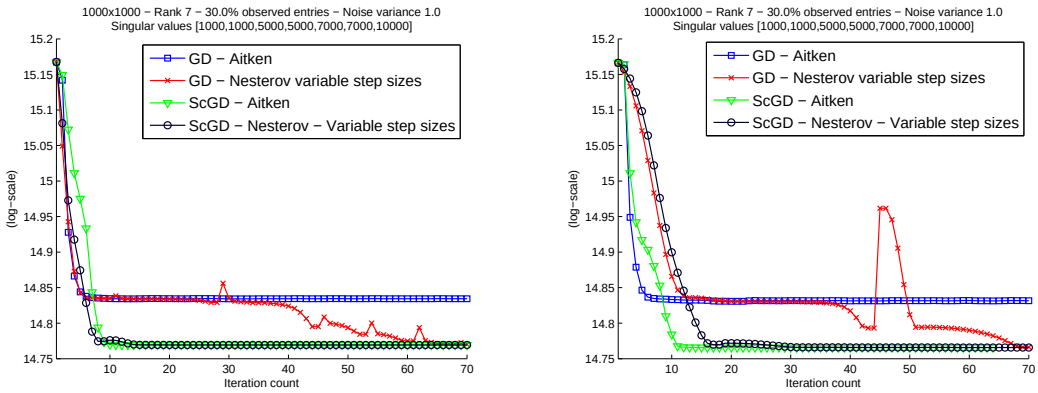


Figure 4.8: Accelerated versions of GD stagnate, while those of ScGD converge fast. (A) $\alpha_i = 10^{-4}$. (B) $\alpha_i = 10^{-5}$.

Finally, we test the methods on real datasets for matrix completion including Jester and Netflix datasets. For Jester dataset, we randomly sample 20% of the ratings and the resulted data matrix is 70×4000 with 17.3% observed entries. We use a rank 10 approximation and $\lambda = 50$. For Netflix dataset, the data matrix is 480189×17770 with 1.16% observed entries and we use a rank 10 approximation and $\lambda = 2000$. The initial step size is 10^{-4} for Jester and 10^{-5} for Netflix. The results are shown in Figure 4.9. It can be observed that scaled metrics help improve the convergence speed although the improvement is not as clear as it is for synthesis data. We examine the spectrum of the recovered matrices and find out that there are 1 or 2 big singular values, and the rest of the singular values are significantly smaller. This may explain why the improvement is not significantly observable as that for synthesis data. We also see that the acceleration techniques work quite well and Aitken's method outperforms other techniques for Jester. For Netflix dataset, Aitken's method and Nesterov's method have similar performances.

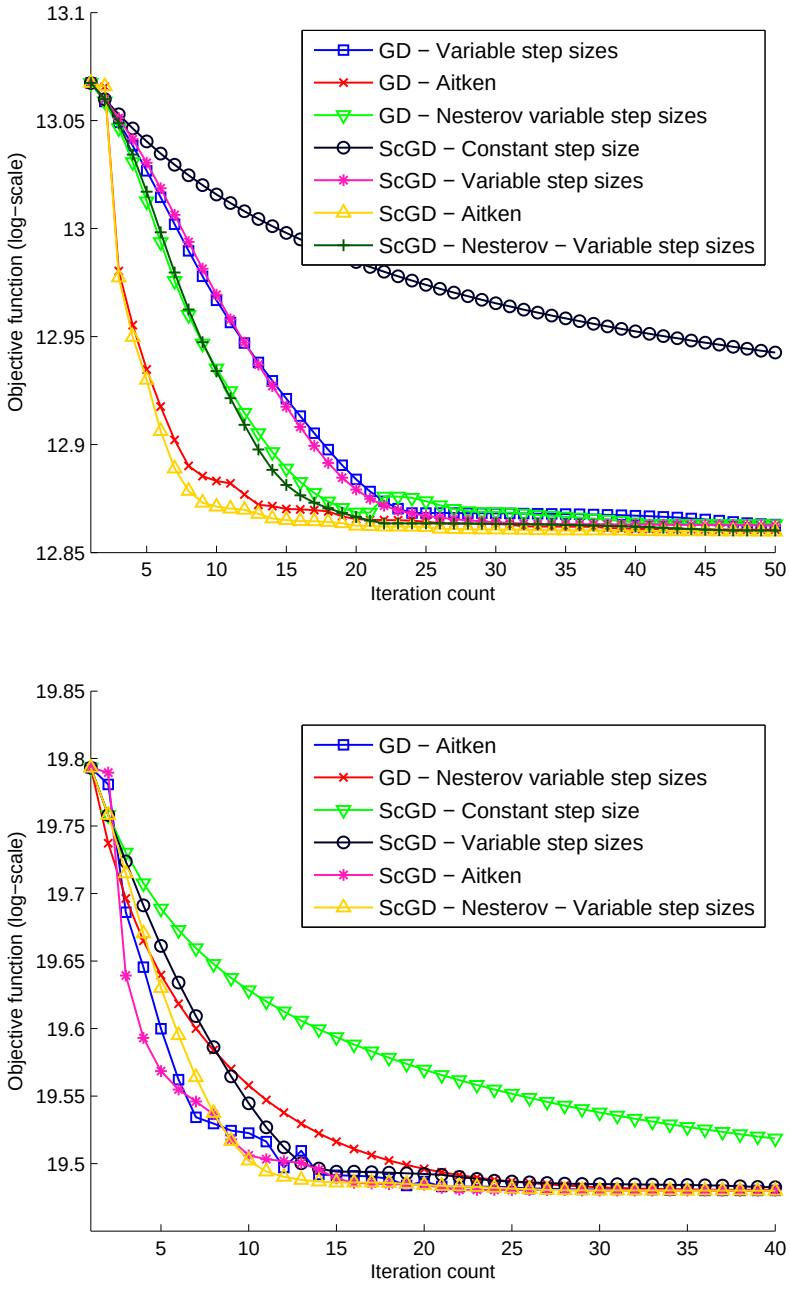


Figure 4.9: The objective function versus the number of iterations for Jester dataset (top) and Netflix dataset (bottom).

4.7 Summary and discussion

In this chapter, we examined the low rank approximation problem for incomplete matrices. We proposed a modified subspace iteration method to solve this problem efficiently. In addition, we have established a connection between scaled gradient methods on Grassmann manifolds and the subspace iteration method. The gradients obtained from a scaled metric on the Grassmann manifold can result in improved convergence of gradient methods on matrix manifolds for matrix completion while maintaining good global convergence and exact recovery guarantees. The relaxed versions of the proposed gradient methods, adapted from the subspace iteration, are faster than previously discussed algorithms, sometimes much faster depending on the conditionining of the data matrix. We then applied the scaled metric idea to the widely used gradient descent method in linear domain for matrix completion together with some acceleration techniques and obtained improvements on both synthesis data and real datasets.

It is interesting to investigate if the subspace iteration and the relaxed versions of the optimization on Grassmann manifolds achieve similar performance guarantees to those derived from OptSpace. The convergence condition of OptSpace depends on $\kappa(A)^6$ and weakening this dependency for the proposed algorithms is also an interesting future direction. The Aitken delta-squared acceleration works very well for this problem and it needs to be analyzed more carefully. It is also worth studying how well other vector acceleration techniques perform in this case.

Chapter 5

The trace ratio optimization problem

In the previous chapter, we discussed low rank approximation in the situation of missing entries. In this chapter, we discuss an extension of the classical low rank approximation in another direction. Specifically, we consider the optimization of the ratio of the traces of two matrices projected onto a low dimensional subspace. This problem can be applied to formulate the LDA method (discussed in Section 2.2) and several other supervised dimension reduction methods whose classical formulations are based on the generalized eigenvalue problem.

The material in this chapter is based on the paper [11]. The chapter is organized as follows. Section 5.1 gives some preliminaries on the trace ratio optimization problem and section 5.2 discusses the existence and uniqueness of a solution of this problem. The problem is then analyzed in detail in section 5.3 and an algorithm for finding the optimum of the trace ratio is proposed. Experiments of the proposed algorithm are reported in section 5.4 and concluding remarks are stated in section 5.5.

5.1 Preliminaries

Given a symmetric matrix A of dimension $n \times n$ and an arbitrary unitary matrix V of dimension $n \times r$, it is known that the trace of $V^T AV$ reaches its maximum (resp., minimum) when V is an orthogonal basis of the eigenspace of A associated with the r

algebraically largest (resp., smallest) eigenvalues. In particular, it is achieved for the eigenbasis itself: if eigenvalues are labeled decreasingly and $u_1, \ldots, u_r$ are eigenvectors associated with the first r eigenvalues $\lambda_1, \ldots, \lambda_r$, and $U = [u_1, \ldots, u_r]$, with $U^T U = I$, then,

$$\max_{V \in \mathbb{R}^{n \times r},\ V^T V = I} \text{Tr } \left[V^T A V\right] = \text{Tr } \left[U^T A U\right] = \lambda_1 + \cdots + \lambda_r. \tag{5.1}$$

This result is an immediate consequence of the Courant–Fisher characterization (2.4) which is used to formulate PCA. The optimal V is not unique since any system V that is an orthonormal basis of the eigenspace associated with the first r eigenvalues will be optimal. In other words, it is the subspace that matters rather than any particular orthonormal basis for the subspace.

