Columbia International Publishing
Journal of Advanced Computing
(2014) Vol. 3 No. 3 pp. 98-222
doi:10.7726/jac.2014.1007
Review

Imputing Missing Entries of a Data Matrix: A review

Achiya Dax

Hydrological Service, P.O.B. 36118, Jerusalem 91360, ISRAEL; E-mail: dax20@water.gov.il

Received 14 August 2014; Published online 22 November 2014

Abstract

The problem of imputing missing entries of a data matrix is easy to state: Some entries of the matrix are unknown and we want to assign "appropriate values" to these entries. The need for solving such problems arises in several applications, ranging from traditional fields to modern ones. Typical examples of traditional fields are statistical analysis of incomplete survey data, business reports, operations management, psychometrika, meteorology and hydrology. Modern applications arise in machine learning, data mining, DNA microarrays data, chemometrics, computer vision, recommender systems and collaborative filtering. The problem is highly interesting and challenging. Many ingenious algorithms have been proposed, and there is vast literature on imputing techniques. Yet, most of the papers consider the imputing problem within the context of a specific application or a specific approach. The current survey provides a broad view of the problem, one that exposes the large variety of imputing methods. Old and new methods are examined with focus on the ideas behind the methods. It is illustrated how simple ideas evolve into highly sophisticated algorithms. The review enables us to identify a number of basic concepts and tools which are shared by several imputing methods. Equivalence theorems that we prove reveal surprising relations between apparently different methods.

Keywords: Imputing; Missing data; Matrix completion problems; Low-rank approximations; Nearest neighbors; Iterative SVD; Least squares methods; Rank minimization; Nuclear norm minimization; Error assessment; Training set; Probe set; Cross-validation; Rank determination

Contents

1 Introduction

Let A be a given $m \times n$ real data matrix. That is, the entries of A are obtained by some real life process, such as physical measurements, experiments, commercial surveys, business reports, etc. The problem discussed in this paper arises when some entries of A are "missing". That is, the values of these entries are unknown. One difficulty in handling such matrices stems from the fact that many models and matrix algorithms require a complete set of data. Thus, even one missing entry may prevent us from using the collected data. In such a case it is often desired to assign "appropriate values" to the missing entries. Once all the missing values are imputed, the matrix can be analyzed and processed by any standard method. The task of imputing missing values is, therefore, to design an algorithm that substitutes appropriate values in the missing entries of A. Other common names of this task are "missing value estimation", "completing missing entries", "matrix completion", "recovering", "reconstructing", etc. The main question is, of course, what is an appropriate value? One feature that makes imputing both difficult and interesting is that there is no ultimate solution to this problem. Indeed, in many cases the answer depends on the origin of the data, on known properties of the data, and the applications intended for the data.

The need for completing missing entries of a data matrix comes up in a broad range of areas. Many algorithms have been proposed and there is vast literature on this issue. The statistical treatment of missing data is a relatively mature area which preceded most of the other fields. See Section 16. Other "traditional" fields include Business Reports, Operations Management, Psychometrika, Hydrology and Meteorology, Biology and Medicine. Detailed references to these applications are given in Table 5.

Recently the problem has emerged in a number of new areas. One such example is completion of DNA microarray data matrices. In these matrices each row presents a different gene, each column refers to a different sample (or a different experimental condition), and the entries in a certain row provide the gene's expression levels in n different samples (or under n different experimental conditions). An important feature of microarray data matrices is that the rows of the matrix can be clustered into groups of similar rows. That is, groups of genes that show similar expression patterns under a variety of experimental conditions. It is this property that enables the identification of certain types of cancer and other diseases. However, the number of clusters, their characterization, and the level of similarity within each cluster, are not known in advance. These questions are answered by applying a "clustering" algorithm that is able to reveal such features. Indeed, clustering algorithms turn out to be an important tool in studying and analyzing microarray data. Most of the clustering algorithms require a complete matrix as input. Yet in practice gene expression data frequently contain missing values, which raises the need to apply an imputing algorithm before starting the clustering analysis. See Table 5 for detailed references on these applications.

Another recent area comes from Computer Vision applications, such as Object Recognition and Structure From Motion (SFM). Consider, for example, the recovery of 3D structure from a video clip. As explained in [117], here the observation matrix collects 2D trajectories of projections of feature points. In real life video clips these projections are not visible along

the entire image sequence, due to occlusion and limited field of view. Thus, the observation matrix has missing entries. The recovery of these entries is often achieved by computing a low-rank matrix that best matches the known entries. For detailed discussion of Computer Vision and SFM applications that require matrix completion see [44, 117, 134, 135, 264] and the references in Table 5.

The field of Recommender Systems and Collaborative Filtering is, perhaps, the most celebrated example of a new field which requires the solution of huge imputing problems. The name collaborative filtering refers to methods used by recommender systems. See Table 5 for references. The public interest in this field has been raised by the Netflix prize competition [20, 92, 203, 256]. The Netflix matrix provides the data for a movie recommendation system. The rows correspond to viewers, the columns to movies, and the (i, j) entry (if known) provides the rating of viewer i to movie j. There are about 480,000 viewers and 18,000 movies, where the average number of movies rated by one viewer is about 200. The known rates are integers in the range 1 to 5, but the computed rates are not forced to be in a certain range. The task is to compute the unknown entries of the matrix. The organizers of the competition are reserving another 3 million ratings as a "test set", which is unknown to the public. The test set is used to define a target function, based on an RMSE criterion.

Another well known matrix completion problem is related to the Jester matrix [105, 180, 257, 261] which contains rates of jokes. Similar commercial recommendation systems are used by MovieLens [49, 198, 257, 261], Tivo [7], Amazon [171], CDnow (www.cdnow.com), Barnes and Noble (www.barnesandnoble.com), and Apple. Further examples of new fields which require matrix completion are Chemometrics, Data Mining, Information Retrieval and Machine Learning. See Table 5 for detailed references.

However, this paper is not about applications. It is about imputing methods, and the motivation behind these methods. Although there is a huge amount of literature on imputing methods, most of the papers consider the problem within the context of a specific application or a specific approach. In the statistical literature there is a number of review papers and books on analysis of incomplete data, e.g., [10, 64, 173, 230, 236]. But outside the statistical world it is difficult to find such works. The coming survey provides a broad view of the problem, one that exposes the large variety of imputing methods. The focus is on the ideas behind the methods, the ways these ideas are converted into matrix algorithms, and the properties that characterize each method.

The paper is intended to serve a broad spectrum of readers. On one end it is written for readers who are not familiar with the imputing problem. An attempt is made, therefore, to give a clear exposition of the methods, and a self-contained presentation of the necessary background theory. For this purpose we derive several new proofs which are based on elementary linear algebra. On the other end of the spectrum the review might be helpful to expert readers, who conduct research in a specific imputing area and want to compare their approach with existing works in other "communities". (An inspection of the literature shows that the field of imputing methods is divided into several "communities", where each community concentrates on a certain application, or a certain solution approach, and uses a different terminology.) Thus another aim of the paper is to put the main ideas and methods on one plat-

form (the numerical linear algebra platform) and to see how the basic ideas move between the communities. To reach this goal we derive equivalence theorems that reveal surprising links between apparently different methods. Other readers who might benefit from the paper are practitioners who want to decide which algorithm to use on their data. The paper presents a large variety of imputing methods, where for each algorithm it considers the type of problems it intended to solve, the main computational steps, and related references. The final choice depends on the nature of the data at hand, the size of the matrix, the percentage of missing entries, and so forth. Thus, although we are unable to give specific recommendations, the paper provides some useful guidelines for choosing an algorithm.

It is assumed throughout the paper that $A = (a_{ij})$ is a real $m \times n$ matrix, that some entries of A are missing, and that $m \geq n$. (Otherwise, when A has more columns than rows, replace A with A^T.) As we shall see, several methods achieve imputing by constructing a low-rank approximation of A. Let k denote the rank of the desired approximation, and let

$$\mathbb{B}_k = \{B | B \in \mathbb{R}^{m \times n},\ \text{rank}(B) \leq k\}$$

denote the set of all real $m \times n$ matrices, $B = (b_{ij})$, whose rank is at most k. Then often the computed rank-k approximation attempts to solve a least squares problem of the form

$$\begin{aligned} &\text{minimize} \quad F(B) = \sum_{(i,j)\in\Omega} (a_{ij} - b_{ij})^2 \\ &\text{subject to} \quad B \in \mathbb{B}_k, \end{aligned} \tag{1.1}$$

where

$$\Omega = \{(i,j) \,\big|\, a_{ij} \ \text{ is known } \}.$$

That is, the sum in (1.1) is restricted to known entries of A. Let B_k^* denote a computed solution of (1.1). Then the missing entries of A attain the values of the corresponding entries in B_k^*.

Let the matrix operator P_Ω be defined in the following way. Given two matrices $Y = (y_{ij}) \in \mathbb{R}^{m \times n}$ and $Z = (z_{ij}) \in \mathbb{R}^{m \times n}$, the equality $Z = P_\Omega(Y)$ implies

$$z_{ij} = y_{ij} \ \text{ when } (i,j) \in \Omega \quad \text{and} \quad z_{ij} = 0 \quad \text{when} \quad (i,j) \notin \Omega.$$

With these notations at hand (1.1) can be rewritten as

$$\begin{aligned} &\text{minimize} \quad F(B) = \|P_\Omega(A) - P_\Omega(B)\|_F^2 \\ &\text{subject to} \ \ B \in \mathbb{B}_k, \end{aligned} \tag{1.2}$$

where $\|\cdot\|_F$ denotes the Frobenius matrix norm. For the sake of clarity we mention that the Frobenius norm of a matrix $Y = (y_{ij}) \in \mathbb{R}^{m \times n}$ is defined as

$$\|Y\|_F = \Big(\sum_{i=1}^{m} \sum_{j=1}^{n} (y_{ij})^2 \Big)^{1/2}.$$

One motivation behind the above approach comes from the following observations. If A has no missing entries then (1.1) coincides with the Eckart--Young minimum norm problem

$$\begin{aligned} &\text{minimize} \quad F(B) = \|A - B\|_F^2 \\ &\text{subject to} \ \ B \in \mathbb{B}_k, \end{aligned} \tag{1.3}$$

In this case, when all the entries of A are known, the solution is given in terms of the Singular Value Decomposition (SVD) of A. Let $T_k(A)$ denote a rank-k Truncated SVD of A, then $T_k(A)$ solves (1.3). See, for example, [23, 58, 71, 253]. A detailed definition of $T_k(A)$ is given in Section 5.

However, when A has missing entries, standard SVD algorithms are not applicable. In fact, in some applications the ultimate goal of the completion process is to produce a (Truncated) SVD of A. Let A^* denote the completed matrix in which the missing entries of A equal the corresponding entries of B_k^*. Then, clearly, B_k^* is a rank-k Truncated SVD of A^*.

Similar remarks apply to the question of how to carry out a Principal Components Analysis (PCA), or a Factor Analysis, of a data matrix with missing entries. Again, in some cases the final aim of the imputing process is to enable the desired analysis. For detailed discussions of these issues see the references in Table 5.

The low-rank approximation approach raises two crucial issues. First, we need effective algorithms for solving (1.1). Second, we need to find a suitable value for k. The first question is addressed in Section 8, in which we present several minimization methods for solving (1.1). The second problem is closely related to the problem of determining the rank of a matrix, e.g., [23, 110, 253]. It is also related to the question of how to determine the number of principle components (principle factors) of a matrix, e.g., [87, 132, 164, 297]. The last problem has attracted a lot of attention in the PCA literature, and several solution methods have been proposed. For example, the surveys in [133] and [215] examine over 20 "stopping rules" for determining the number of significant principal components. Yet, when A has missing entries we face a further difficulty: Writing B in a factord form shows that we have to solve a system of ν nonlinear equations, where ν denotes the number of known entries in A. On the other hand, the number of "degrees of freedom" that we have in the choice of a rank-k matrix $B \in \mathbb{B}_k$ equals $k(m + n - k)$. (These claims are explained in Sections 9 and 10.) Thus, as k increases beyond a certain threshold, the number of "degrees of freedom" in (1.1) exceeds the number of equations, and the solution of (1.1) loses its uniqueness. Then the computed solution matrix, B_k^*, is free to move away from the "true" rank-k TSVD of A, and the quality of the imputed values starts to deteriorate. One aim of the paper is to focus attention on this difficulty and to outline some possible remedies.

Another related issue regards the possibility to achieve "exact recovery". Assume for a moment that the known entries of A are "sampled" from those of a given (known) matrix $\hat{A} \in \mathbb{R}^{m\times n}$. If the imputed values of the missing entries in A attain their "true" values then we say that we have "exact recovery". The challenge of recovering the whole matrix from a small sample of observed entries has attracted the attention of several researchers. Recently Candès and Recht [38] have proved that under certain conditions on A and $\hat{A}$ solving the

nuclear norm problem (10.1) is likely to result in exact recovery. See Section 10. This theoretical breakthrough has initiated a burst of papers that attempt to harness the nuclear norm approach for solving large "Netflix-like" problems. The nuclear norm of a matrix is the sum of its singular values and direct minimization of this sum is rather complicated. For this reason most methods reformulate (10.1) in alternative ways that allow efficient solution methods. To address this issue we provide a systematic derivation of equivalent formulations of the nuclear norm problem. (As a rule, when mentioning known results we add references. New results are stated and proved without references.)

The paper is divided into three parts. The first one provides an overview of the basic imputing approaches. Starting from averaging methods we describe some well-known imputing algorithms, including k Nearest Neighbors (KNN) impute, Iterative Column Regression, and Iterative SVD. Then we move to consider recently proposed methods: Minimizing the singular values "tail" (FRAA), least squares methods, Riemannian optimization, rank minimization, and nuclear norm minimization.

The second part considers a number of closely related problems and methods. It starts with examples of special cases, as the Euclidean Distance matrix completion problem. Then it gives examples of extended problems, such as missing values in tensors. It ends with discussions of imputing methods that come from different disciplines: Biological methods and Statistical methods. Special attention is given to differences between the Statistical approach and the current numerical linear algebra approach.

The third part of the paper is dedicated to the question of how to assess the quality of the imputed entries. In real life problems the true values of the missing entries remain unknown and implicit methods are used to answer this question. Following the statistical "cross-validation" idea, the set of known entries is split into a "training set" and a "probe set". Then the imputing algorithm runs on the training set and tested on the probe set. It is shown that this approach can be used to determine an optimal matrix rank, k, when solving (1.1). Finally we add some guidelines and tables that might be helpful when choosing an imputing algorithm for solving a specific problem.

Part I: Basic approaches

The next sections describe some typical imputing algorithms and explain their motivation. The review is not comprehensive. (The field is so huge that it is difficult to cover all the existing approaches.) It concentrates on methods that can be described by standard concepts and techniques from the fields of numerical linear algebra and optimization. Methods that are mainly based on different disciplines, such as statistical or biological methods, are deferred to Part II.

2 Averaging methods

Let ν denote the number of known entries in A. Then the mean value of the known entries is

$$\alpha = \Big(\sum_{(i,j)\in\Omega} a_{ij} \Big) / \nu. \tag{2.1}$$

The simplest way of imputing is, perhaps, substituting every missing entry of A by α. This "mean imputing" method is useful for handling "random" matrices, whose entries are considered as "random numbers" that obey the same probabilistic distribution. In this case substituting a missing entry by the corresponding mean (or median) seems to be a reasonable choice. The disadvantage of this method (from a statistical point of view) is that the variance of the data after the imputing is smaller than the "true" variance.

The basic mean imputing method can be modified in a number of ways. In "row-averaging" the missing entries in a certain row attain the mean value of the known entries in this row. Here there is tacit assumption that all the entries in one row obey the same distribution function. In "column-averaging" the missing entries in a certain column attain the mean value of the known entries in this column. Subtracting the computed row average from the (known) entries of the row is called "row-centering". After performing row-centering the sum of (known) entries in each row equals zero. The methods mentioned above are often combined into an "additive model" of the form

$$a_{ij} = \alpha + x_i + y_j + \varepsilon_{ij}, \quad i = 1, \ldots, m,\ j = 1, \ldots, n, \tag{2.2}$$

where ε_{ij} denotes the model error. The interpretation of this model depends on the origin of the data. In monthly collected data $n = 12$, each column refers to a different month, and each row refers to a different year. In this case $x_1, \ldots, x_m$ reflect the annual trend, and $y_1, \ldots, y_{12}$, reflect the seasonal behavior of the data, e.g., [53, 90]. A second way to interpret (2.2) occurs in annually collected data, such as annual rainfall in a group of neighboring monitoring stations ("rain-gauges"). Here each column refers to a different monitoring station, and each row refers to a different year. In this example α denotes the average amount of annual rainfall in the area, $\alpha + x_i$ gives the average amount of rainfall during the ith year, and y_j expresses the deviation of the jth station from the annual average.

The Netflix data matrix provides another interpretation of (2.2). Here α denotes the average rating of a movie; x_i expresses the tendency of the ith user to rate higher (or lower) than others; and y_j denotes the "quality" of the jth movie, expressing its tendency to be rated higher (or lower) than other movies, e.g., [19, 151, 152].

The parameters of an additive model are easily found by solving the problem

$$\text{minimize} \quad f(\alpha, x_1, \ldots, x_m, y_1, \ldots, y_n) = \sum_{(i,j)\in\Omega} (a_{ij} - \alpha - x_i - y_j)^2. \tag{2.3}$$

See [53] for details. Let the parameters $\tilde{\alpha}, \tilde{x}_1, \ldots, \tilde{x}_m,\ \tilde{y}_1, \ldots, \tilde{y}_n$ solve the last problem, then the (i, j) missing entry attains the value $\tilde{\alpha} + \tilde{x}_i + \tilde{y}_j$. Moreover, the solution of (2.3) enables us

to convert the data matrix, A, into a matrix which is both row-centered and column-centered. For this purpose define $\tilde{A} = (\tilde{a}_{ij}) \in \mathbb{R}^{m\times n}$ to be the related residual matrix. That is, $\tilde{a}_{ij} = a_{ij} - \tilde{\alpha} - \tilde{x}_i - \tilde{y}_j$ when $(i,j) \in \Omega$, and $\tilde{a}_{ij} = 0$ otherwise. Then, clearly, $\tilde{A}$ is both "row-centered" and "column-centered".

In annually collected data, such as annual rainfall in neighboring monitoring stations, one may prefer a "multiplicative" model of the form

$$a_{ij} = x_i y_j + \varepsilon_{ij}, \quad i = 1,\ldots,m, \;\; j = 1,\ldots,n. \tag{2.4}$$

Here x_i denote the annual average of rainfall at the ith year, while y_j denotes the relative intensity of rain in the jth station. The parameters of this model are found by solving the problem

$$\text{minimize} \quad f(x_1,\ldots,x_m,y_1,\ldots,y_n) = \sum_{(i,j)\in\Omega} (a_{ij} - x_i y_j)^2, \tag{2.5}$$

whose solution is discussed in Section 8.1.

Let $\mathbf{x} = (x_1,\ldots,x_m)^T \in \mathbb{R}^m$ and $\mathbf{y} = (y_1,\ldots,y_n)^T \in \mathbb{R}^n$ denote the corresponding vectors of unknowns. Let $\mathbf{e} = (1,1,\ldots,1)^T$ denote a vector of ones whose length is either m or n, depending on the context. Then the five averaging schemes mentioned above are essentially low-rank approximations of A regarding the matrices $\alpha\mathbf{e}\mathbf{e}^T, \mathbf{x}\mathbf{e}^T, \mathbf{e}\mathbf{y}^T, \alpha\mathbf{e}\mathbf{e}^T + \mathbf{x}\mathbf{e}^T + \mathbf{e}\mathbf{y}^T$ and $\mathbf{x}\mathbf{y}^T$, respectively. This suggests that higher rank approximations are likely to produce smaller model error. Indeed, for example, in the Netflix problem the basic model (2.2) is extended to higher rank approximations, e.g., [19, 151, 152, 212].

Averaging schemes take part in several imputing methods. In particular, iterative methods which require a good starting point. The use of averaging is often done by "centering" the data matrix before starting the iterative process. Then setting zeros instead of the missing entries provides a reasonable starting point, e.g., [94, 125, 269].

3 Nearest Neighbors imputing

Assume for a moment that we have a pair of columns (or rows) that show "similar behavior". In this case it is possible to impute the missing entries in one column by using the corresponding entries of the other column (if available). In statistical literature this approach is known as "Hot-Deck Imputation". The name comes from the use of computer punch cards for data storage. Then a missing entry in a data card was replaced by the corresponding entry from another "similar" card. Here the similar cards are called "donors" instead of "neighbors". In "hot-deck" imputation the donor's cards are searched from the same recent survey as the imputed card, while in "cold-deck" imputation the donors are taken from previous surveys. For recent discussion of statistical hot-deck methods see [15]. To implement this idea into a matrix algorithm we need to answer a number of questions. The first one is how to quantify the "similarity" between two data vectors.

3.1 Measuring similarity

Let $\mathbf{u} = (u_1, u_2, \ldots, u_m)^T$ and $\mathbf{v} = (v_1, v_2, \ldots, v_m)^T$ be two m-vectors that provide the records of two columns. Assume first, for the sake of simplicity, that all the entries of these vectors are known. A typical way for measuring the similarity between the two vectors is based on the following three-step procedure. The first step is to "centralize" the two vectors. The "centering" of a vector is carried out by computing the mean value of its entries and subtracting the mean from each entry. In the second step the corresponding centered vectors, $\mathbf{u}_c$ and $\mathbf{v}_c$, are normalized to have unit Euclidean length. Let $\mathbf{u}_n$ and $\mathbf{v}_n$ denote the resulting unit vectors. In the third step we measure the Euclidean distance between $\mathbf{u}_n$ and $\mathbf{v}_n$. Then a smaller distance implies better similarity. For the sake of clarity we mention that the Euclidean norm of an m-vector, $\mathbf{y} = (y_1, \ldots, y_m)^T$, is defined as

$$\|\mathbf{y}\|_2 = (\mathbf{y}^T\mathbf{y})^{1/2} = (y_1^2 + \cdots + y_m^2)^{1/2},$$

while the inner product $\mathbf{u}^T\mathbf{v}$ is defined as

$$\mathbf{u}^T\mathbf{v} = u_1v_1 + \cdots + u_mv_m.$$

Hence the Euclidean distance between $\mathbf{u}_n$ and $\mathbf{v}_n$ satisfies

$$\|\mathbf{u}_n - \mathbf{v}_n\|_2^2 = \mathbf{u}_n^T\mathbf{u}_n - 2\mathbf{u}_n^T\mathbf{v}_n + \mathbf{v}_n^T\mathbf{v}_n = 2 * (1 - \mathbf{u}_n^T\mathbf{v}_n).$$

Recall that the Pearson correlation coefficient between $\mathbf{u}$ and $\mathbf{v}$ is defined as

$$\pi(\mathbf{u}, \mathbf{v}) = \mathbf{u}_n^T\mathbf{v}_n = (\mathbf{u}_c^T\mathbf{v}_c)/(\|\mathbf{u}_c\|_2\|\mathbf{v}_c\|_2). \tag{3.1}$$

That is, $\pi(\mathbf{u}, \mathbf{v})$ gives the cosine of the angle between $\mathbf{u}_n$ and $\mathbf{v}_n$. The value of $\pi(\mathbf{u}, \mathbf{v})$ lies between -1 and 1, and as $\pi(\mathbf{u}, \mathbf{v})$ increases the distance $\|\mathbf{u}_n - \mathbf{v}_n\|_2$ decreases. So a value of $\pi(\mathbf{u}, \mathbf{v})$ close to 1 means good similarity.