As we discussed in Section 2.1, maximizing the trace in (5.1) requires the solution of a standard eigenvalue problem. Sometimes, it is necessary to maximize $\text{Tr } [V^T A V]$ subject to a new normalization constraint for V, one that requires that V be B-orthogonal; i.e., $V^T B V = I$. Assuming that A is symmetric and B is positive definite, we know that there are n real eigenvalues for the generalized problem $Au = \lambda Bu$ with B-orthogonal eigenvectors. If these eigenvalues are labeled decreasingly, and if $U = [u_1, \ldots, u_r]$ is the set of eigenvectors associated with the first r eigenvalues, with $U^T B U = I$, then we have:

$$\max_{V \in \mathbb{R}^{n \times r},\ V^T B V = I} \text{Tr } \left[V^T A V\right] = \text{Tr } \left[U^T A U\right] = \lambda_1 + \cdots + \lambda_r. \tag{5.2}$$

In reality, problem (5.2) often arises as a simplification of an objective function that is more difficult to maximize, namely

$$\max_{V \in \mathbb{R}^{n \times r},\ V^T C V = I} \frac{\text{Tr } \left[V^T A V\right]}{\text{Tr } \left[V^T B V\right]}. \tag{5.3}$$

Here B and C are assumed to be symmetric and positive definite for simplicity. The matrix C defines the desired orthogonality and in the simplest case it is just the identity matrix. The original version shown above has resurfaced in recent years; see, e.g., [4, 5, 6, 7, 8, 9, 10] among others. One of the main reasons for the regained interest in this problem is that it seems to yield markedly improved results for supervised learning tasks compared to its simplified counterpart (5.2).

5.2 Existence and uniqueness of a solution

There is no loss of generality in assuming that C is the identity matrix. Problem (5.3) may not have a solution when B is not positive definite. This is because in this situation it will be possible to find subspaces for which $\mathrm{Tr}\,[V^T BV]$ is zero while $\mathrm{Tr}\,[V^T AV]$ is nonzero, making the maximum ratio (5.3) infinite. A simple example is

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad V = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

It is helpful to examine the trace $\mathrm{Tr}\,[V^T BV]$ in detail. Let B be diagonalized into $B = Q\Lambda_B Q^T$, where Q is unitary and $\Lambda_B = \mathrm{diag}(\mu_1, \mu_2, \ldots, \mu_n)$. Let $v_1, \ldots, v_r$ be the columns of V, and define $\tilde{v}_j = Q^T v_j$. Then clearly

$$\mathrm{Tr}\,[V^T BV] = \sum_{j=1}^{p} \sum_{i=1}^{n} \mu_i \tilde{v}_{ij}^2 = \sum_{i=1}^{n} \mu_i \sum_{j=1}^{p} \tilde{v}_{ij}^2. \tag{5.4}$$

The following lemma examines the conditions under which $\mathrm{Tr}\,[V^T BV]$ is nonzero in the situation when B is positive semidefinite.

Lemma 5.1 *Assume that B is positive semidefinite and let r be the number of columns of V. If B has at most $r-1$ zero eigenvalues, then $Tr[V^T BV]$ is nonzero for any unitary V.*

PROOF: Using the previous notation, let $\tilde{V} = [\tilde{v}_1, \ldots, \tilde{v}_r]$. $\tilde{V}$ has at least one $r \times r$ submatrix which is nonsingular, so it has at least r rows that have a nonzero norm. Then in the sum (5.4), at least one of the $n - r + 1$ nonzero eigenvalues μ_i will coincide with one of these row norms and this sum will be nonzero. □

Therefore, the problem is well-posed under the condition that the null space of B is of dimension less than r, i.e., that its rank be at least $n - r + 1$. In this case the maximum is finite.

Proposition 5.2 *Let A, B be two symmetric matrices and assume that B is positive semidefinite with rank greater than $n-r$. Then the ratio* (5.3) *admits a finite maximum (resp., minimum) value ρ_*.*

PROOF: The set of matrices V such that $V^T V = I$ is closed and, under the assumptions, the ratio trace function in the right-hand side of (5.3) is a continuous function of its argument. Therefore, using Lemma 5.1 the maximum of the trace ratio (5.3) is reached. □

The maximum is reached for a certain V that will be characterized in the next section. As will be seen, under mild conditions, this V is *unique up to unitary transforms of the columns*.

This is indeed a mild condition as it is verified for all real datasets in our experiments on dimensionality reduction.

5.3 Conversion to a scalar problem

In the remainder of this chapter we will assume that C is the identity and that B satisfies the conditions of Proposition 5.2. From Proposition 5.2, we know that there is a maximum ρ_* that is reached for a certain (non-unique) orthogonal matrix, which we will denote by U_*. Then, for any orthogonal V we have $\mathrm{Tr}\,[V^T A V]/\mathrm{Tr}\,[V^T B V] \leq \rho_*$, and hence,

$$\mathrm{Tr}\,[V^T A V] - \rho_* \,\mathrm{Tr}\,[V^T B V] \leq 0.$$

This means that for this ρ_* we have $\mathrm{Tr}\,[V^T(A - \rho_* B)V] \leq 0$ for any orthogonal V, and also $\mathrm{Tr}\,[U_*^T(A - \rho_* B)U_*] = 0$. Therefore, we have the following necessary condition for the pair ρ_*, U_* to be optimal:

$$\max_{V^T V = I} \mathrm{Tr}\,[V^T(A - \rho_* B)V] = \mathrm{Tr}\,[U_*^T(A - \rho_* B)U_*] = 0. \tag{5.5}$$

According to (5.1), the maximum trace of $V^T(A - \rho_* B)V$ over all unitary matrices V of size $n \times r$ is simply the sum of the largest r eigenvalues of $A - \rho_* B$, and U_* is the set of corresponding eigenvectors. If ρ_* maximizes the trace ratio (5.3) (with $C = I$), then the sum of the largest r eigenvalues of the pencil $A - \rho_* B$ must be equal to zero, and the corresponding eigenvectors form the desired optimal solution of (5.3). This characterizes the optimal V of problem (5.3) as a set of eigenvectors associated with the r largest eigenvalues of $A - \rho_* B$. Any basis of this eigenspace will be optimal. If the eigenvalues of $A - \rho_* B$ are such that

$$\mu_1 \geq \mu_2 \cdots \geq \mu_r > \mu_{r+1} \geq \mu_{r+2} \geq \cdots \geq \mu_n,$$

then this eigenspace is unique and the solution V, which is any orthonormal basis of this space, is unique up to unitary transforms.

Consider now the function

$$f(\rho) = \max_{V^T V = I} \mathrm{Tr}\,[V^T (A - \rho B) V] \ . \tag{5.6}$$

The matrices V that reach the above maximum are not unique since any orthogonal transformation of the columns of V will not change the trace. We can select the optimal V to be a set of eigenvectors of the matrix $A - \rho B$. We will denote by $V(\rho)$ a set of the r eigenvectors which reach the above maximum and by $G(\rho)$ the matrix

$$G(\rho) \equiv A - \rho B \ , \tag{5.7}$$

whose n eigenvalues labeled decreasingly are

$$\mu_1(\rho) \geq \mu_2(\rho) \geq \cdots \geq \mu_n(\rho) \ . \tag{5.8}$$

With this notation, it is clear that

$$f(\rho) = \mu_1(\rho) + \mu_2(\rho) + \cdots + \mu_r(\rho) \ . \tag{5.9}$$

The following properties of f can now be proved.

Lemma 5.3 1. *f is a non-increasing function of ρ;*

2. *$f(\rho) = 0$ iff $\rho = \rho_*$.*

Proof: To prove (1) of Lemma 5.3, we need to compare the sums of the r largest eigenvalues of $A - \rho_2 B$ and $A - \rho_1 B$ for $\rho_2 \geq \rho_1$. We have

$$G(\rho_2) - G(\rho_1) = -(\rho_2 - \rho_1) B.$$

Since B is positive semidefinite, classical monotonicity results (see, e.g., [105, p. 396]) show that the r largest eigenvalues of $G(\rho_2)$ will not exceed those of $G(\rho_1)$.

To prove (2) of Lemma 5.3, we start by observing that the sufficient condition is trivial; i.e., according to (5.5), $\rho = \rho_*$ implies $f(\rho) = 0$. Next, since $\mathrm{Tr}\,[V^T B V] > 0$ for any $V \in \mathcal{U}_r$ we can write

$$\mathrm{Tr}\,[V^T A V - \rho V^T B V] < 0 \ \forall \ V \in \ \mathcal{U}_r \quad \text{iff} \quad \frac{\mathrm{Tr}\,[V^T A V]}{\mathrm{Tr}\,[V^T B V]} < \rho \ \forall \ V \in \ \mathcal{U}_r \ .$$

This can be restated as

$$f(\rho) < 0 \quad \text{iff} \quad \rho_* < \rho \ . \tag{5.10}$$

Suppose now that $f(\rho) > 0$ for a certain ρ. Then, there is a V_0 such that

$$\mathrm{Tr}\,[V_0^T A V_0 - \rho V_0^T B V_0] > 0 \ \rightarrow \ \frac{\mathrm{Tr}\,[V_0^T A V_0]}{\mathrm{Tr}\,[V_0^T B V_0]} > \rho.$$

This means that

$$\max_{V \in \mathcal{U}_r} \ \frac{\mathrm{Tr}\,[V^T A V]}{\mathrm{Tr}\,[V^T B V]} > \rho,$$

and therefore $\rho_* > \rho$. This can be restated as

$$f(\rho) > 0 \quad \rightarrow \quad \rho_* > \rho \ . \tag{5.11}$$

Equations (5.10) and (5.11) together, along with the continuity of f, show that $f(\rho) = 0$ implies $\rho = \rho_*$. This completes the proof. $\square$

It is to be noted that the function f is actually strictly decreasing, as will be shown later. This will provide another way to prove the second part of the proposition.

Knowing that the optimal trace ratio can be found as the root of a decreasing function $f(\rho)$, one may ask if it is possible to find an interval where the root lies. Interested readers can find some answers to this question in Section A.1 of Appendix A.