Assume further that we have a list of vectors, $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_\ell$, and we want to assess their similarity to a given vector, $\mathbf{b}$. This task can be achieved by computing the related Pearson coefficients $\pi(\mathbf{b}, \mathbf{c}_j)$, $j = 1, \ldots, \ell$. Then, clearly, a large value of $\pi(\mathbf{b}, \mathbf{c}_j)$ means good similarity. The last method has a number of variants. For example, let $\mathbf{r}_j, j = 1, \ldots, \ell$, denote the computed residual vector of the linear least squares problem

$$\text{minimize} f_j(\alpha, \beta) = \|\alpha\mathbf{c}_j + \beta\mathbf{e} - \mathbf{b}\|_2^2, \tag{3.2}$$

where $\mathbf{e} = (1, 1, \ldots, 1)^T \in \mathbb{R}^m$. Then $\|\mathbf{r}_j\|_2^2$ reflects the ability of $\mathbf{c}_j$ to estimate $\mathbf{b}$ (after centering the two vectors). Now it is easy to verify that the similarity relations which are induced by inspecting the computed residual norms, $\|\mathbf{r}_j\|_2^2, j = 1, \ldots, \ell$, are equivalent to those derived by inspecting the related Pearson coefficients $\pi(\mathbf{b}, \mathbf{c}_j), j = 1, \ldots, \ell$. That is,

$$\|\mathbf{r}_i\|_2 \geq \|\mathbf{r}_j\|_2 \text{ if and only if } |\pi(\mathbf{b}, \mathbf{c}_i)| \leq |\pi(\mathbf{b}, \mathbf{c}_j)|. \tag{3.3}$$

Let us turn now to consider the situation when **u** and **v** have some missing entries. The simplest option is to delete all the rows in which **u** or **v** have missing entries. This results in a pair of "shorter" vectors without missing entries, $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{v}}$, and this pair of vectors is used for testing the similarity, e.g., [24] and [147]. In some cases it is helpful to take into account the number of known entries that are "shared" by the two vectors. The idea is that a Pearson coefficient which is based on a smaller number of entries is less reliable. So the modified correlation coefficient has the form

$$\tilde{\pi}(\mathbf{u}, \mathbf{v}) = (\tilde{m}/(\tilde{m} + \delta))\pi(\tilde{\mathbf{u}}, \tilde{\mathbf{v}}) \tag{3.4}$$

where $\tilde{m}$ is the number of entries in the "shorter" vectors, $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{v}}$, and δ is a given positive integer. A typical value of δ, in the Netflix context, is 100. See [151] and [189].

Another common practice is to substitute some trial values instead of the missing values (such as column averages) and then to test similarity. Further modifications of this approach are described in [151] and [235].

3.2 Best Neighbor imputing

This method achieves one sweep over the columns of A, considering one column at a time. The basic idea is to replace the missing entries in one column with the corresponding entries of a "similar" column.

Treating the jth column, $j = 1, \ldots, n$.

a) Let the m-vector **b** denote the jth column of A. If **b** has no missing entries skip to the next column.
b) Select $\mathbf{c}_1$ to be another column of A which is most similar to **b**.
c) Replace the missing entries in **b** with those of the vector $\alpha_1\mathbf{c}_1 + \beta_1\mathbf{e}$, where α_1 and β_1 solve (3.2) for $\mathbf{c}_1$.

At the end of the last step it is still possible that some missing entries of **b** have not been imputed. This happens at rows in which both **b** and $\mathbf{c}_1$ have missing entries. In this case one selects a second column, $\mathbf{c}_2$, which is the "second best" substitute of **b**. Then the remaining missing entries are determined by repeating step c) with $\mathbf{c}_2$ instead of $\mathbf{c}_1$. The process repeats in this way until all the missing entries of **b** are imputed. Variants of this method are considered in [86], [143], and [281].

3.3 The basic K Nearest Neighbors (KNN) algorithm

This popular algorithm improves the former one by using k "neighbors" instead of one. A typical number of neighbors lies between 4 and 50. The algorithm performs a different search for k neighbors for any missing entry. Let the matrix $\tilde{A} = (\tilde{a}_{ij}) \in \mathbb{R}^{m\times n}$ be obtained from A by centering and normalizing the columns of A. So $\tilde{A}$ has the same pattern of missing entries as

A. To simplify the algorithm A is replaced by $\tilde{A}$. (Once the missing entries of $\tilde{A}$ are computed, the missing entries of A are easily retrieved.) Assume that the (i,j) entry of $\tilde{A}$ is unknown. Then $\tilde{a}_{ij}$ is computed in the following way.

a) Let $\mathbf{b} \in \mathbb{R}^m$ denote the jth column of $\tilde{A}$. Select k columns of $\tilde{A}$ which are most similar to **b**. The search is restricted to columns whose ith entry is known. It is also restricted to columns that have positive correlation with **b**.

b) Let $\mathbf{c}_1, \ldots, \mathbf{c}_k$, denote the selected k columns of $\tilde{A}$. Let $\psi_\ell, \ell = 1, \ldots, k$, denote the related similarity coefficients, and let η_ℓ denote the ith entry of $\mathbf{c}_\ell, \ell = 1, \ldots, k$. Then the (i,j) entry of $\tilde{A}$ is defined as

$$\tilde{a}_{ij} = (\sum_{\ell=1}^{k} \psi_\ell \eta_\ell)/(\sum_{\ell=1}^{k} \psi_\ell). \tag{3.5}$$

That is, $\tilde{a}_{ij}$ is a weighted average of the related entries in the selected neighboring columns. The larger similarity we have the larger the weight.

This simple idea has many versions. In fact, it is difficult to find two papers that provide an identical description of the KNN algorithm. Methods differ by how they normalize and center the data, the way the similarity factors are computed, the selection of neighboring columns, the weighting scheme, and so on. See, for example, [19, 24, 46, 127, 128, 151, 235].

An alternative way to implement KNN imputing is by searching for k similar rows. The "row" version is often used in DNA microarray data, where each row refers to a different gene, e.g., [94, 125, 147, 269]. In recommender systems the row oriented KNN is referred to as "user oriented" approach, or "user nearest neighbor", while the column version is referred to as "item oriented" approach, or "item nearest neighbor", see [152] and [189].

The basic KNN algorithm repeats the search for k suitable neighbors for any missing entry of A. This turns out to be time consuming when A is a large matrix with high percentage of missing entries. Below we describe some ways to reduce the computational efforts.

3.4 Modified KNN methods

The main saving is achieved by conducting one search per column (or one search per row). For this purpose we use a "trial" matrix, T, which is obtained from $\tilde{A}$ by setting some "trial values" in the positions of the missing entries. In column-KNN the columns of $\tilde{A}$ are assumed to be centered and normalized. Then setting zeros into the missing entries provides a trail matrix T whose columns remain centered and normalized. The columns of $\tilde{A}$ and T are denoted by $\tilde{\mathbf{c}}_1, \ldots, \tilde{\mathbf{c}}_n$ and $\mathbf{t}_1, \ldots, \mathbf{t}_n$, respectively.

Treating the jth column, $j = 1, \ldots, n$.

a) If $\tilde{\mathbf{c}}_j$ has no missing entries skip to the next column.
b) Select k columns of T which are most similar to $\mathbf{t}_j$. These columns are used to construct an $m \times k$ matrix N_j.
c) Compute a k-vector **x** such that the vector $\mathbf{d} = N_j\mathbf{x}$ estimates $\mathbf{t}_j$.

d) Set the missing entries in $\tilde{\mathbf{c}}_j$ to be the corresponding entries of **d**.

The above four steps are repeated for each column of $\tilde{A}$. Let $\mathbf{n}_1, \ldots, \mathbf{n}_k$, denote the columns of N_j, and let $\psi_\ell, \ell = 1, \ldots, k$, denote the computed similarity factors between $\mathbf{t}_j$ and $\mathbf{n}_\ell$. The larger is ψ_ℓ the more similar is $\mathbf{n}_\ell$. Then a common practice is to define **x** as the related vector of weights. That is, $\mathbf{x} = (\xi_1, \ldots, \xi_k)^T$, where

$$\xi_\ell = \psi_\ell/(\psi_1 + \cdots + \psi_k), \quad \ell = 1, \ldots, k. \tag{3.6}$$

This way the estimating vector

$$\mathbf{d} = N_j\mathbf{x} = (\psi_1\mathbf{n}_1 + \cdots + \psi_k\mathbf{n}_k)/(\psi_1 + \cdots + \psi_k), \tag{3.7}$$

is a weighted average of the k nearest neighbors.

Another way to define **x** was recently proposed in [19] and [147]. In this method **x** is computed by solving the linear least squares problem

$$\text{minimize}\|N_j\mathbf{x} - \mathbf{t}_j\|_2^2. \tag{3.8}$$

The idea of [19] is based on the observation that **x** can be computed by solving the related system of normal equations

$$N_j^T N_j\mathbf{x} = N_j^T\mathbf{t}_j. \tag{3.9}$$

Let the symmetric $n \times n$ matrix $S = (s_{ij})$ be defined by the equality $S = T^T T$. Then, assuming that the columns of T are centered and normalized, $s_{ij} = \mathbf{t}_i^T\mathbf{t}_j$ is the similarity coefficient between $\mathbf{t}_i$ and $\mathbf{t}_j$. So the search for neighbors of $\mathbf{t}_j$ can be achieved by considering the entries in the jth row (or column) of S. Moreover, once the k neighbors are determined, the matrix $N_j^T N_j$ and the vector $N_j^T\mathbf{t}_j$ are easily constructed from S. Hence, as noted in [19], it might be advantageous to compute and store S before starting the KNN algorithm. (Since S is symmetric, it is sufficient to store its upper diagonal part, which requires $n(n-1)/2$ storage locations.) Nevertheless, as n increases behind a certain level, implementing the column-KNN becomes "too expensive".

A similar difficulty arises in row-KNN when m is "too large". Here the rows of $\tilde{A}$ and T are assumed to be centered and normalized, and the similarity matrix, $S = TT^T$, is an $m \times m$ matrix. Therefore row-KNN becomes expensive as m increases behind a certain level.

The solution proposed by some authors is based on the computation of a rank-k matrix that approximates $\tilde{A}$. The actual computation of such a matrix is achieved by solving (8.1) or (8.17), where $F(U, V)$ is defined with $\tilde{a}_{ij}$ instead of a_{ij}. Let the matrices $\tilde{U} \in \mathbb{R}^{m\times k}$ and $\tilde{V} \in \mathbb{R}^{n\times k}$ solve the related least squares problem. Then the unknown entries of $\tilde{A}$ attain the values of the corresponding entries in the rank-k matrix $\tilde{U}\tilde{V}^T$. Let $\tilde{\mathbf{u}}_i^T$ denote the ith row of $\tilde{U}$, $i = 1, \ldots, m$. Then the missing entries in the ith row of $\tilde{A}$ are replaced by the corresponding entries of the row vector $(\tilde{V}\tilde{\mathbf{u}}_i)^T$. Therefore this method can be interpreted as row-KNN in which the k columns of $\tilde{V}$ serve as neighbors for all the rows of $\tilde{A}$. Similarly, it can be interpreted as column-KNN method in which the columns of $\tilde{U}$ serve as neighbors. For further discussions of this approach see [19], [151], [152], and [235].

The use of a "trial" matrix, T, simplifies the computation of the similarity factors and the approximating vector **d**. The definition of T is based on crude estimates of the missing entries. However, once the imputing process is finished, we have better estimates of the missing entries, and we are able to construct an improved trial matrix. This suggests repeating the imputing process with the new trial matrix. The newly computed entries are used to construct a new trail matrix, and so forth. This idea stands behind the following iteration.

4 Iterative Columns Regression (ICR)

As its name says, ICR is an iterative algorithm. It starts by setting initial trial values to the missing entries of A. Then these values are updated during the iterative process. The basic iteration is composed of n steps, where the jth step updates the values of the missing entries in $\mathbf{c}_j$, the jth column of A, $j = 1, \ldots, n$. Let the $m \times n$ matrix $\tilde{A} = (\tilde{a}_{ij})$ denote the current "completion" of A at the beginning of the jth step. That is, $\tilde{a}_{ij} = a_{ij}$ whenever a_{ij} is "known", while the other entries are assigned the best current trial values. Then the jth step, $j = 1, \ldots, n$, is carried out as follows:

Step 1: Compute $\tilde{m}$, the number of known entries in $\mathbf{c}_j$. If $\tilde{m} = m$ proceed to the next column.

Step 2: Let the $\tilde{m}$-vector **b** be obtained from $\mathbf{c}_j$ by deleting the missing entries.

Step 3: The $m \times (n-1)$ matrix $\tilde{A}_j$ is obtained from $\tilde{A}$ by deleting the jth column of $\tilde{A}$.

Step 4: The $\tilde{m} \times (n-1)$ matrix $\hat{A}_j$ is obtained from $\tilde{A}_j$ by deleting the rows which correspond to missing entries in the jth column of A.

Step 5: Compute $\hat{\mathbf{x}}$, the minimum norm solution of the least squares problem

$$\text{minimize } \|\hat{A}_j\mathbf{x} - \mathbf{b}\|_2^2. \tag{4.1}$$

Step 6: Compute the m-vector $\mathbf{d} = \tilde{A}_j\hat{\mathbf{x}}$.

Step 7: Set new trial values to the missing entries in $\mathbf{c}_j$. The new values are those of the corresponding entries in **d**.

The ICR algorithm is quite intuitive. It is based on the tacit assumption that the columns of the matrix "behave" in a similar way. So regressing one column against the others is likely to provide good results. (The term "regressing" means that we solve a linear least squares problem of the form (4.1).) Of course, if the columns of A can be classified into clusters of similar columns ("neighbors") then the algorithm is applied on each cluster separately. The algorithm described above is easily converted into a "row-version" that takes advantage of similarity between rows.

In statistician's eyes ICR can be viewed as an Expectation-Maximization (EM) method. Yet it is not clear if the algorithm converges, nor what are the properties of a limit point. The experiments in [61] and [125] reveal a slow rate of convergence (as is typical of the EM), but

the quality of the imputed entries competes successfully against KNN imputing and Iterative SVD methods.

A simpler implementation of ICR is achieved by skipping steps 2 and 4. In this version **b** equals the jth column of $\tilde{A}$, and $\hat{A}_j$ is composed from the other columns of $\tilde{A}$.

5 Restricted SVD imputing

Let $\tilde{m}$ denote the number of complete rows in A, and assume for simplicity that $\tilde{m} \geq n$. Let $\tilde{A}$ denote the $\tilde{m} \times n$ matrix containing all the rows of A which don't have any missing entries. In some applications $\tilde{A}$ is sufficiently large and the SVD of $\tilde{A}$ provides valuable information on the missing entries, e.g., [44], [88], and [125]. Let

$$\tilde{A} = U\Sigma V^T \tag{5.1}$$

be a Singular Value Decomposition (SVD) of $\tilde{A}$. That is, the matrices

$$U = [\mathbf{u}_1, \ldots, \mathbf{u}_n] \in \mathbb{R}^{\tilde{m}\times n} \quad \text{and} \quad V = [\mathbf{v}_1, \ldots, \mathbf{v}_n] \in \mathbb{R}^{n\times n} \tag{5.2}$$

have orthonormal columns,

$$\Sigma = \text{diag}\{\sigma_1, \ldots, \sigma_n\} \tag{5.3}$$

is a diagonal matrix, and

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0. \tag{5.4}$$

Define

$$U_k = [\mathbf{u}_1, \ldots, \mathbf{u}_k] \in \mathbb{R}^{\tilde{m}\times k},\ V_k = [\mathbf{v}_1, \ldots, \mathbf{v}_k] \in \mathbb{R}^{n\times k},$$
$$\text{and} \quad \Sigma_k = \text{diag}\{\sigma_1, \ldots, \sigma_k\}. \tag{5.5}$$

That is, U_k is composed of the first k columns of U, V_k is composed of the first k columns of V, and the diagonal $k \times k$ matrix Σ_k consists of the largest k singular values of $\tilde{A}$. Then the matrix

$$T_k(\tilde{A}) = U_k\Sigma_k V_k^T = \sum_{j=1}^{k} \sigma_j \mathbf{u}_j \mathbf{v}_j^T \tag{5.6}$$

is called a rank-k Truncated SVD of $\tilde{A}$. (TSVD in brief.) As mentioned in the introduction, $T_k(\tilde{A})$ is a rank-k approximation of $\tilde{A}$ regarding the Frobenius matrix norm.

To start the restricted SVD algorithm we determine k and compute V_k. Then the columns of V_k serve as the k nearest neighbors of each row with missing entries. Assume for simplicity that $\mathbf{a}_1^T$, the first row of A, has missing entries. Then these entries are imputed via the following four steps. The other rows are treated in a similar way.

Step 1: Compute an $\hat{n}$-vector, $\hat{\mathbf{a}}_1$, by deleting the missing entries of $\mathbf{a}_1$. That is, $\hat{\mathbf{a}}_1$ consists of the known entries of $\mathbf{a}_1$ in their original order.

Step 2: Compute an $\hat{n} \times k$ matrix, $\hat{V}_k$, by deleting the corresponding rows of V_k.

Step 3: Compute $\hat{\mathbf{x}}$, the minimum norm solution of the linear least squares problem

$$\underset{\mathbf{x}}{\text{minimize}} \ \|\hat{V}_k\mathbf{x} - \hat{\mathbf{a}}_1\|_2^2. \tag{5.7}$$

Step 4: Compute the estimating n-vector $V_k\hat{\mathbf{x}}$, and set the missing entries in $\mathbf{a}_1$ to be the corresponding entries in $V_k\hat{\mathbf{x}}$.

Recall that a principal submatrix of A is a smaller matrix which is obtained by deleting some rows and columns of A. The restricted SVD method can be extended to data matrices that include a large principal submatrix without missing entries. See [44] for the details.

Observe that the SVD of $\tilde{A}$ serves as a substitute for that of A. However, once all the missing entries in A are imputed, it is possible to compute an SVD of the resulting $m \times n$ matrix. Then, using the new SVD, it is possible to derive improved estimates of the missing entries. Repeating this process leads to the following "iterative SVD" algorithm.

6 Iterative SVD imputing

This method is both simple and elegant. Let

$$\mathbb{A} = \{\tilde{A} = (\tilde{a}_{ij}) \,|\, \tilde{A} \in \mathbb{R}^{m\times n} \ \text{ and } \ \tilde{a}_{ij} = a_{ij} \ \text{ when } \ a_{ij} \ \text{ is known } \} \tag{6.1}$$

denote the set of all "admissible" matrices for completing A. The iterative SVD method generates a sequence of admissible matrices

$$A_\ell \in \mathbb{A}, \quad \ell = 1, 2, \ldots, \tag{6.2}$$

where $A_{\ell+1}$ is obtained by using the SVD of A_ℓ. Let

$$A_\ell = U\Sigma V^T \tag{6.3}$$

be an SVD of A_ℓ and let

$$T_k(A_\ell) = U_k\Sigma_k V_k^T \tag{6.4}$$

denote the corresponding rank-k TSVD of A_ℓ. (These matrices satisfy (5.1)--(5.6) with A_ℓ and m instead of $\tilde{A}$ and $\tilde{m}$, respectively.) Then the entries of $A_{\ell+1}$ which refer to missing entries in A obtain the values of the corresponding entries in $T_k(A_\ell)$. Using the matrix operator P_Ω the lth iteration, $\ell = 1, 2, \ldots$, is summarized as follows. Given A_ℓ, compute

$$B_\ell = T_k(A_\ell). \tag{6.5}$$

Then $A_{\ell+1}$ is obtained by the rule

$$A_{\ell+1} = P_\Omega(A) + [B_\ell - P_\Omega(B_\ell)]. \tag{6.6}$$

Note that B_ℓ is the projection of A_ℓ on $\mathbb{B}_k$ (regarding the Frobenius matrix norm), while $A_{\ell+1}$ is the projection of B_ℓ on $\mathbb{A}$. So the method can be interpreted as "alternating projections" between $\mathbb{A}$ and $\mathbb{B}_k$.

The iterative SVD algorithm is, perhaps, the most celebrated example of an imputing method that descends from the statistical maximum likelihood approach and the related expectation-maximization (EM) framework: The algorithm alternates between the imputing process (expectation) and SVD computation (maximization). Indeed in many applications the algorithm is presented in this way, e.g., [64, 94, 117, 156, 269, 289].

In Optimization terminology it is a proximal point algorithm. To see this point we relate A_ℓ with the proximal problem

$$\begin{array}{ll} \text{minimize} & \Pi_\ell(B) = \|A_\ell - B\|_F^2 \\ \text{subject to} & B \in \mathbb{B}_k, \end{array} \tag{6.7}$$

which has a minimizer at B_ℓ. Note that $F(B)$, the objective function of (1.1), is a partial sum of the proximal function $\Pi_\ell(B)$. Therefore,

$$\Pi_\ell(B) \geq F(B) \quad \forall B \in \mathbb{R}^{m\times n}$$

and

$$\Pi_\ell(B_{\ell-1}) = F(B_{\ell-1}).$$

Combining these relations yields the "decreasing property"

$$F(B_{\ell-1}) = \Pi_\ell(B_{\ell-1}) \geq \Pi_\ell(B_\ell) \geq F(B_\ell), \tag{6.8}$$

which enables us to consider the iterative SVD algorithm as minimization method for solving (1.1).

It is also possible to consider the iterative SVD method as a gradient projection method. The gradient vector of $F(B)$ is given by the matrix $-2[P_\Omega(A) - P_\Omega(B)]$, while (6.6) can be rewritten as

$$A_{\ell+1} = B_\ell + [P_\Omega(A) - P_\Omega(B_\ell)]. \tag{6.9}$$

That is, first $A_{\ell+1}$ is obtained by moving from B_ℓ along the steepest descent direction, then $B_{\ell+1}$ is defined as the projection of $A_{\ell+1}$ on $\mathbb{B}_k$. This interpretation has led Meka et al. [190] to suggest a modified iteration that is called Singular Value Projection (SVP). In this version

$$A_{\ell+1} = B_\ell + \theta_\ell[P_\Omega(A) - P_\Omega(B_\ell)] \tag{6.10}$$

and

$$B_{\ell+1} = T_k(A_{\ell+1}), \tag{6.11}$$

where $\theta_\ell > 0$ is a "step-length" parameter.

The arguments behind (6.8) can be extended to give

$$\|A_\ell - T_k(A_\ell)\|_F^2 \geq \|A_{\ell+1} - T_k(A_\ell)\|_F^2 \geq \|A_{\ell+1} - T_k(A_{\ell+1})\|_F^2.$$

Hence the related sequence of "tail" values,

$$\tau_k(A_\ell) = \|A_\ell - T_k(A_\ell)\|_F^2, \quad \ell = 1, 2, \ldots,$$

is also decreasing. That is,

$$\tau_k(A_\ell) \geq \tau_k(A_{\ell+1}). \tag{6.12}$$

Recall that $\|A_\ell - T_k(A_\ell)\|_F$ is the distance between A_ℓ and $\mathbb{B}_k$ (when using the Frobenius matrix norm). Moreover, combining (6.8) with (6.12) yields the interlacing relations

$$\tau_k(A_\ell) \geq F(B_\ell) \geq \tau_k(A_{\ell+1}) \geq F(B_{\ell+1}). \tag{6.13}$$

The FRAA algorithm, which is described in the next section, uses a different way to minimize the "tail" function.

The success of the Iterative SVD method depends on proper choices of k and the starting matrix A_1. In practice A_1 is often obtained from A by a simple averaging method, and the algorithm is terminated as soon as the norm of the difference $A_{\ell+1} - A_\ell$ falls below a certain preassigned threshold value, e.g., [88, 94, 125, 156, 269]. However, it is not clear whether the algorithm always converges, nor what properties has a limit point. (The difficulties stem from the fact that $\mathbb{B}_k$ is not a convex set.) Using the decreasing property (6.8) it is easy to verify that any solution of (1.1) is a limit point of the iterative SVD process. This tempts one to believe that the sequence $\{B_\ell\}$ might converge toward a solution of (1.1). Yet, as the following example shows, not every limit point of this sequence solves (1.1).