In [5] and [4], algorithms were proposed to solve (5.3) by computing this root. No matter what method is used, it appears at the outset that it will be more complicated to solve (5.3) than (5.2), because the search for the root ρ_* may involve solving several eigenvalue problems instead of just one. However, this does not necessarily mean that it will be more costly. The use of Newton's method combined with the Lanczos procedure will alleviate this search. This is taken up in the next two sections.

5.3.1 The derivative of f

To obtain the derivative of the function f, we first assume that the eigenvalues of $G(\rho)$ are all simple. Then the derivative of each individual eigenvalue $\mu_i(\rho)$ with respect to ρ is explicitly known in terms of the associated eigenvector. When ρ is perturbed to $\rho + \delta$,

the matrix $G(\rho)$ is perturbed by $-\delta B$. The corresponding infinitesimal perturbation to the individual eigenvalue $\mu_i(\rho)$ is then given by (see, e.g., [27, Chapter III, section 3])

$$\mu_i(\rho+\delta) - \mu_i(\rho) = -\delta\langle Bv_i(\rho), v_i(\rho)\rangle,$$

where $v_i(\rho)$ is a unit eigenvector of $G(\rho)$ associated with $\mu_i(\rho)$, and $\langle x, y\rangle$ denotes the inner product of the two vectors x and y. As a result, the derivative of $\mu_i(\rho)$ is simply

$$\mu_i'(\rho) = -\langle Bv_i(\rho), v_i(\rho)\rangle,$$

and this is translated for $f(\rho)$ by

$$f'(\rho) = -\mathrm{Tr}\,[V(\rho)^T B V(\rho)].$$

The extension of this expression to the general case where there may be multiple eigenvalues is doable. Readers can find the analysis in Section A.2 of Appendix A.

5.3.2 Practical implementation via Newton's method

From the expression of the differential of f, Newton's method takes the form

$$\rho_{new} = \rho - \frac{\mathrm{Tr}\,[V(\rho)^T(A-\rho B)V(\rho)]}{-\mathrm{Tr}\,[V(\rho)^T B V(\rho)]} = \frac{\mathrm{Tr}\,[V(\rho)^T A V(\rho)]}{\mathrm{Tr}\,[V(\rho)^T B V(\rho)]}.$$

Remarkably, Newton's method for finding the zero of f amounts to a form of fixed point iteration. The function on the right side of the above equality is

$$g(\rho) = \frac{\mathrm{Tr}\,[V^T(\rho) A V(\rho)]}{\mathrm{Tr}\,[V^T(\rho) B V(\rho)]},$$

in which $V(\rho)$ was defined above. An approach of this type was proposed in the literature and it was observed that convergence is fast [5]. The reason for this is that it is in essence a Newton method.

It is possible to exploit the Lanczos algorithm to provide a highly effective procedure.

Algorithm 5.1 Newton-Lanczos algorithm for Trace Ratio maximization

Input: A, B and a dimension r.

Output: V.

1: Select initial unitary matrix $V \in \mathbb{R}^{n\times r}$ and compute $\rho_0 = \frac{\mathrm{Tr}\,[V^T A V]}{\mathrm{Tr}\,[V^T B V]}$.
2: **for** i = 0,1,2,... **do**
3: Call the Lanczos algorithm to compute the r largest eigenvalues of $G(\rho_i) = A - \rho_i B$ and associated eigenvectors $[v_1, v_2, \cdots, v_r] \equiv V_i$
4: Set $\rho_{i+1} := \frac{\mathrm{Tr}\,[V_i^T A V_i]}{\mathrm{Tr}\,[V_i^T B V_i]}$
5: **end for**

A number of practical refinements can make the above procedure highly effective. The most important of these is based on the observation that variable accuracy techniques can be exploited to reduce cost. Initially, when we are away from the solution, there is no need to compute the eigenspace accurately at all. As we get closer to the solution ρ_*, it becomes essential to tighten the accuracy of the eigenvectors in order for the procedure to enjoy a superlinear convergence. The well-known paper [106] discusses the theory and the practical application of these inexact Newton methods.

5.4 Experiments

This section illustrates the methods discussed in this chapter with applications in dimensionality reduction. Specifically, the projectors V will be computed using two different methods: optimizing the trace ratio using iterative Algorithm 5.1 and solving the generalized eigenvalue problem (5.2). V will then be used to project high dimensional data to low dimensional space. We will first describe our experiments with some synthesis datasets to show the differences between the methods. Then, we will provide experiments on real datasets for face recognition and handwritten digit recognition.

As for the dimensionality reduction techniques leading to the trace ratio optimization problem and its analogues, we consider both a global method, LDA, and a local method which uses kNN graphs for within-class graphs and between-class graphs called local discriminant embedding (LDE) [39]. LDE in its original form uses the generalized eigenvalue problem (5.2) as its optimization.

The notation for the various methods tested is as follows:

- LDA and LDE refer to methods that rely on the eigenvectors of $B^{-1}A$. LDA uses nonlocal matrices and LDE uses local matrices.

- LDA-ITER and LDE-ITER refer to methods which optimize the trace ratio iteratively, using the Newton approach described in section 5.3.2. The matrices A and B are formed in a nonlocal way for LDA-ITER and in a local way for LDE-ITER.

For local methods, within-class graphs and between-class graphs are formed separately using k-nearest neighbors. We take $k = 3$ for within-class graphs and $k = 10$ for between-class graphs. The Gaussian kernel (2.11) is employed to compute similarities between nodes in graphs.

5.4.1 Experiments on synthesis datasets

Figure 5.1 shows the results of projecting a synthesis 3-D dataset to a 2-D space using LDA and LDA-ITER. The dataset consists of 3 classes which have the same small variance along the y and z axes and different large variances along the x axis. They form 3 stripes with different lengths parallel to the x axis. The center of class "+" lies at the origin. The two other class centers lie on the xy plane with equal distances to the x axis and small positive x-coordinates. The projection plane that LDA-ITER found is very close to the yz plane; i.e., it is perpendicular to the x axis. Meanwhile, the projection plane found by LDA deviates a little from this plane toward the xy plane. This is why we see three separated clouds with the same small variance in both directions in the results of LDA-ITER (C) and 3 thin stripes (large variance in one direction and small variance in the other one) in the results of LDA (B). If the class centers lie on the y axis, LDA will yield similar results to LDA-ITER. This shows that LDA is more sensitive to between-class variance. If we increase the between-class variance, the projection plane found by LDA will get closer to the xy plane.

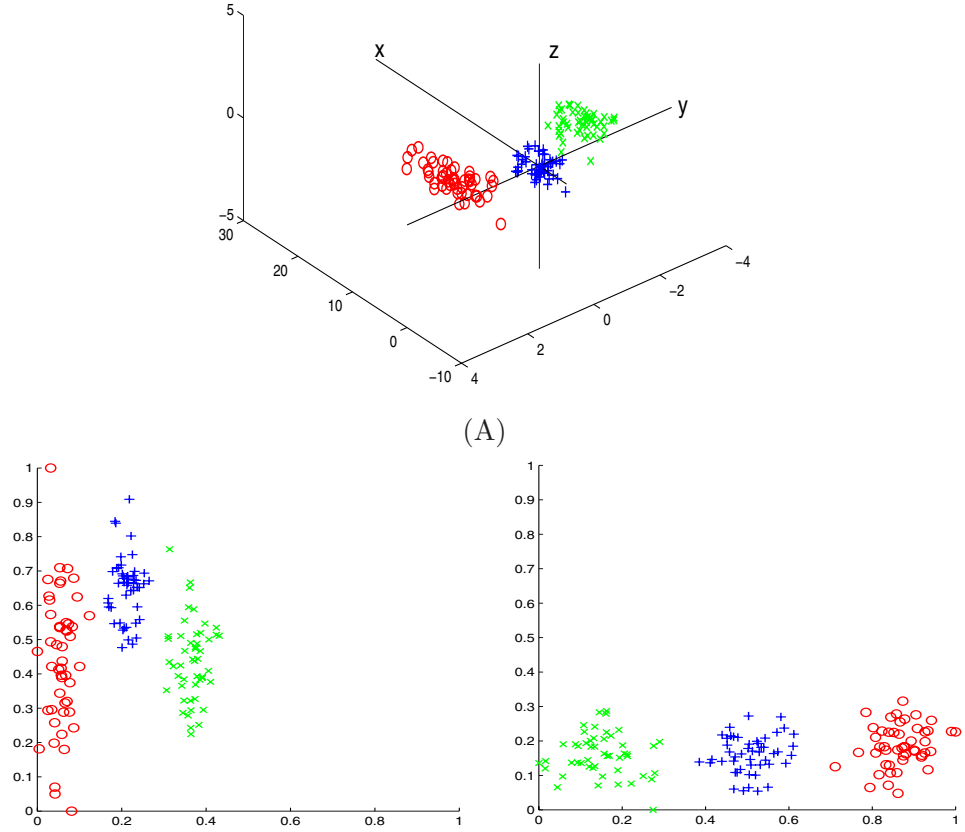


Figure 5.1: Synthesis data 1. (A) original data, (B) projected data by LDA, and (C) projected data by LDA-ITER.

In Figure 5.2, we have 4 classes. The centers of class "o" and class "x" have larger x-coordinates, and at the origin, we have 2 classes; one is a little above the xy plane and one is a little below it. The projection plane found by LDA is very close to the xy plane and makes the two classes at the origin overlap each other. LDA-ITER still manages to find a plane which is perpendicular to the xy plane and well separates the classes.