Example: Let A be a 2×2 matrix, $a_{11} = 4,\ a_{22} = 1$, while a_{12} and a_{21} are considered as unknown. Then here $k = 1$ is the only meaningful choice of k. Now the trial values $a_{12} = a_{21} = 0$ result in a limit point that fails to solve (1.1). On the other hand, setting $a_{12} = a_{21} = 2$ provides a limit point that solves (1.1).

In some applications, when m is considerably larger than n, it might be faster to avoid the computation of U_k and $T_k(A_\ell)$. In this version V_k is derived from the spectral decomposition

$$A_\ell^T A_\ell = V \Sigma^2 V^T. \tag{6.14}$$

Then the missing values at a certain row of A_ℓ are updated by regressing the row against the columns of V_k. Let us consider for example $\mathbf{a}_1$, the first row of A_ℓ. Regressing $\mathbf{a}_1$ against the columns of V_k results in the vector $\hat{\mathbf{x}} = V_k^T \mathbf{a}_1$, which solves the linear least square problem

$$\text{minimize } \|V_k \mathbf{x} - \mathbf{a}_1\|_2^2. \tag{6.15}$$

This way the missing entries in the first row of $A_{\ell+1}$ are defined as the corresponding entries in the row vector $(V_k V_k^T \mathbf{a}_1)^T$. In matrix notations this updating means that the entries of $A_{\ell+1}$ which refer to missing entries in A obtain the values of the corresponding entries in $A_\ell V_k V_k^T$. In exact arithmetic

$$T_k(A_\ell) = A_\ell V_k V_k^T, \tag{6.16}$$

and both versions generate the same sequence of matrices. We see, therefore, that the basic SVD iteration can be interpreted as row-KNN in which the k columns of V_k serve as neighbors for all the rows of A. Similarly, it can be interpreted as column-KNN in which the columns of U_k serve as neighbors.

A different version of iterative SVD is described in [269]. Here the regression of $\mathbf{a}_1$ against the columns of V_k is done after omitting the rows corresponding to missing entries in $\mathbf{a}_1$. That is, by following Steps 1--4 of the restricted SVD algorithm. Consequently for each row with missing entries we use a different matrix $\hat{V}_k$, and solve a different least squares problem of the form (5.7). The matrices generated in this method differ, therefore, from those generated by the other versions.

The Hard-Impute algorithm, which is proposed in [188], is another example of a closely related method. The basic iteration of this algorithm is similar to that of Iterative SVD. The only difference is that here B_ℓ is obtained from A_ℓ by solving the problem

$$\text{minimize } \chi(B) = {}_{1/2}\|B - A_\ell\|_F^2 + \lambda \text{rank}(B), \tag{6.17}$$

where $\lambda > 0$ is a preassigned positive constant. That is, here B_ℓ is defined as the solution of (6.17), which is given by a Truncated SVD of the form

$$B_\ell = T_{k'}(A_\ell), \tag{6.18}$$

where the rank index, k', satisfies

$$\chi(T_{k'}(A_\ell)) = \min_{k=1,\ldots,n} \{\chi(T_k(A_\ell))\}. \tag{6.19}$$

Let $\sigma_1(A_\ell) \geq \sigma_2(A_\ell) \geq \cdots \geq \sigma_n(A_\ell) \geq 0$ denote the singular values of A_ℓ. Then (6.19) means that k' satisfies

$$(\sigma_{k'}(A_\ell))^2 \geq \lambda \geq (\sigma_{k'+1}(A_\ell))^2. \tag{6.20}$$

As before, once B_ℓ is computed, $A_{\ell+1}$ is updated by (6.6). The solution of (6.17) is sometimes called "hard-thresholding" of A_ℓ. This gives the algorithm its name. The main difference is that here the rank of the computed TSVD is not known in advance and it may change during the iterative process. Nevertheless, the practical value of the Hard-Impute algorithm relies on the assumption that the sequences $\{A_\ell\}$ and $\{B_\ell\}$ converge. Let A^* denote the limit point of the sequence $\{A_\ell\}$, if exists. Then, as A_ℓ approaches the vicinity of A^*, the rank index k' stops changing, and the Hard-Impute algorithm coincides with Iterative SVD.

In Netflix-like problems the matrix is so huge that computing $T_k(A_{\ell+1})$ from the SVD of $A_{\ell+1}$ becomes prohibitively expensive. One way to handle this difficulty is to approximate $T_k(A_{\ell+1})$ by applying a Lanczos method on $A_{\ell+1}$. Recall that Lanczos method requires only matrix-vector products of the form $A_{\ell+1}\mathbf{x}$ and $A_{\ell+1}^T\mathbf{y}$. Thus, as noted in [156], it is possible to take advantage of the special structure of $A_{\ell+1}$. From (6.9) we see that $A_{\ell+1}$ is the sum of two matrices: A rank-k matrix, B_ℓ, plus a highly sparse matrix, $P_\Omega(A) - P_\Omega(B_\ell)$. The non-zero entries of the last matrix correspond to known entries of A. Hence a matrix-vector product needs only $k(m+n+1)+\nu$ multiplications. This use of Lanczos method is also mentioned in

[188] and [190]. The Lanczos algorithm is often implemented by using "PROPACK", a Matlab subroutine that was written by R.M. Larsen [159, 160]. A different approach is proposed in [180], where the authors prefer to approximate $T_k(A_{\ell+1})$ by applying the Monte Carlo algorithm [70].

7 Minimizing the singular values tail (FRAA)

The FRAA algorithm [88, 89, 205] is aimed at minimizing the "tail" of the imputed matrix. We shall start by introducing some notations that help to explain this idea. Let

$$\sigma_1(H) \geq \sigma_2(H) \geq \cdots \geq \sigma_n(H) \geq 0 \tag{7.1}$$

denote the singular values of a given matrix $H \in \mathbb{R}^{m\times n}$. Let $T_k(H)$ denote the corresponding rank-k TSVD of H. Then the related "tail" function of H is defined as

$$\tau_k(H) = \|H - T_k(H)\|_F^2 = \sum_{i=k+1}^{n} \sigma_i^2(H). \tag{7.2}$$

As before we use $\mathbb{A}$ to denote the set (6.1) of all "admissible" matrices. The idea of FRAA is to compute an admissible matrix that solves the "tail" problem

$$\begin{aligned} &\text{minimize} \quad \tau_k(H) \\ &\text{subject to} \quad H \in \mathbb{A}. \end{aligned} \tag{7.3}$$

In other words, we seek an admissible matrix which is closest to $\mathbb{B}_k$. The name FRAA stands for Fixed Rank Approximation Algorithm. It is an iterative algorithm that generates a sequence of admissible matrices, $H_\ell,\ \ell = 1, 2, \ldots$, which satisfy

$$\tau_k(H_\ell) \geq \tau_k(H_{\ell+1}), \quad \ell = 1, 2, \ldots. \tag{7.4}$$

The basic iteration starts with H_ℓ and ends with $H_{\ell+1}$. The construction of $H_{\ell+1}$ from H_ℓ is done via the following four steps.

Step 1: Compute the $n \times n$ symmetric matrix

$$G_\ell = H_\ell^T H_\ell. \tag{7.5}$$

Step 2: Compute a spectral decomposition of G_ℓ,

$$G_\ell = V_\ell D_\ell V_\ell^T, \tag{7.6}$$

where $V_\ell = [\mathbf{v}_1, \ldots, \mathbf{v}_n] \in \mathbb{R}^{n\times n}$ has orthonormal columns, $V_\ell^T V_\ell = I$, and $D_\ell \in \mathbb{R}^{n\times n}$ is a diagonal matrix,

$$D_\ell = \text{diag}\{\sigma_1^2(H_\ell), \ldots, \sigma_n^2(H_\ell)\}, \quad \text{and} \quad \sigma_1(H_\ell) \geq \sigma_2(H_\ell) \geq \cdots \geq \sigma_n(H_\ell) \geq 0.$$

Step 3: Compute the $n \times (n-k)$ matrix

$$W_\ell = [\mathbf{v}_{k+1}, \ldots, \mathbf{v}_n], \tag{7.7}$$

which is composed of the last $n-k$ columns of V_ℓ. That is, the columns of W_ℓ are built from right singular vectors of H_ℓ which correspond to the smallest singular values of H_ℓ.

Step 4: Define $H_{\ell+1}$ to be an admissible matrix that solves the minimum norm problem

$$\begin{array}{ll} \text{minimize} & \|HW_\ell\|_F^2 \\ \text{subject to} & H \in \mathbb{A}. \end{array} \tag{7.8}$$

The definitions of W_ℓ and $H_{\ell+1}$ imply the relations

$$\tau_k(H_\ell) = \|H_\ell W_\ell\|_F^2, \quad \ell = 1, 2, \ldots,$$

and

$$\|H_\ell W_\ell\|_F^2 \geq \|H_{\ell+1} W_\ell\|_F^2.$$

Also, as noted in [88], the inequality

$$\|H_{\ell+1} W_\ell\|_F^2 \geq \|H_{\ell+1} W_{\ell+1}\|_F^2$$

is a direct consequence of Ky Fan's minimum principle. (For recent discussion of Ky Fan's extremum principles see [58].) Combining these relations gives

$$\tau_k(H_\ell) = \|H_\ell W_\ell\|_F^2 \geq \|H_{\ell+1} W_\ell\|_F^2 \geq \|H_{\ell+1} W_{\ell+1}\|_F^2 = \tau_k(H_{\ell+1}), \tag{7.9}$$

which proves (7.4).

The solution of (7.8) is based on the following observation. Let $H \in \mathbb{A}$ be any admissible matrix, and let $\mathbf{h}_q^T,\ q = 1, \ldots, m$, denote the rows of H. That is, $H = [\mathbf{h}_1, \ldots, \mathbf{h}_m]^T \in \mathbb{R}^{m \times n}$. Then $\mathbf{h}_q^T W_\ell$ is the qth row of the product matrix HW_ℓ, and

$$\|HW_\ell\|_F^2 = \sum_{q=1}^{m} \|\mathbf{h}_q^T W_\ell\|_2^2. \tag{7.10}$$

Therefore, since the term $\|\mathbf{h}_q^T W_\ell\|_2^2$ can be affected only by the (missing) entries in $\mathbf{h}_q^T$, the overall minimization of the sum (7.10) is carried out by considering one row at a time. Let us consider for example, $\mathbf{a}_1^T$, the first row of A. (The other rows are treated in the same manner.) Then the unknown entries in $\mathbf{a}_1$ are determined in an attempt to minimize the term $\|W_\ell^T \mathbf{a}_1\|_2^2$. This is achieved in the following way. Assume for simplicity that the first p entries of $\mathbf{a}_1$ are missing. That is, $\mathbf{a}_1^T = (\tilde{\mathbf{a}}_1^T, \hat{\mathbf{a}}_1^T)$, where $\tilde{\mathbf{a}}_1 \in \mathbb{R}^p$ denotes the unknown part of $\mathbf{a}_1$, and $\hat{\mathbf{a}}_1 \in \mathbb{R}^{n-p}$ denotes the known part. Let

$$W_\ell^T = [\tilde{W}_\ell, \hat{W}_\ell], \quad \tilde{W}_\ell \in \mathbb{R}^{(n-k)\times \ell}, \quad \hat{W}_\ell \in \mathbb{R}^{(n-k)\times(n-p)} \tag{7.11}$$

denote the corresponding row partition of W_ℓ. Then the unknown entries in $\mathbf{a}_1$ are obtained by solving the linear least squares problem

$$\underset{\mathbf{x}\in\mathbb{R}^\ell}{\text{minimize}} \ \|\tilde{W}_\ell \mathbf{x} - \mathbf{z}_\ell\|_2^2, \tag{7.12}$$

where $\mathbf{z}_\ell = -\hat{W}_\ell \hat{\mathbf{a}}_1$. This way for each row with missing entries we need to solve a different least squares problem of the form (7.12). If the solution of (7.12) is not unique we take the smallest one. Also it is necessary to guard against cases when $\tilde{W}_\ell$ is a null or ill-conditioned matrix. (See the example given in the previous section.)

As we shall see in the next section, similar linear least squares problems are solved in the ALS iteration. However, the concept of FRAA is somewhat surprising and ingenious. The decreasing property (7.4) reduces the possibility of converging into a nonoptimal point, but the convergence properties of the sequence $\{H_\ell\}$ have not yet been studied. The algorithm described in [88] terminates after a preassigned number of iterations. Another input parameter of the algorithm is k. The role of k in FRAA is similar to its role in iterative SVD. However, the iterative SVD method uses only the first k "principal" eigenvectors, while FRAA is using the last $n-k$ eigenvectors. This might be a drawback in some cases. For example, when n is considerably larger than k, or when H_ℓ happens to be an ill-conditioned matrix.

The tail problem (7.3) looks quite different from the least squares problem (1.1). However, as we now show, the two problems are closely related.

Theorem 1 (An Equivalence theorem)**.** *Let the rank-k matrix B_k^* solve the least squares problem* (1.1)*, and let the admissible matrix $A^* \in \mathbb{A}$ be obtained from B_k^* by setting the unknown entries of A to be the corresponding entries of B_k^*. Then A^* solves the tail problem* (7.3) *and*

$$F(B_k^*) = \tau_k(A^*). \tag{7.13}$$

Conversely, let $\hat{A} \in \mathbb{A}$ be an admissible matrix that solves the tail problem (7.3)*, and let $T_k(\hat{A})$ denote a rank-k TSVD of $\hat{A}$. Then $T_k(\hat{A})$ solves* (1.1) *and*

$$F(T_k(\hat{A})) = \tau_k(\hat{A}). \tag{7.14}$$

In other words, a solution of (1.1) *provides a solution of* (7.3)*, and vice versa. Furthermore, both problems share the same optimal value.*

Proof. Let B_k^* and A^* be as above. Then, since B_k^* solves (1.1), it also solves the problem

$$\begin{aligned} &\text{minimize} \quad \|A^* - B\|_F^2 \\ &\text{subject to} \quad B \in \mathbb{B}_k. \end{aligned} \tag{7.15}$$

The last observation means that $B_k^* = T_k(A^*)$. That is, B_k^* is a rank-k TSVD of A^*. The "tail" of A^* satisfies, therefore,

$$\tau_k(A^*) = \|A^* - B_k^*\|_F^2 = F(B_k^*). \tag{7.16}$$

Now let $\tilde{A} \in \mathbb{A}$ be any other admissible matrix, and let $T_k(\tilde{A}) \in \mathbb{B}_k$ be a rank-k TSVD of $\tilde{A}$. Then

$$F(B_k^*) \leq F(T_k(\tilde{A})) \leq \|\tilde{A} - T_k(\tilde{A})\|_F^2 = \tau_k(\tilde{A}). \tag{7.17}$$

Combining (7.16) and (7.17) gives the inequality

$$\tau_k(A^*) \leq \tau_k(\tilde{A}), \tag{7.18}$$

which proves that A^* solves (7.3).

The converse claim is derived in a similar way. Let $\hat{A} \in \mathbb{A}$ solve (7.3) and let $T_k(\hat{A})$ be a rank-k TSVD of $\hat{A}$. Then

$$\tau_k(\hat{A}) = \|\hat{A} - T_k(\hat{A})\|_F^2 \geq F(T_k(\hat{A})), \tag{7.19}$$

where the last inequality is due to the fact that $F(\hat{A})$ is a partial sum of $\|\hat{A} - T_k(\hat{A})\|_F^2$. Now let $\tilde{B} \in \mathbb{B}_k$ be any rank-k matrix, and let $\tilde{A} \in \mathbb{A}$ denote the corresponding admissible matrix. That is, $\tilde{A}$ is obtained by setting the unknown entries to be those of $\tilde{B}$. As before, $T_k(\tilde{A})$ denotes a rank-k TSVD of $\tilde{A}$. Then these matrices satisfy

$$F(\tilde{B}) = \|\tilde{A} - \tilde{B}\|_F^2 \geq \|\tilde{A} - T_k(\tilde{A})\|_F^2 = \tau_k(\tilde{A}). \tag{7.20}$$

On the other hand, since $\hat{A}$ solves (7.3),

$$\tau_k(\tilde{A}) \geq \tau_k(\hat{A}). \tag{7.21}$$

Hence combining (7.19)--(7.21) shows that

$$F(\tilde{B}) \geq F(T_k(\hat{A})), \tag{7.22}$$

which proves that $T_k(\hat{A})$ solves (1.1). □

The last theorem adds important insight into the nature of both problems. In particular we see that solving (1.1) provides a matrix A^* that has the "minimum tail" property.

An improved version of FRAA, which is called IFRAA, is proposed in [89]. It is aimed to take advantage of the clustering feature that characterizes DNA microarray data matrices. For this purpose the imputing process is carried out in three stages. The first stage uses FRAA to complete the missing entries in A, generating an admissible matrix H^* that solves (7.3). The second stage applies a clustering algorithm to separate the rows of H^* into a small number of clusters, where each cluster consists of similar rows. Then, at the last stage, FRAA is applied again, but this time it is used on each cluster separately, updating the values of the missing entries in that cluster.

8 Least Squares Methods

We have seen that both Iterative SVD and FRAA are implicitly aimed at solving (1.1). In this section we consider direct methods for solving this problem. Many of them utilize the following observation. Given a rank-k matrix $B = (b_{ij}) \in \mathbb{R}^{m\times n}$, there exists a pair of matrices, $U = (u_{ij}) \in \mathbb{R}^{m\times k}$ and $V = (v_{ij}) \in \mathbb{R}^{n\times k}$, such that $B = UV^T$. (The matrices U and V are easily derived from the SVD of B.) The last equality can be written as

$$b_{ij} = \mathbf{u}_i^T \mathbf{v}_j, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n,$$

where $\mathbf{u}_i^T$ denotes the ith row of U, and $\mathbf{v}_j^T$ denotes the jth row of V. These relations enable us to rewrite (1.1) in the form

$$\begin{aligned} &\text{minimize} \quad F(U,V) = \|P_\Omega(A) - P_\Omega(UV^T)\|_F^2 = \sum_{(i,j)\in\Omega} (a_{ij} - \mathbf{u}_i^T \mathbf{v}_j)^2 \\ &\text{subject to} \ \ U \in \mathbb{R}^{m\times k} \quad \text{and} \quad V \in \mathbb{R}^{n\times k}. \end{aligned} \tag{8.1}$$

The last problem is equivalent to (1.1) in the sense that any solution of one problem provides a solution to the other problem, and both solutions share the same optimal value.

In some applications problem (8.1) has a simple interpretation. Let us consider for example the Netflix rating matrix, and assume that a movie is characterized by k principal features. (Such as being funny, romantic, musical, dramatic, having "action", and so forth.) In this case the jth movie can be characterized by a k-vector, $\mathbf{v}_j$, whose entries quantify the related features. Similarly, the ith viewer is characterized by a k-vector, $\mathbf{u}_i$, whose entries reflect the user "taste" with regard to these features. Then the rating given by the ith user to the jth movie is expected to be about $\mathbf{u}_i^T \mathbf{v}_j$, e.g., [19, 92, 151, 152, 188, 209]. This kind of interpretation stands behind the factor analysis methodology. However, usually the number of "principal features", k, is not known in advance. So the value of k has to be determined in some way.

The methods described in this section are "descent methods" for solving (8.1). Starting from a given initial point (U_0, V_0) they generate a sequence of points $(U_\ell, V_\ell), \ell = 1, 2, \ldots$ such that

$$F(U_{\ell-1}, V_{\ell-1}) \geq F(U_\ell, V_\ell), \quad \ell = 1, 2, \ldots.$$

As we shall see, many of these methods are classical minimization techniques that have been modified to take advantage of the special structure of (8.1).

The solution of (1.1) and (8.1) faces a number of difficulties. First note that the set of optimal solutions can be empty. The following example of Gillis and Gilneur [100] illustrates this point. Let the matrix to complete have the form $A = (a_{ij}) \in \mathbb{R}^{2\times 2}$, $a_{11} = a_{22} = 1$, $a_{21} = 0$, and a_{12} is unknown. Now consider the approximation of A by a rank-one matrix of the form $\mathbf{u}\mathbf{v}^T$. In this case the optimal value, if exists, must differ from zero. But taking $\mathbf{u} = (1, 1/\mu)^T$, $\mathbf{v} = (1, \mu)^T$, and increasing μ toward infinity shows that the infimum value is never achieved. It is also shown in [100] that for certain types of matrices with nonnegative entries the rank-one approximation problem (8.3) is NP-hard. (The proof takes an NP-hard

problem on a bipartite graph, "The Maximum-Edge Biclique Problem", and shows its relation to a rank-one problem.)

Observe that $\mathbb{B}_k$ is not a bounded set. Also, using a truncated SVD of an identity matrix, it is easy to see that $\mathbb{B}_k$ is not a convex set, and that the solution of (1.1) is not always unique. These drawbacks are inherent to problem (8.1), as the following example shows. Let us consider problem (8.1) in the special case when $m = n = 2, \ \ k = 1, \quad$ and $\quad a_{11} = a_{22} = 1.$ The other entries are "missing". Then here $F(U, V) = 0$ for any pair of matrices of the form $U = [\alpha, \beta]^T \in \mathbb{R}^{2\times 1}$ and $V = [1/\alpha, 1/\beta]^T \in \mathbb{R}^{2\times 1}, \ \ \alpha \neq 0, \ \ \beta \neq 0$. So (8.1) has infinitely many solutions. To see non-convexity consider two specific solution points, $U_1 = V_1 = [1, 1]^T$, and $U_2 = V_2 = [1, -1]^T$. Then, clearly, $F(U_1, V_1) = F(U_2, V_2) = 0$. Now let the point (U_3, V_3) be defined by the equalities $U_3 = V_3 = [1, 0]^T = \ {}_{1/2}(U_1 + U_2) = \ {}_{1/2}(V_1 + V_2)$. Then this point lies on a line segment that connects the two solution points, but $F(U_3, V_3) = 1$, which shows that $F(U, V)$ is not a convex function.

Another type of non-uniqueness stems from the following observation. Let U and V be as in (8.1) and let S be any invertible matrix in $\mathbb{R}^{k\times k}$. Then

$$UV^T = (US)(VS^{-T})^T \tag{8.2}$$

and

$$F(U, V) = F(US, VS^{-T}).$$

Moreover, using the Neumann series expansion we see that for any matrix $R \in \mathbb{R}^{k\times k}$ that satisfies $\rho(R) < 1$, the set

$$\{(U(I - \alpha R), \ \ V(I - \alpha R)^{-T}) \mid \ -1 \leq \alpha \leq 1\}$$

defines a continuous trajectory on which the objective function attains a constant value. This feature means that $F(U, V)$ has no isolated local minimizers. It also suggests that the related Jacobian and Hessian matrices are likely to be ill-conditioned. Further situations that invite nonuniqueness are discussed in Section 8.5 below.

The difficulties mentioned above may hinder some descent methods from converging toward a global minimizer of (8.1). Yet the use of regularization methods, Riemannian optimization techniques and "Hybrid Methods" provides some remedies to these difficulties. The question of how to find a suitable value for k is deferred until Section 20. As we shall see, it is possible to answer this question by constructing a finite sequence of matrices, $B_k^*, \ k = 1, 2, \ldots, \hat{k}$, such that B_k^* solves (1.1). Then the related sequence of objective function values helps to determine an appropriate value for k. Thus in practice we would like to solve (8.1) for $k = 1, 2, \ldots, \hat{k}$. This suggests the use of B_k^* to produce an improved starting point for the computation of B_{k+1}^*. The details of this idea are elaborated in the coming sections.