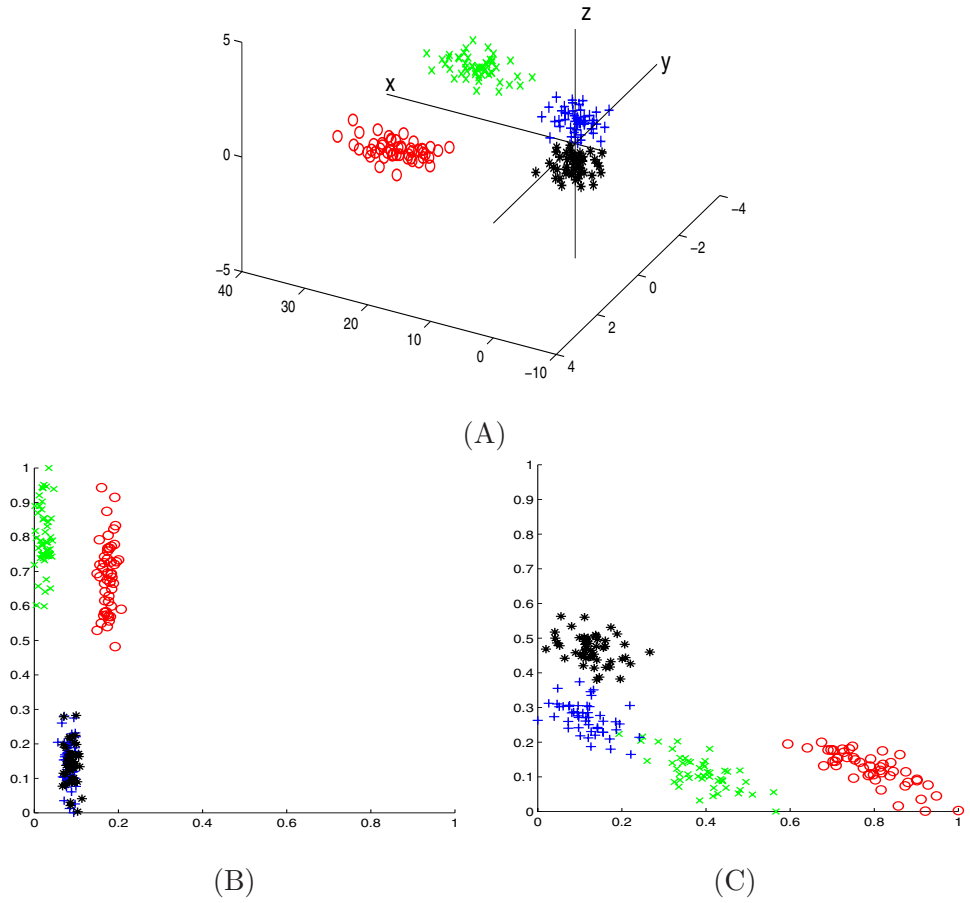


Figure 5.2: Synthesis data 2. (A) original data, (B) projected data by LDA, and (C) projected data by LDA-ITER.

On the other hand, LDA-ITER tends to be more sensitive to within-class variance. In Figure 5.3, the class centers lie on the xy plane. LDA-ITER still yields projection planes which are perpendicular to the x axis and mix up class "*" and class "+". This gives a clue on the poor results given by the iterative method for the PIE dataset discussed later in this section.

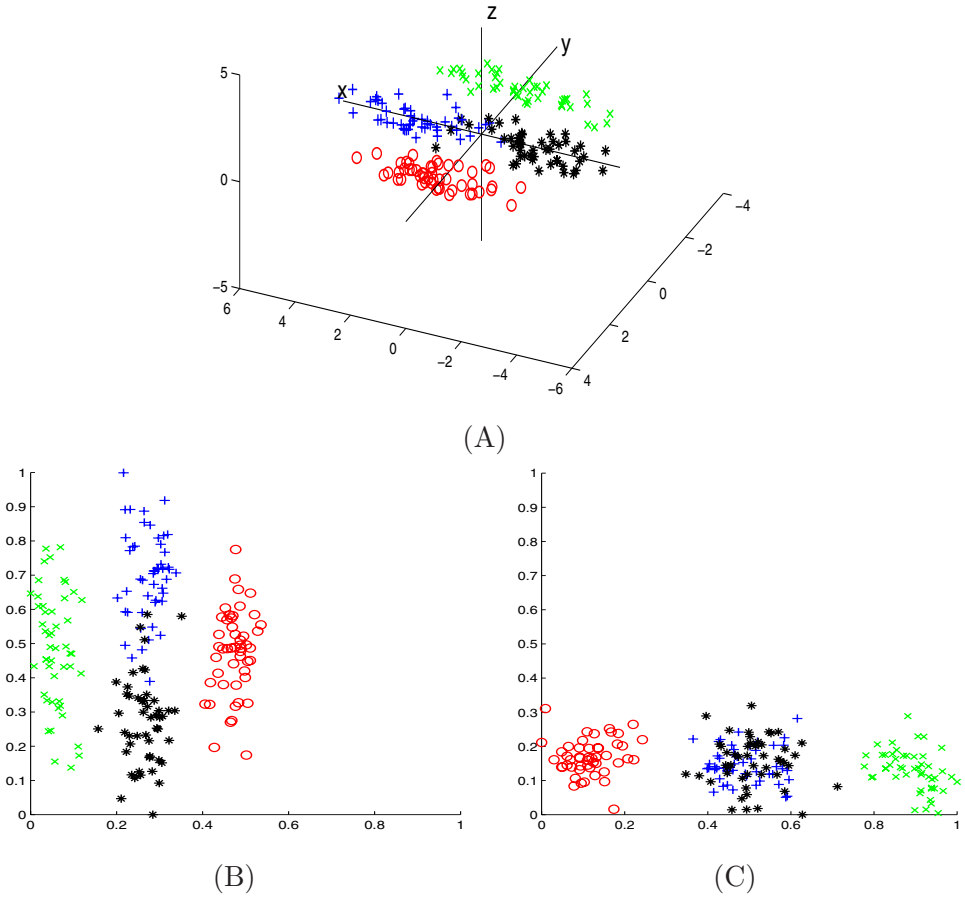


Figure 5.3: Synthesis data 3. (A) original data, (B) projected data by LDA, and (C) projected data by LDA-ITER.

5.4.2 Experiments on real datasets

Experimental setup

The six real datasets that we use are ORL, AR, UMIST, PIE, Essex (found at http://face-rec.org/databases/), and the USPS handwritten character dataset. Each image in ORL, AR, and UMIST is downsampled to 38×31 and is represented as a 1178-dimensional vector. Similarly, each image in PIE is represented as a 1024-dimensional vector. The ORL dataset contains 40 individuals with 10 images for each individual under variation in facial expression and pose. From the AR dataset, we use 126 subjects, each of

which has 8 images taken under different facial expressions and lightning conditions. The UMIST database contains 20 people with different poses and the number of images per person varies from 19 to 48. The PIE dataset contains 68 subjects, each of which has about 170 images. The Essex database contains 4 different sets, from easy to hard—face94, face95, face96, and grimaces. We chose the collection face95 which has 72 individuals, 20 images per individual. Each individual was photographed with a fixed camera, while the subject took one step towards the camera. Each image in the Essex dataset is cropped by removing the top 20 rows and 10 bottom rows and 20 columns on each side and yields a 36×43 image or, equivalently, a 1548-dimensional vector. We use a portion of the USPS dataset which consists of 1100 grayscale images of handwritten digits with 110 images for each digit. Each image is represented as a 256-dimensional vector.

Images of the same subject are divided randomly into training sets and test sets. We perform 5 different random realizations of the training/testing sets and average the errors. The numbers of training samples for ORL, AR, UMIST, PIE, Essex, and USPS are 5, 4, 10, 10, 10, and 20, respectively.

In all experiments, both matrices A and B are scaled to have unit trace before optimization. In addition, a preliminary PCA step using the Lanczos algorithm (see, e.g., [105]) is employed to preprocess these methods to reduce the dimensionality of the data to $n - c$, where n is the number of training samples and c is the number of classes. For all datasets, the obtained B satisfies the existence and uniqueness condition in Proposition 5.2; i.e., it is positive semidefinite with rank greater than $n - r$. We regularize A and B by adding small numbers to their diagonals (10^{-5} of the traces in our experiments). This regularization is not essential for the methods but it does help to ensure a more stable convergence for the iterative method when the eigenvalues of B are close to 0.

Nearest neighbor classifiers (see, e.g., [3, section 4.6]) are employed on the reduced space to classify images into subjects. Classification rates are shown to illustrate the performance of dimensionality reduction.

Results and discussion

Because they optimize the ratio of the traces, the Newton–Lanczos iterative procedure yielded significantly better trace ratios than did non-iterative ones. This is depicted in Table 5.1 for the PIE dataset.

Table 5.1: Values of $\mathrm{Tr}\,[V^T AV]/\mathrm{Tr}\,[V^T BV]$.

Dims	10	20	30	40	50	60
LDE-ITER	32.4648	19.3766	13.6758	11.7107	28.2914	16.9605
LDE	23.5373	13.5477	9.4640	8.0027	20.0822	12.7405

As we argued in earlier sections, LDA-ITER and LDE-ITER converge very fast. In our experiments, they usually take 6–11 iterations to converge to the optimum. One implementation issue we may mention when using Matlab is that often Matlab's `eigs` function drops eigenvalues down to 0 which causes convergence difficulties for the Newton-based iterative methods. We do not expect this to be an issue in a production-type procedure implemented in C, C++, or Fortran.

Figure 5.4 compares LDA-ITER and LDE-ITER against LDA and LDE in terms of recognition rates for different datasets with the dimensions of reduced spaces ranging from 10 to 100. We can see that the LDA-ITER/LDE-ITER outperforms LDA/LDE for most datasets. The improvement gets more significant as the dimension gets bigger.

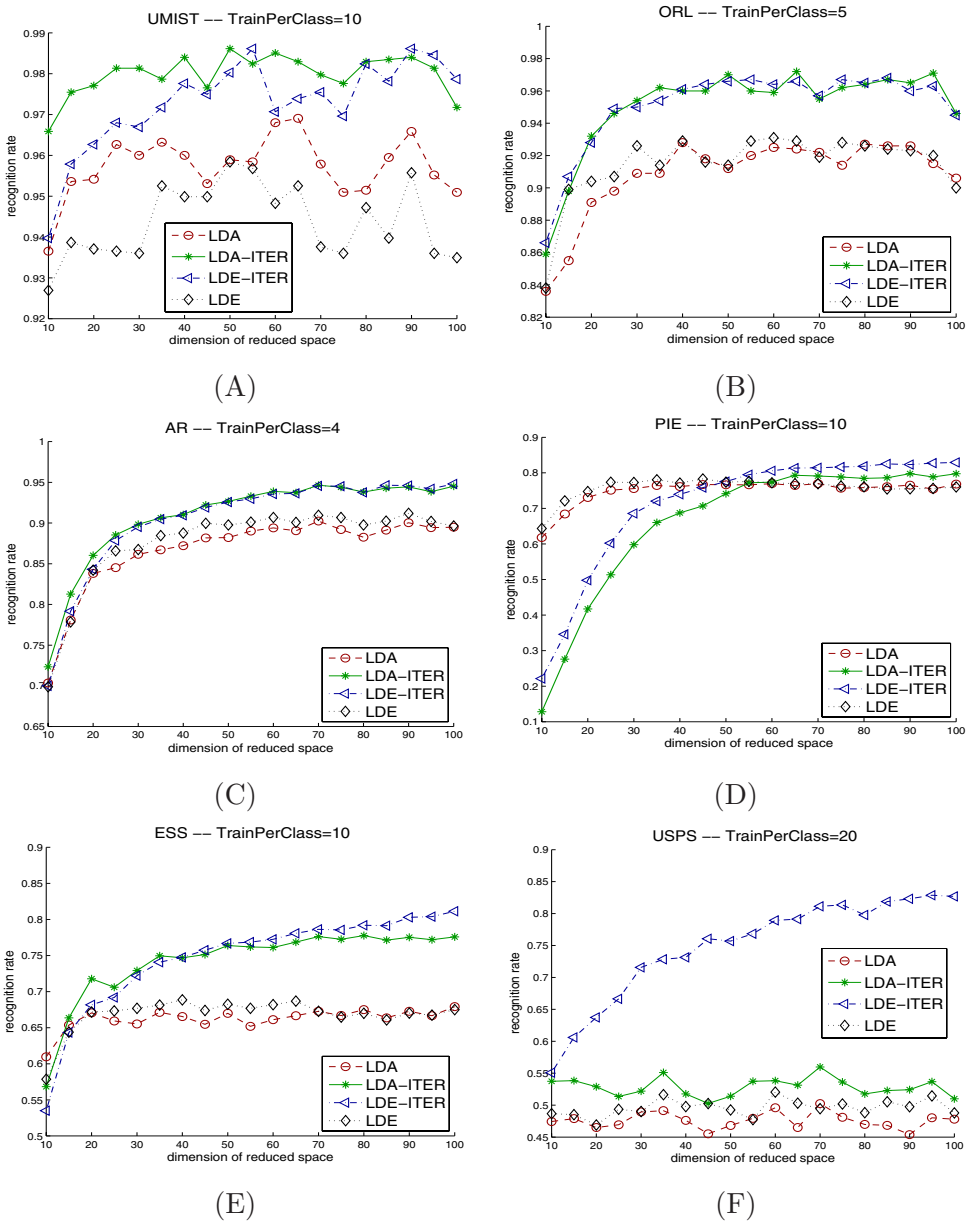


Figure 5.4: Comparison of recognition rates between methods using iterative Algorithm 5.1 and methods relying on the eigenvectors of $B^{-1}A$ for different datasets: (A) UMIST, (B) ORL, (C) AR, (D) PIE, (E) Essex, and (F) USPS. Dimensions range from 10 to 100.

However, for the PIE dataset, the iterative method based on Algorithm 5.1 performed worse than LDA and LDE for low dimensional spaces and started to perform better when $r \geq 50$. This is in spite of the fact that in all experiments with the PIE dataset, the optimal ratios obtained by the trace-ratio based variants LDA-ITR and LDE-ITER are always better than those of their non-ratio-based sibblings LDA and LDE. The reason for this may be the phenomenon we observed with synthesis data in Figure 5.3. Figure 5.5 shows the results of 2-D projection for the PIE dataset using LDA and LDA-ITER. On the left-hand side are projected training data and on the right-hand side are projected testing data. Only 3 random subjects are displayed, but the whole set of 68 subjects give a similar pattern. We can see that trace-ratio based iterative methods tend to minimize inner-class variance and yield a 2-D projection that is almost a 1-D one.

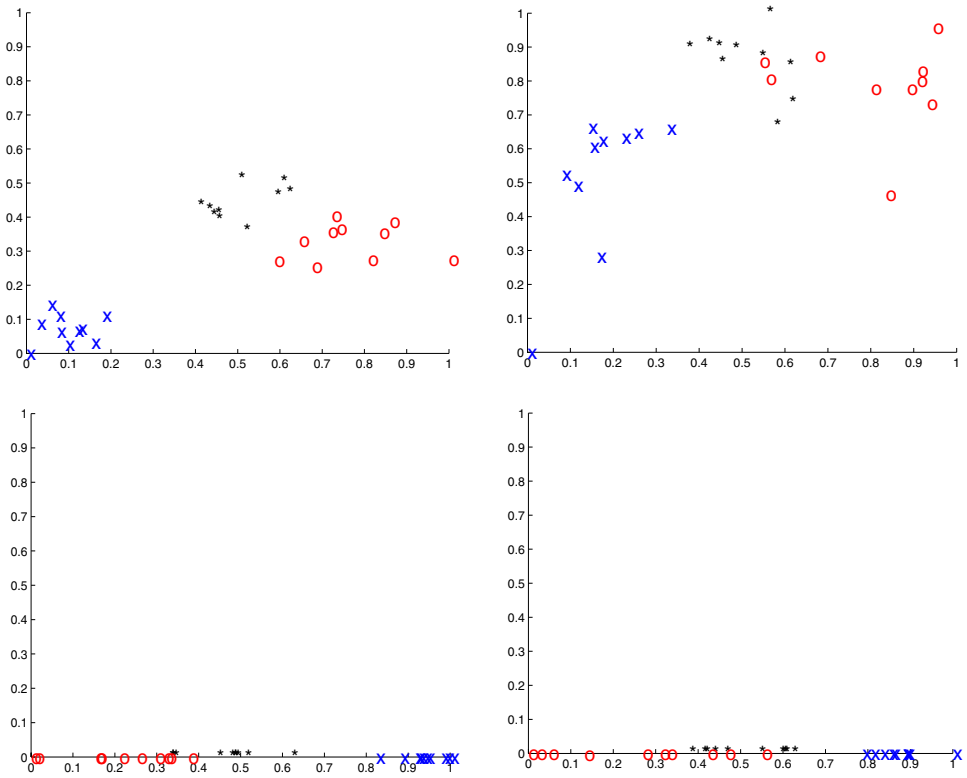


Figure 5.5: 2-D projections of the PIE dataset using LDA (top) and LDA-ITER (bottom). Left side plots show training samples and right side plots show test samples. For LDA-ITER, projected data points almost lie on a 1-D line.

We can also see that for some small datasets such as ORL, the results of LDA and LDE are very similar due to the fact that training sets are small and therefore the local information used in LDE is roughly the same as the global information used in LDA.

5.5 Summary

In this chapter, we have analyzed the problem of maximizing the trace ratio $\mathrm{Tr}\,[V^T AV]/\mathrm{Tr}\,[V^T BV]$ and carried out experiments of a Newton-based method for solving it. We also compared the results obtained by solving this problem with those obtained from its simpler analogues for dimensionality reduction. Our experiments show that with a judicious use of the Lanczos procedure, a good initialization, and inexact

eigenvector calculations in the early stages of the Newton procedure, the overall procedure may be much less expensive than common approaches which rely on solving the generalized eigenvalue problem. The experiments with image recognition also confirm observations made by other researchers that maximizing the trace ratio generally yields better results than solving the generalized eigenvalue problem.

We can see that the trace ratio function only depends on the space spanned by columns of V. Therefore, an interesting future direction is to analyze the trace ratio optimization problem as an optimization on the Grassmann manifold.

Chapter 6

Concluding remarks

We analyzed some low dimensional approximation problems in data analysis and developed efficient algorithms to solve these problems. The solvability of the trace ratio optimization problem has been addressed. Algorithms to find the dominant subspaces of incomplete matrices have been proposed. Apart from unanswered questions raised throughout the thesis [1] , the author would also like to emphasize the importance of studying numerical linear algebra techniques to derive efficient methods for problem of this type.

Finding a low rank approximation of a matrix is to unravel a hidden subspace which satisfies some optimality condition. When it comes to subspaces of matrices, numerical linear algebra provides irreplaceable tools. Classical low rank approximation problems have been studied extensively. New problems arise and require tools from other fields such as convex optimization and probabilistic analysis. It is beneficial to re-investigate traditional methods from different perspectives rather than using these methods as is. New algorithmic aspects and new insights on the structures of the problems will be discovered.