8.1 Computing a rank-one approximation: The "criss-cross" iteration

In this section we describe a simple iterative algorithm for computing a rank-one approximation of A. Recall that a rank-one matrix in $\mathbb{R}^{m\times n}$ has the form $\mathbf{uv}^T$, where $\mathbf{u} = (u_1, \ldots, u_m)^T \in$

$\mathbb{R}^m$ and $\mathbf{v} = (v_1, \ldots, v_n)^T \in \mathbb{R}^n$. Hence in this case the least squares problem (8.1) is reduced to

$$\text{minimize} \quad f(\mathbf{u}, \mathbf{v}) = \sum_{(i,j)\in\Omega} (a_{ij} - u_i v_j)^2. \tag{8.3}$$

The algorithm described below is aimed at solving this problem. Let $\mathbf{u}_\ell$ and $\mathbf{v}_\ell$ denote the current estimate of the solution at the beginning of the ℓth iteration, $\ell = 1, 2, \ldots$. Then the ℓth iteration is composed of the following two steps.

Step 1: Given $\mathbf{v}_\ell = (\tilde{v}_1, \ldots, \tilde{v}_n)^T \in \mathbb{R}^n$ the vector $\mathbf{u}_{\ell+1} = (\hat{u}_1, \ldots, \hat{u}_m)^T \in \mathbb{R}^m$ is obtained by solving the linear least squares problem

$$\text{minimize} \quad \varphi(\mathbf{u}) = \sum_{i=1}^{m} \sum_{j\in\mathbb{R}_i} (a_{ij} - u_i \tilde{v}_j)^2.$$

That is,

$$\hat{u}_i = \Big(\sum_{j\in\mathbb{R}_i} a_{ij}\tilde{v}_j \Big) / \Big(\sum_{j\in\mathbb{R}_i} \tilde{v}_j^2 \Big), \quad i = 1, \ldots, m, \tag{8.4}$$

where the index set

$$\mathbb{R}_i = \{ j \,|\, a_{ij} \ \text{ is known } \}$$

contains all the column indices of known entries in the ith row of A.

Step 2: Given $\mathbf{u}_{\ell+1} = (\hat{u}_1, \ldots, \hat{u}_m)^T \in \mathbb{R}^m$ the vector $\mathbf{v}_{\ell+1} = (\hat{v}_1, \ldots, \hat{v}_n)^T \in \mathbb{R}^n$ is obtained by solving the linear least squares problem

$$\text{minimize} \quad \psi(\mathbf{v}) = \sum_{j=1}^{n} \sum_{i\in\mathbb{C}_j} (a_{ij} - \hat{u}_i v_j)^2.$$

That is,

$$\hat{v}_j = \Big(\sum_{i\in\mathbb{C}_j} a_{ij}\hat{u}_i \Big) / \Big(\sum_{i\in\mathbb{C}_j} \hat{u}_i^2 \Big), \quad j = 1, \ldots, n, \tag{8.5}$$

where the index set

$$\mathbb{C}_j = \{ i \,|\, a_{ij} \ \text{ is known } \}$$

contains all the row indices of known entries in the jth column of A.

The iteration (8.4)--(8.5) was proposed by Gabriel and Zamir [93] under the name "criss-cross regressions". Observe that minimizing $f(\mathbf{u}, \mathbf{v})$ by changing one variable at a time, in the order $u_1, \ldots, u_m, \ v_1, \ldots, v_n$, results in the same basic iteration. In the optimization literature the last method is called the "coordinate descent" method, e.g., [22], or the "alternating variables" method, e.g., [84]. Note also that

$$f(\mathbf{u}, \mathbf{v}) = f(\alpha\mathbf{u}, \mathbf{v}/\alpha) \quad \forall \alpha \neq 0.$$

This enables us to extend the basic iteration with the following normalization step: First set $\alpha = \|\mathbf{v}_{\ell+1}\|_2$, then $\mathbf{u}_{\ell+1}$ and $\mathbf{v}_{\ell+1}$ are replaced with $\alpha\mathbf{u}_{\ell+1}$ and $\mathbf{v}_{\ell+1}/\alpha$, respectively, e.g., [276].

An important insight into the nature of the above iteration is revealed by considering the case when A has no missing entries. In this case the extended iteration,

$$\mathbf{u}_{\ell+1} = A\mathbf{v}_\ell \quad \text{and} \quad \mathbf{v}_{\ell+1} = A^T\mathbf{u}_{\ell+1}/\|A^T\mathbf{u}_{\ell+1}\|_2, \tag{8.6}$$

coincides with the basic iteration of the Power method applied to A^TA. The last method is known to have a linear rate of converges. The asymptotic rate is proportional to $\sigma_2^2(A)/\sigma_1^2(A)$, and can be arbitrarily slow. However, often the initial rate of convergence is considerably faster than the asymptotic rate, and the power method provides a valuable estimate of the solution within a small number of iterations. See [55] and [57]. If the initial vector, $\mathbf{v}_1$, is perpendicular to the space spanned by dominant eigenvectors of A^TA, then the sequence $\{\mathbf{v}_\ell\}$ may fail to approach this space. This difficulty is passed on to the iterations (8.4)--(8.5). Return, for example, to the 2×2 matrix in which $a_{11} = 4,\ a_{22} = 1$, while a_{12} and a_{21} are unknown. In this example starting from $\mathbf{v}_1 = (0,1)^T$, or $\mathbf{v}_1 = (1,0)^T$, results in $\mathbf{v}_\ell = \mathbf{v}_1$ for $\ell = 2,3,\ldots$. A common way to reduce the possibility of converging into a nonoptimal point is to use a "random" starting vector, whose entries are random numbers between -1 and 1. A more sophisticated approach is proposed in [93].

Finally we mention that a closely related iteration is achieved when applying rank-one iterative SVD. Recall that iterative SVD generates a sequence of admissible matrices, $A_i,\ i = 1,2,\ldots$. A rank-one iterative SVD computes a dominant pair of singular vectors of A_i by applying the Power iteration (8.6) on A_i, e.g., [146] or [276].

8.2 Failure of the basic deflation procedure

The basic deflation procedure generates a sequence of $m \times n$ matrices, $B_k,\ k = 1,2,\ldots$, where B_k is a rank-k matrix that attempts to approximate A. Given $B_k = (b_{ij})$ the next matrix is computed as follows. Let the $m \times n$ matrix $\tilde{A}_k = (\tilde{a}_{ij})$ be defined in the following way: $\tilde{a}_{ij} = a_{ij} - b_{ij}$ when a_{ij} is known. Otherwise, when a_{ij} is unknown, $\tilde{a}_{ij}$ is also considered as unknown. Note that $\tilde{A}_k$ has the same pattern of missing entries as A. Let the vectors $\tilde{\mathbf{u}}_{k+1} \in \mathbb{R}^m$ and $\tilde{\mathbf{v}}_{k+1} \in \mathbb{R}^n$ be obtained by solving the rank-one approximation problem (8.3) with $\tilde{A}_k$ instead of A. That is, the problem to solve has the form

$$\text{minimize} \quad f(\mathbf{u},\mathbf{v}) = \sum_{(i,j)\in\Omega} (\tilde{a}_{ij} - u_iv_j)^2, \tag{8.7}$$

The solution of this problem is carried out by the iterative "criss-cross" algorithm, using (8.4) and (8.5) with $\tilde{a}_{ij}$ instead of a_{ij}. Once $\tilde{\mathbf{u}}_{k+1}$ and $\tilde{\mathbf{v}}_{k+1}$ are computed, the matrix B_{k+1} is defined by the rule

$$B_{k+1} = B_k + \tilde{\mathbf{u}}_{k+1}\tilde{\mathbf{v}}_{k+1}^T. \tag{8.8}$$

For small values of k it is convenient to keep B_k in the factored form

$$B_k = \tilde{U}_k \tilde{V}_k^T = \sum_{j=1}^{k} \tilde{\mathbf{u}}_j \tilde{\mathbf{v}}_j^T, \tag{8.9}$$

where

$$\tilde{U}_k = [\tilde{\mathbf{u}}_1, \ldots, \tilde{\mathbf{u}}_k] \in \mathbb{R}^{m\times k} \quad \text{and} \quad \tilde{V}_k = [\tilde{\mathbf{v}}_1, \ldots, \tilde{\mathbf{v}}_k] \in \mathbb{R}^{n\times k}.$$

If A has no missing entries then the above scheme can be viewed as a variant of the deflation by subtraction method. The last method is often attributed to Hotelling [131]. Given a symmetric positive semidefinite matrix, the deflation by subtraction method computes the eigenpairs of the matrix, one after another in decreasing order, using the power method to compute dominant eigenpairs of the deflated matrices. For detailed discussions of this method see [211] and [280]. Recently Dax [55] has proposed a modified scheme for calculating an SVD-type decomposition of a general real $m \times n$ matrix A. The modified scheme generates orthogonal bases of range(A) and range(A^T) regardless to the number of Power iterations per eigenpair. Another related method is Wold's NIPALS algorithm, e.g., [75, 201, 232, 282, 283, 284]. In this scheme $\tilde{\mathbf{v}}_k$ is obtained by applying two Power iterations on $\tilde{A}_k^T \tilde{A}_k$, and the method results in an orthogonal sequence of vectors, which is identical to the sequence generated via Lanczos bidiagonalization, e.g., [75, 283].

However, when A has missing entries the orthogonality features are lost, and the basic deflation procedure (8.8) fails to generate a sequence of matrices $\{B_k\}$ such that B_k solves (1.1). Indeed, as demonstrated in [59], the larger the percentage of missing entries, the poorer solution we get. The algorithms described below enable us to overcome this drawback.

8.3 Successive rank-one modifications (SROM)

This iterative method is proposed in [59]. The ℓth iteration starts with W_ℓ and ends with $W_{\ell+1}$. The matrix W_ℓ is kept in the form

$$W_\ell = U_\ell V_\ell^T = \sum_{p=1}^{k} \mathbf{u}_p \mathbf{v}_p^T \tag{8.10}$$

where

$$U_\ell = [\mathbf{u}_1, \ldots, \mathbf{u}_k] \in \mathbb{R}^{m\times k} \quad \text{and} \quad V_\ell = [\mathbf{v}_1, \ldots, \mathbf{v}_k] \in \mathbb{R}^{n\times k}.$$

That is, here the vectors $\mathbf{u}_p \in \mathbb{R}^m$ and $\mathbf{v}_p \in \mathbb{R}^n$ denote the pth columns of U_ℓ and V_ℓ, respectively. The ℓth iteration of the SROM algorithm is composed of k steps. At the pth step, $p = 1, \ldots, k$, the vectors $\mathbf{u}_p$ and $\mathbf{v}_p$ alone are changed in an attempt to reduce the objective function value, while the other columns of U_ℓ and V_ℓ are kept fixed. The details of the pth step, $p = 1, \ldots, k$, are explained below.

The pth step

a) Let the matrix $\tilde{W} = \left(\tilde{w}_{ij}\right) \in \mathbb{R}^{m\times n}$ be defined by the equality

$$\tilde{W} = U_\ell V_\ell^T - \mathbf{u}_p \mathbf{v}_p^T = \sum_{\substack{q=1 \\ q\neq p}}^{k} \mathbf{u}_q \mathbf{v}_q^T.$$

b) Let the matrix $\tilde{A} = \left(\tilde{a}_{ij}\right) \in \mathbb{R}^{m\times n}$ be defined in the following way. $\tilde{a}_{ij} = a_{ij} - \tilde{w}_{ij}$ when a_{ij} is known. Otherwise, when a_{ij} is unknown, $\tilde{a}_{ij}$ is also unknown. That is, $\tilde{A}$ has the same pattern of missing entries as A.

c) The new values of the vectors $\mathbf{u}_p$ and $\mathbf{v}_p$ are those which solve the related rank-one approximation problem, which has the same form as (8.7). The last problem is solved by applying the "criss-cross" algorithm, using (8.4) and (8.5) with $\tilde{a}_{ij}$ instead of a_{ij}.