Among other viewpoints, the author is keen on a deeper study on the geometry of low rank matrix manifolds and the intersections between these manifolds and other geometric entities such as the sparse matrix manifold, the nuclear norm ball and the semidefinite cone. As being noted in the literature, the intuition of numerical methods such as subspace iteration and Krylov subspace methods becomes clearer in geometric

[1] Summary sections of Chapter 4 and Chapter 5

terms. The author believes that such study will enable us to attack a broad class of problems in low rank approximation.

References

[1] J. R. Chelikowsky and J. C. Phillips. Quantum-defect theory of heats of formation and structural transition energies of liquid and solid simple metal alloys and compounds. *Phys. Rev. B*, 17:2453–2477, Mar 1978.

[2] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(7):498–520, 1933.

[3] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. Wiley, 2001.

[4] Yue-Fei Guo, Shi-Jin Li, Jing-Yu Yang, Ting-Ting Shu, and Li-De Wu. A generalized Foley-Sammon transform based on generalized Fisher discriminant criterion and its application to face recognition. *Pattern Recogn. Lett.*, 24(1-3):147–158, 2003.

[5] Huan Wang, Shuicheng Yan, Dong Xu, and Xiaoou Tang Huang. Trace ratio vs. ratio trace for dimensionality reduction. In *CVPR '07. IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[6] Feiping Nie, Shiming Xiang, Yangqing Jia, Changshui Zhang, and Shuicheng Yan. Trace ratio criterion for feature selection. In *AAAI*, pages 671–676, 2008.

[7] Chunhua Shen, Hongdong Li, and Michael J. Brooks. Supervised dimensionality reduction via sequential semidefinite programming. *Pattern Recognition*, 41(12):3644–3652, 2008.

[8] Shiming Xiang, Feiping Nie, and Changshui Zhang. Learning a mahalanobis distance metric for data clustering and classification. *Pattern Recognition*, 41(12):3600 – 3612, 2008.

[9] Shuicheng Yan and Xiaoou Tang. Trace quotient problems revisited. In A. Leonardis, H. Bischof, and A. Pinz, editors, *Proceedings of the European Conference on Computer Vision*, volume 2 of *Lecture Notes in Computer Science, Number 3952*, pages 232–244, Berlin-Heidelberg, 2006. Springer Verlag.

[10] Chunhua Shen, Hongdong Li, and Michael J. Brooks. A convex programming approach to the trace quotient problem. In *ACCV (2)*, pages 227–235, 2007.

[11] T. T. Ngo, M. Bellalij, and Y. Saad. The trace ratio optimization problem. *SIAM review*, 54(3):545–569, 2012.

[12] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John T. Riedl. Application of dimensionality reduction in recommender system – a case study. In *IN ACM WEBKDD WORKSHOP*, 2000.

[13] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, August 2009.

[14] E. Candes and T. Tao. The power of convex relaxation: Near-optimal matrix completion, 2009.

[15] Emmanuel Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Commun. ACM*, 55(6):111–119, June 2012.

[16] J-F. Cai, E. J. Candes, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.

[17] Kim-Chuan Toh and Sangwoon Yun. An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems. *Pacific Journal of Optimization*, 6(3):615–640, 2009.

[18] R. H. Keshavan, S. Oh, and A. Montanari. Matrix completion from a few entries. *CoRR*, abs/0901.3150, 2009.

[19] Raghunandan H. Keshavan, Andrea Montanari, and Sewoong Oh. Matrix completion from noisy entries. *Journal of Machine Learning Research*, 11:2057–2078, 2010.

[20] Prateek Jain, Praneeth Netrapalli, and Sujay Sanghavi. Low-rank matrix completion using alternating minimization. *CoRR*, abs/1212.0467, 2012.

[21] Moritz Hardt. On the provable convergence of alternating minimization for matrix completion. *CoRR*, abs/1312.0925, 2013.

[22] G. W. Stewart and Ji guang Sun. *Matrix Perturbation Theory*. Academic Press, 1990.

[23] C. Davis and W. M. Kahan. The rotation of eigenvectors by a perturbation, III. *SIAM J. Numer. Anal.*, 7, March 1970.

[24] Per-Ake Wedin. Perturbation bounds in connection with singularvalue decomposition. *BIT Numerical Mathematics*, 12(1):99–111, 1972.

[25] Michel Ledoux. *The Concentration of Measure Phenomenon*, volume 89 of *Mathematical Surveys and Monographs*. American Mathematical Society, 2001.

[26] Beresford N. Parlett. *The Symmetric Eigenvalue Problem*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1998.

[27] Y. Saad. *Numerical Methods for Large Eigenvalue Problems- classics edition*. SIAM, Philadelpha, PA, 2011.

[28] Moritz Hardt. Robust subspace iteration and privacy-preserving spectral analysis. *CoRR*, abs/1311.2495, 2013.

[29] Van Vu. Singular vectors under random perturbation. *Random Struct. Algorithms*, 39(4):526–538, December 2011.

[30] Sean O'Rourke, Van Vu, and Ke Wang. Matrix perturbation bounds with random noise and their applications. 2013, arXiv/1311.2657.

[31] Fan R. K. Chung. *Spectral Graph Theory (CBMS Regional Conference Series in Mathematics, No. 92)*. American Mathematical Society, December 1996.

[32] Mikhail Belkin and Partha Niyogi. Towards a theoretical foundation for laplacian-based manifold methods. *J. Comput. Syst. Sci.*, 74(8):1289–1308, 2008.

[33] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems 14*, pages 585–591. MIT Press, 2001.

[34] Lawrence K. Saul, Sam T. Roweis, and Yoram Singer. Think globally, fit locally: Unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research*, 4:119–155, 2003.

[35] Peng Jia, Junsong Yin, Xinsheng Huang, and Dewen Hu. Incremental laplacian eigenmaps by preserving adjacent information between data points. *Pattern Recognition Letters*, 30(16):1457–1463, 2009.

[36] Xiaofei He and Partha Niyogi. Locality preserving projections. In *In Advances in Neural Information Processing Systems 16*. MIT Press, 2003.

[37] Effrosyni Kokiopoulou and Yousef Saad. Orthogonal neighborhood preserving projections: A projection-based dimensionality reduction technique. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(12):2143–2156, 2007.

[38] E. Kokiopoulou and Y. Saad. Enhanced graph-based dimensionality reduction with repulsion laplaceans. *Pattern Recogn.*, 42(11):2392–2402, 2009.

[39] Hwann-Tzong Chen, Huang-Wei Chang, and Tyng-Luh Liu. Local discriminant embedding and its variants. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2*, pages 846–853, Washington, DC, USA, 2005. IEEE Computer Society.

[40] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE*, 41(6):391–407, 1990.

[41] P. Praks, J. Dvorsky, and V. Snasel. Latent semantic indexing for image retrieval systems. *Proc. SIAM Conf. on Applied Linear Algebra*, 2003.

[42] Fabrice Souvannavong, Bernard Mrialdo, and Benoit Huet. Improved video content indexing by multiple latent semantic analysis. In *CIVR*, volume 3115 of *Lecture Notes in Computer Science*, pages 483–490. Springer, 2004.

[43] Tamara G. Kolda and Dianne P. O'Leary. A semidiscrete matrix decomposition for latent semantic indexing in information retrieval. *ACM TRANSACTIONS ON INFORMATION SYSTEMS*, 16:322–346, 1998.

[44] Hongyuan Zha and Horst D Simon. On updating problems in latent semantic indexing. *SIAM Journal on Scientific Computing*, 21(2):782–791, 1999.

[45] Matthew Brand. Fast low-rank modifications of the thin singular value decomposition. *Linear Algebra and its Applications*, 415(1):20 – 30, 2006. Special Issue on Large Scale Linear and Nonlinear Eigenvalue Problems.

[46] Eugene Vecharynski and Yousef Saad. Fast updating algorithms for latent semantic indexing. Technical Report ys-2013-5, 2013. arXiv: http://arxiv.org/abs/1310.2008.

[47] Lieven Vandenberghe and Stephen Boyd. Semidefinite programming. *SIAM Review*, 38:49–95, 1994.

[48] Prateek Jain, Praneeth Netrapalli, and Sujay Sanghavi. Low-rank matrix completion using alternating minimization. In *STOC*, pages 665–674, 2013.

[49] Morteza Mardani, Gonzalo Mateos, and Georgios B Giannakis. Distributed nuclear norm minimization for matrix completion. In *Signal Processing Advances in Wireless Communications (SPAWC), 2012 IEEE 13th International Workshop on*, pages 354–358. IEEE, 2012.

[50] Bart Vandereycken. Low-rank matrix completion by riemannian optimization. Technical report, Mathematics Section, Ecole Polytechnique Federale de de Lausanne, 2011.

[51] Nicolas Boumal and P.-A. Absil. Rtrmc: A riemannian trust-region method for low-rank matrix completion. In *NIPS*, 2011.

[52] Thanh T. Ngo and Yousef Saad. Scaled gradients on grassmann manifolds for matrix completion. In *NIPS*, pages 1421–1429, 2012.

[53] Krishna Rajan. Materials informatics. *Materials Today*, 8(10):38 – 45, 2005.

[54] Aaron N. Bloch and Gary Simons. Structural index for elemental solids. *Journal of the American Chemical Society*, 94(24):8611–8613, 1972, http://pubs.acs.org/doi/pdf/10.1021/ja00779a071.

[55] Y. Saad, D. Gao, T. Ngo, S. Bobbitt, J. Chelikowsky, and W. Andreoni. Data mining for materials: Computational experiments with AB compounds. *Phys. Rev. B*, 85(10):104104–13, 2012.

[56] Valeria Simoncini and Daniel B Szyld. Theory of inexact krylov subspace methods and applications to scientific computing. *SIAM Journal on Scientific Computing*, 25(2):454–477, 2003.

[57] G. W. Stewart. *Matrix Algorithms, Volume II: Eigensystems*. SIAM: Society for Industrial and Applied Mathematics, August 2001.

[58] Y. Saad. On the rates of convergence of the Lanczos and the block Lanczos methods. *SIAM J. Numer. Anal.*, 17:687–706, 1980.

[59] Lloyd N. Trefethen and David Bau. *Numerical Linear Algebra*. SIAM, 1997.

[60] Y. Saad. *Iterative Methods for Sparse Linear Systems, 2nd edition*. SIAM, Philadelpha, PA, 2003.

[61] Yurii Nesterov. *Introductory lectures on convex optimization : a basic course*. Applied optimization. Kluwer Academic Publ., Boston, Dordrecht, London, 2004.

[62] Gerard L. G. Sleijpen and Henk A. Van Der Vorst. A jacobi-davidson iteration method for linear eigenvalue problems. *SIAM J. Matrix Anal. Appl*, 17:401–425, 2000.

[63] Franoise Chatelin. Simultaneous newtons iteration for the eigenproblem. In Klaus Bhmer and HansJ. Stetter, editors, *Defect Correction Methods*, volume 5 of *Computing Supplementum*, pages 67–74. Springer Vienna, 1984.

[64] B. Philippe and Y. Saad. On correction equations and domain decomposition for computing invariant subspaces. *Computer Methods in Applied Mechanics and Engineering (special issue devoted to Domain Decomposition)*, 196:1471–1483, 2007.

[65] P. Baldi and K. Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Netw.*, 2(1):53–58, January 1989.

[66] Jorge Nocedal and Stephen J. Wright. *Numerical optimization*. Springer series in operations research and financial engineering. Springer, second edition edition, 2006.

[67] W. Dai, E. Kerman, and Olgica Milenkovic. A geometric approach to low-rank matrix completion. *IEEE Transactions on Information Theory*, 58(1):237–247, 2012.

[68] Prateek Jain, Raghu Meka, and Inderjit S. Dhillon. Guaranteed rank minimization via singular value projection. In *NIPS*, pages 937–945, 2010.

[69] Anastasios T. Kyrillidis and Volkan Cevher. Matrix recipes for hard thresholding methods. *CoRR*, abs/1203.4481, 2012.

[70] Morteza Mardani, Gonzalo Mateos, and Georgios B. Giannakis. Decentralized sparsity-regularized rank minimization: Algorithms and applications. *IEEE Transactions on Signal Processing*, 61(21):5374–5388, 2013.

[71] Samuel Burer and Renato D. C. Monteiro. Local minima and convergence in low-rank semidefinite programming. *Math. Program.*, 103(3):427–444, 2005.

[72] Cho-Jui Hsieh and Peder A. Olsen. Nuclear norm minimization via active subspace selection. In *In Proceedings of the 31st International Conference on Machine Learning (ICML*. ACM, 2014.

[73] Nathan Srebro and Adi Shraibman. Rank, trace-norm and max-norm. In *Proceedings of the 18th Annual Conference on Learning Theory*, pages 545–560. Springer-Verlag, 2005.

[74] Benjamin Recht, Maryam Fazel, and Pablo A. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Rev.*, 52(3):471–501, August 2010.

[75] Ruslan Salakhutdinov and Nathan Srebro. Collaborative filtering in a non-uniform world: Learning with the weighted trace norm. In *NIPS*, pages 2056–2064. Curran Associates, Inc., 2010.

[76] Alan Edelman, Tomas Arias, and Steven T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM J. Matrix Anal. Appl*, 20:303–353, 1998.

[77] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, Princeton, NJ, 2008.

[78] Benjamin Recht. Projected gradient methods, November 2012.

[79] J. B. Rosen. The gradient projection method for nonlinear programming. part ii. nonlinear constraints. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):pp. 514–532, 1961.

[80] 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, http://www.pnas.org/content/106/3/697.full.pdf+html.

[81] N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev.*, 53(2):217–288, May 2011.

[82] Petros Drineas, Ravi Kannan, and Michael W. Mahoney. Fast monte carlo algorithms for matrices i: Approximating matrix multiplication. *SIAM J. Comput.*, 36(1):132–157, 2006.

[83] Petros Drineas, Ravi Kannan, and Michael W. Mahoney. Fast monte carlo algorithms for matrices ii: Computing a low-rank approximation to a matrix. *SIAM J. Comput.*, 36(1):158–183, 2006.

[84] Petros Drineas, Ravi Kannan, and Michael W. Mahoney. Fast monte carlo algorithms for matrices iii: Computing a compressed approximate matrix decomposition. *SIAM J. Comput.*, 36(1):184–206, 2006.

[85] Ravi Kannan and Santosh Vempala. Spectral algorithms. *Foundations and Trends in Theoretical Computer Science*, 4(3-4):157–288, 2009.

[86] B. Recht. A simpler approach to matrix completion. *CoRR*, abs/0910.0651, 2009.

[87] N. Srebro and T. Jaakkola. Weighted low-rank approximations. In *In 20th International Conference on Machine Learning*, pages 720–727. AAAI Press, 2003.

[88] J. D. M. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *In Proceedings of the 22nd International Conference on Machine Learning (ICML*, pages 713–719. ACM, 2005.

[89] Pei Chen and David Suter. Recovering the Missing Components in a Large Noisy Low-Rank Matrix: Application to SFM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8):1051–1063, 2004.

[90] Yonatan Amit, Michael Fink, Nathan Srebro, and Shimon Ullman. Uncovering shared structures in multiclass classification. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 17–24, 2007.

[91] Shiqian Ma, Donald Goldfarb, and Lifeng Chen. Fixed point and bregman iterative methods for matrix rank minimization. *Math. Program.*, 128(1-2):321–353, 2011.

[92] Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *J. Mach. Learn. Res.*, 11:2287–2322, August 2010.

[93] Z. Wen, W. Yin, and Y. Zhang. Solving a low-rank factorization model for matrix completion using a non-linear successive over-relaxation algorithm. In *CAAM Technical Report*. Rice University, 2010.

[94] J. Baglama, D. Calvetti, G. H. Golub, and L. Reichel. Adaptively preconditioned gmres algorithms. *SIAM J. Sci. Comput.*, 20(1):243–269, December 1998.

[95] L. Armijo. Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific Journal of Mathematics*, 16(1):1–3, 1966.

[96] Raghunandan H. Keshavan and Andrea Montanari. Regularization for matrix completion. In *ISIT*, pages 1503–1507, 2010.

[97] Laura Balzano, Robert Nowak, and Benjamin Recht. Online identification and tracking of subspaces from highly incomplete information. In *Proceedings of Allerton*, September 2010.

[98] B. Marlin. Collaborative filtering: A machine learning perspective, 2004.

[99] Bruce M. Irons and Robert C. Tuck. A version of the aitken accelerator for computer iteration. *International Journal for Numerical Methods in Engineering*, 1(3):275–277, 1969.

[100] C. Brezinski. *Projection methods for systems of equations*. Studies in computational mathematics. Elsevier Science, 1997.

[101] Haw ren Fang and Yousef Saad. Two classes of multisecant methods for nonlinear acceleration. 16(3):197–221, 2009.

[102] Avram Sidi. *Practical Extrapolation Methods: Theory and Applications*. Cambridge University Press, New York, NY, USA, 1st edition, 2002.

[103] Sepandar D. Kamvar, Taher H. Haveliwala, Christopher D. Manning, and Gene H. Golub. Extrapolation methods for accelerating pagerank computations. In Gusztv Hencsey, Bebo White, Yih-Farn Robin Chen, Lszl Kovcs, and Steve Lawrence, editors, *WWW*, pages 261–270. ACM, 2003.

[104] Jonathan Barzilai and Jonathan M. Borwein. Two-Point Step Size Gradient Methods. *IMA Journal of Numerical Analysis*, 8(1):141–148, January 1988.

[105] Gene H. Golub and Charles F. Van Loan. *Matrix Computations (3rd Ed.)*. Johns Hopkins University Press, Baltimore, MD, USA, 1996.

[106] R. S. Dembo, S. C. Eisenstat, and T. Steihaug. Inexact newton methods. *SIAM J. Numer. Anal.*, 18(2):400–408, 1982.

[107] Tosio Kato. *Perturbation Theory for Linear Operators*. Classics in Mathematics. U.S. Government Printing Office, 1995.

Appendix A

Extended analysis of the trace ratio optimization problem

A.1 Localization of the optimum of the trace ratio

In this section, we describe some results on the interval where the root of the function $f(\rho)$ defined in (5.6) lies. When A is positive definite, then $f(\rho) \geq 0$ for $\rho = 0$, since $G(0) = A$ (recall that $G(\rho) \equiv A - \rho B$). For $\rho > \lambda_1(A, B)$, we have $f(\rho) < 0$, where $\lambda_1(A, B)$ is the largest generalized eigenvalue of the pencil (A, B). Therefore, the root belongs to the interval $[0,\ \lambda_1(A, B)]$.

A more refined location interval for the root may be found by exploiting Sylvester's inertia theorem. For simplicity we assume that B is positive definite. Let Z be the matrix which diagonalizes the pencil A, B:

$$Z^T A Z = \Lambda, \quad Z^T B Z = I\ .$$

Here the diagonal entries $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ of Λ are the generalized eigenvalues of the pencil A, B. Then,

$$Z^T[A - \rho B]Z = \Lambda - \rho I.$$

According to the Sylvester inertia theorem, the number of negative and positive eigenvalues for the matrices $G(\rho)$ and those of $\Lambda - \rho I$ are the same. Thus for $\rho = \lambda_r$, the first (largest) r eigenvalues of $G(\rho)$ will be nonnegative and so their sum $f(\rho)$ is nonnegative.

On the other side of the spectrum, for $\rho = \lambda_1$ all eigenvalues of $G(\rho)$ will be negative and so $f(\rho) \leq 0$. We have just proved the following proposition.

Proposition A.1 *The root ρ_* of $f(\rho)$ is located in the interval $[\lambda_r,\ \lambda_1]$, where λ_i is the i-th largest eigenvalue of the pair (A, B).*

An alternative to the above bound uses eigenvalues of A and B instead of those of the generalized eigenvalue problem.

Proposition A.2 *Assume that B is positive definite. Then the root ρ_* of $f(\rho)$ is such that*

$$\frac{\sum_{i=1}^{r} \lambda_i(A)}{\sum_{i=1}^{r} \lambda_i(B)} \leq \rho_* \leq \frac{\sum_{i=1}^{r} \lambda_i(A)}{\sum_{i=1}^{r} \lambda_{n-i+1}(B)}, \tag{A.1}$$

where $\lambda_i(A)$ and $\lambda_i(B)$ are the i-th largest eigenvalues of the matrices A and B, respectively.

Proof: Let U be the unitary matrix whose columns are the eigenvectors of A associated with $\lambda_1(A), \ldots, \lambda_r(A)$. Then clearly $\mathrm{Tr}\,[U^T A U] = \lambda_1(A) + \cdots + \lambda_r(A)$, $\mathrm{Tr}\,[U^T B U] \leq \lambda_1(B) + \cdots + \lambda_r(B)$, so

$$\frac{\lambda_1(A) + \cdots + \lambda_r(A)}{\lambda_1(B) + \cdots + \lambda_r(B)} \leq \frac{\mathrm{Tr}\,[U^T A U]}{\mathrm{Tr}\,[U^T B U]} \leq \max_{V^T V = I} \frac{\mathrm{Tr}\,[V^T A V]}{\mathrm{Tr}\,[V^T B V]} = \rho_*.$$

For the right-hand side inequality, we exploit the fact that for any unitary matrix U we have $\mathrm{Tr}\,[U^T A U] \leq \lambda_1(A) + \cdots + \lambda_r(A)$, $\mathrm{Tr}\,[U^T B U] \geq \lambda_n(B) + \lambda_{n-1}(B) + \cdots + \lambda_{n-r+1}(B)$. Hence, for any unitary U,

$$\frac{\mathrm{Tr}\,[U^T A U]}{\mathrm{Tr}\,[U^T B U]} \leq \frac{\lambda_1(A) + \cdots + \lambda_r(A)}{\lambda_n(B) + \lambda_{n-1}(B) + \cdots + \lambda_{n-r+1}(B)},$$

which is therefore an upper bound for ρ_*. □

A.2 Obtaining the derivative of f

In this section, we formulate the derivative of the function f defined in (5.6) in the general case where $G(\rho)$ may have multiple eigenvalues. For this, we will consider the differential of $V(\rho)^T (A - \rho B) V(\rho)$ which is the diagonal matrix of eigenvalues. We need to define the eigenvectors so the mapping $V(\rho)$ is differentiable. The existence of such

an analytic family of orthonormal basis of eigenvectors is discussed in detail in [107, Chapter II, section 6.2] for the general case of analytic perturbations.

In what follows, the notation is simplified: $V(\rho)$, which is assumed to be a differentiable function of ρ, is denoted simply by V. In addition, we assume that V diagonalizes $A - \rho B$ and that we have $(A - \rho B)V = VD$, where D is a diagonal matrix of size $r \times r$ (note that D is a function of ρ).

First, observe that from the equality $V^T V = I$, it follows that

$$0 = \frac{d}{d\rho}[V^T V] = \frac{dV^T}{d\rho}V + V^T\frac{dV}{d\rho} = 0 \quad \rightarrow \quad \text{Diag}\left[V^T\frac{dV}{d\rho}\right] = 0. \tag{A.2}$$

This means that the matrix $V^T\ dV/d\rho$ has a zero diagonal, a property which will be exploited shortly. Next, we proceed with the differentiation of $f(\rho)$. First, consider

$$\begin{aligned}
\frac{d}{d\rho}[V^T(A-\rho B)V] &= \frac{d}{d\rho}[V^T AV] - \frac{d}{d\rho}[V^T\rho BV] \\
&= \frac{dV^T}{d\rho}AV + V^T A\frac{dV}{d\rho} - \frac{dV^T}{d\rho}\rho BV - V^T\left[BV + \rho B\frac{dV}{d\rho}\right] \\
&= \frac{dV^T}{d\rho}[A-\rho B]V + V^T[A-\rho B]\frac{dV}{d\rho} - V^T BV \\
&= \frac{dV^T}{d\rho}VD + DV^T\frac{dV}{d\rho} - V^T BV.
\end{aligned}$$

Now, taking the trace in the above final expression yields

$$\begin{aligned}
\frac{df(\rho)}{d\rho} &= \text{Tr}\left[\frac{dV^T}{d\rho}VD + DV^T\frac{dV}{d\rho} - V^T BV\right] \\
&= 2\ \text{Tr}\left[DV^T\frac{dV}{d\rho}\right] - \text{Tr}\,[V^T BV] \\
&= -\text{Tr}\,[V^T BV].
\end{aligned}$$

The last equality comes from the fact that the matrix $V^T dV/d\rho$ has a zero diagonal as was established above. Therefore, we can state the following result.

Proposition A.3 *The function $f(\rho)$ admits the derivative $-Tr[V(\rho)^T BV(\rho)]$. In particular, under the assumption that B is positive semidefinite with fewer than r zero eigenvalues, f is a strictly decreasing function.*

PROOF: Only the second part remains to be shown, which is a consequence of Lemma 5.1. □

An alternative for deriving the above result is based on the Dunford integral formula. The advantage of this viewpoint is that it bypasses the need to restrict the mapping $V(\rho)$ to being differentiable. If we set $P(\rho) = V(\rho)V(\rho)^T$, then clearly

$$f(\rho) = \mathrm{Tr}\,[V(\rho)^T G(\rho) V(\rho)] = \mathrm{Tr}\,[G(\rho)V(\rho)V(\rho)^T] = \mathrm{Tr}\,[G(\rho)P(\rho)]. \tag{A.3}$$

The Dunford integral for expressing $P(\rho)$ is:

$$P(\rho) = \frac{-1}{2\pi i} \int_\Gamma \left(G(\rho) - zI\right)^{-1}\, dz,$$

where Γ is a Jordan curve containing the r eivenvalues of interest. Note that it is necessary to assume that there are no other eigenvalues in the domain enclosed by Γ, which means that with the labeling of eigenvalues used earlier, $\mu_r(\rho) > \mu_{r+1}(\rho)$. We will denote by $R_\rho(z)$ the resolvant

$$R_\rho(z) = (G(\rho) - zI)^{-1} = (A - \rho B - zI)^{-1}. \tag{A.4}$$

From this we obtain the following expression for $f(\rho)$:

$$\begin{aligned}
f(\rho) &= \frac{-1}{2\pi i}\mathrm{Tr} \int_\Gamma G(\rho)(G(\rho) - zI)^{-1}\, dz && \text{(A.5)}\\
&= \frac{-1}{2\pi i}\mathrm{Tr} \int_\Gamma \left(G(\rho) - zI + zI\right)\left(G(\rho) - zI\right)^{-1}\, dz \\
&= \underbrace{\frac{-1}{2\pi i}\mathrm{Tr} \int_\Gamma dz}_{=0} + \frac{-1}{2\pi i}\mathrm{Tr} \int_\Gamma z(G(\rho) - zI)^{-1}\, dz \\
&= \frac{-1}{2\pi i}\mathrm{Tr} \int_\Gamma z(G(\rho) - zI)^{-1}\, dz. && \text{(A.6)}
\end{aligned}$$

Taking the derivative of $f(\rho)$ from this expression yields

$$\begin{aligned}
f'(\rho) &= \frac{-1}{2\pi i}\mathrm{Tr}\int_\Gamma z\frac{d}{d\rho}R_\rho(z)dz \\
&= \frac{-1}{2\pi i}\mathrm{Tr}\int_\Gamma zR_\rho(z)BR_\rho(z)dz \\
&= \frac{-1}{2\pi i}\int_\Gamma z\mathrm{Tr}\left[R_\rho(z)BR_\rho(z)\right]dz \\
&= \frac{-1}{2\pi i}\int_\Gamma z\mathrm{Tr}\left[R_\rho(z)^2B\right]dz \\
&= \frac{-1}{2\pi i}\int_\Gamma \mathrm{Tr}\left(\left[(A-\rho B)-(A-\rho B-zI)\right]R_\rho(z)^2B\right)dz \\
&= \frac{-1}{2\pi i}\int_\Gamma \mathrm{Tr}\left[(A-\rho B)R_\rho(z)^2B-R_\rho(z)B\right]dz \\
&= \frac{-1}{2\pi i}\mathrm{Tr}\int_\Gamma (A-\rho B)R_\rho(z)^2Bdz-\frac{-1}{2\pi i}\mathrm{Tr}\int_\Gamma R_\rho(z)Bdz \\
&= 0-\mathrm{Tr}\left[P(\rho)B\right].
\end{aligned}$$

The integral in the first term of the above expression is zero because the term $(R_\rho(z))^2$ in the integrand is the exact derivative (with respect to z) of $R_\rho(z)$. The integral in the second bracketed term is just $P(\rho)$. This gives the expression

$$f'(\rho) = -\mathrm{Tr}\left[P(\rho)B\right] = -\mathrm{Tr}\left[V(\rho)V(\rho)^TB\right] = -\mathrm{Tr}\left[V(\rho)^TBV(\rho)\right] .$$

Clearly, $f'(\rho) < 0$ which means f is a stricly decreasing function.