The iterative process in **c)** starts with the current values of $\mathbf{u}_p$ and $\mathbf{v}_p$. This ensures successive reduction of the objective function value. The nature of the SROM algorithm suggests that there is no need in accurate solution of the related rank-one approximation problem. Hence the pth step needs a small number of "criss-cross" iterations, see [59].

8.4 Alternating least squares (ALS)

This iterative algorithm is one of the popular methods for solving (1.1). The basic iteration is composed of two major steps. Let

$$W_\ell = X_\ell Y_\ell^T, \quad X_\ell \in \mathbb{R}^{m\times k}, \quad Y_\ell \in \mathbb{R}^{n\times k}, \tag{8.11}$$

denote the current solution at the beginning of the ℓth iteration, $\ell = 1, 2, \ldots$. Then $W_{\ell+1} = X_{\ell+1} Y_{\ell+1}^T$ is obtained in the following way.

Step 1: Define $X_{\ell+1}$ to be an $m \times k$ matrix that solves the problem

$$\begin{aligned} &\text{minimize} \quad F(B) \\ &\text{subject to} \quad B = XY_\ell^T \quad \text{and} \quad X \in \mathbb{R}^{m\times k}. \end{aligned} \tag{8.12}$$

Step 2: Define $Y_{\ell+1}$ to be an $n \times k$ matrix that solves the problem

$$\begin{aligned} &\text{minimize} \quad F(B) \\ &\text{subject to} \quad B = X_{\ell+1}Y^T \quad \text{and} \quad Y \in \mathbb{R}^{n\times k}. \end{aligned} \tag{8.13}$$

Note that in problem (8.12) the unknowns are the entries of the $m \times k$ matrix X. Thus, as explained below, the solution of (8.12) is achieved by computing the rows of X, one row at a time. Similarly, in (8.13) the unknowns are the entries of the $n \times k$ matrix Y, and the rows of this matrix are computed one row at a time.

Let $\mathbf{x}_i^T$ denote the ith row of X, $i = 1, \ldots, m$. Then $\mathbf{x}_i$ is computed in the following way. Observe that when solving (8.12) the ith row of B equals $\mathbf{x}_i^T Y_\ell^T$, and this row is compared

against $\mathbf{a}_i^T$, the ith row of A. The entries of $\mathbf{x}_i$ are determined, therefore, in an attempt to make $Y_\ell \mathbf{x}_i$ as close as possible to $\mathbf{a}_i$. This is achieved by solving the related linear least squares problem. Let $\tilde{n}$ denote the number of known entries in $\mathbf{a}_i$, and let the $\tilde{n}$-vector $\tilde{\mathbf{a}}_i$ be obtained from $\mathbf{a}_i$ by deleting the unknown entries in $\mathbf{a}_i$. That is, $\tilde{\mathbf{a}}_i$ is composed from the known entries of $\mathbf{a}_i$ in their original order. Let the $\tilde{n} \times k$ matrix $\tilde{Y}_\ell$ be obtained from Y_ℓ by deleting the rows of Y_ℓ which correspond to unknown entries in $\mathbf{a}_i$. Then $\mathbf{x}_i$ is determined by solving the linear least squares problem

$$\underset{\mathbf{x}\in\mathbb{R}^k}{\text{minimize}} \quad \|\tilde{Y}_\ell \mathbf{x} - \tilde{\mathbf{a}}_i\|_2^2. \tag{8.14}$$

If the solution of (8.14) is not unique then $\mathbf{x}_i$ is defined as the smallest solution of this problem. That is, $\mathbf{x}_i$ is defined as the unique solution of the minimum norm problem

$$\begin{aligned} &\text{minimize} && \|\mathbf{x}\|_2^2 \\ &\text{subject to} && \tilde{Y}_\ell^T \tilde{Y}_\ell \mathbf{x} = \tilde{Y}_\ell^T \tilde{\mathbf{a}}_i. \end{aligned}$$

The computation of $\mathbf{y}_j^T$, the jth row of $Y_{\ell+1}$, is carried out in a similar way. Observe that when solving (8.13) the jth column of B equals $X_{\ell+1}\mathbf{y}_j$, and this column is compared against $\mathbf{c}_j$, the jth column of A. Hence the entries of $\mathbf{y}_j$ are determined in an attempts to make $X_{\ell+1}\mathbf{y}_j$ as close as possible to $\mathbf{c}_j$. This is achieved by solving the related linear least squares problem. Let $\tilde{m}$ denote the number of known entries in $\mathbf{c}_j$, and let the $\tilde{m}$-vector $\tilde{\mathbf{c}}_j$ be obtained from $\mathbf{c}_j$ by deleting the unknown entries in $\mathbf{c}_j$. Let the $\tilde{m} \times k$ matrix $\tilde{X}_{\ell+1}$ be obtained from $X_{\ell+1}$ by deleting the rows of $X_{\ell+1}$ which correspond to missing entries in $\mathbf{c}_j$. Then $\mathbf{y}_j$ is defined as the solution of the least squares problem

$$\underset{\mathbf{y}\in\mathbb{R}^k}{\text{minimize}} \quad \|\tilde{X}_{\ell+1} \mathbf{y} - \tilde{\mathbf{c}}_j\|_2^2. \tag{8.15}$$

As before, if the solution of (8.15) is not unique then $\mathbf{y}_j$ is defined as the minimum norm solution of this problem.

At this point it is instructive to examine the similarity between the ALS algorithm and the SROM algorithm. Both methods use W_ℓ in a factored form. In the SROM method $W_\ell = U_\ell V_\ell^T$, while in the ALS method $W_\ell = X_\ell Y_\ell^T$. Yet the two factorizations play essentially the same role. The basic iteration of the ALS algorithm consists of a sweep over the rows of X_ℓ followed by a sweep over the rows of Y_ℓ, modifying one row at a time. In the SROM algorithm the basic iteration performs a sweep over the columns of U_ℓ and V_ℓ, where the pth step, $p = 1, \ldots, k$, modifies the pth columns of U_ℓ and V_ℓ. Thus both methods can be viewed as "block" variants of the alternating variables method. This type of minimization methods is also called block relaxation, and is known to have a linear asymptotic rate of convergence, see Schechter [238, 239, 240]. The similarity between SROM and ALS suggests that both methods converge at about the same speed. The experiments in [59] support this observation.

Note also that for $k = 1$ the ALS method coincides with SROM. In this case both methods are reduced to the criss-cross iteration (8.4)--(8.5). Thus, as we have seen, it can be "stuck" at a non-optimal point. Yet the simplicity of the ALS iteration makes it an effective tool for

solving (8.1). See, for example, the reports in [19, 33, 43, 44, 59, 77, 116, 186, 287]. For recent analysis of this method, see [136].

A standard method for solving (8.14) requires an order of $\tilde{n}k^2$ flops (floating point operations) while solving (8.15) needs an order of $\tilde{m}k^2$ flops, e.g., [23, 110, 253]. The overall number of flops which are needed in one ALS iteration is, therefore, about $2mnk^2$. On the other hand, the number of flops required in one SROM iteration is about $2mnk\hat{\ell}$, where $\hat{\ell}$ denotes the average number of "criss-cross" iterations per step. Thus when $k < \hat{\ell}$ the ALS iteration requires less flops.

The current version of the ALS iteration computes minimum norm solutions of (8.14) and (8.15). This modification has two advantages. First it enables us to handle cases in which $\tilde{Y}_\ell$ or $\tilde{X}_{\ell+1}$ are rank deficient. Second, it "pushes" the computed solution toward a certain minimum norm solution of (1.1). The last statement needs some explanation. Assume for a moment that the current matrix, $W_\ell = X_\ell Y_\ell^T$, solves (1.1). Then the next ALS iterations are unable to reduce the objective function value, but the value of the sum $\|X_\ell\|_F^2 + \|Y_\ell\|_F^2$ is strictly decreased at each iteration, unless $X_{\ell+1} = X_\ell$ and $Y_{\ell+1} = Y_\ell$. As proved in Section 10, the nuclear norm $\|\cdot\|_*$ satisfies

$$\|X_\ell Y_\ell^T\|_* \leq \ {}_{1/2}(\|X_\ell\|_F^2 + \|Y_\ell\|_F^2).$$

Hence decreasing the right hand side of this inequality provides a smaller bound on the nuclear norm of the solution matrix.

Moreover, assume further that $F(W_\ell) = 0$. Then from now on $W_\ell = X_\ell Y_\ell^T$ is an admissible matrix. In this case the above decreasing property pushes the matrices X_ℓ and Y_ℓ toward a solution of the problem

$$\begin{array}{ll} \text{minimize} & \varphi(X,Y) = \|X\|_F^2 + \|Y\|_F^2 \\ \text{subject to} & X \in \mathbb{R}^{m\times k}, \ \ Y \in \mathbb{R}^{n\times k}, \quad \text{and} \quad XY^T \in \mathbb{A}. \end{array} \tag{8.16}$$

As we shall see in Section 10, the last problem is essentially the nuclear norm problem (10.1) Note also that both SROM and ALS are easily modified to solve the regularized least squares problem (8.17). This gives another way to obtain solutions with small nuclear norm. It is also worthwhile noting that ALS is easily adapted to run on a parallel computer, e.g., [151, 152].

8.5 Non-uniqueness of least squares solutions

A further inspection of the ALS iteration reveals situations in which the solution of (1.1) loses its uniqueness. As above, the matrices W_ℓ, X_ℓ and Y_ℓ, satisfy (8.11) but here it is assumed that W_ℓ solves (1.1). Observe that there is no loss of generality in assuming that Y_ℓ has orthonormal columns. To verify this observation use an SVD of the form

$$W_\ell = \hat{U}\hat{\Sigma}\hat{V}^T, \quad \text{where } \hat{U} \in \mathbb{R}^{m\times k}, \ \hat{\Sigma} \in \mathbb{R}^{k\times k}, \quad \text{and} \quad \hat{V} \in \mathbb{R}^{n\times k}.$$

Then define $X_\ell = \hat{U}\hat{\Sigma}$ and $Y_\ell = \hat{V}$. Also, since W_ℓ solves (1.1), applying the ALS iteration on these matrices does not change the objective function value.

Next consider the least squares problem (8.14) and assume further that $\tilde{n} < k$. That is, the number of known entries in $\mathbf{a}_i$ is smaller than k. In this case the matrix $\tilde{Y}_\ell$ has less rows than columns, Null$(\tilde{Y}_\ell)$ is a non-trivial subspace of $\mathbb{R}^k$, and (8.14) has infinitely many solutions. More precisely, let $\mathbf{x}^*$ denote the minimum norm solution of (8.14). Then $\mathbf{x}^* \in$ Range$(\tilde{Y}_\ell^T)$ and the set of all the points that solves (8.14) form the linear manifold

$$\{\mathbf{x}^* + \mathbf{z} \mid \mathbf{z} \in \text{Null}(\tilde{Y}_\ell)\}.$$

Note also that the ith row of the matrix X_ℓ solves (8.14). Hence replacing this row by another solution of (8.14) does not change entries in the ith row of W_ℓ which correspond to known entries of $\mathbf{a}_i$. On the other hand, such a replacement changes some of the other entries in that row. (Entries of the ith row of W_ℓ which correspond to unknown entries of $\mathbf{a}_i$.) The last assertion is deduced from the following observations. Since $\tilde{n} < k$ we also have

$$\text{rank}(\tilde{Y}_\ell) \le \tilde{n} < k = \text{rank}(Y_\ell).$$

This means that some rows of Y_ℓ, rows which correspond to unknown entries of $\mathbf{a}_i$, do not belong to Range$(\tilde{Y}_\ell^T)$. Thus at least one of these rows must have a non-zero component in Null$(\tilde{Y}_\ell)$.

The above discussion leads to the following useful conclusion. Let the data matrix, A, have a row with $\tilde{n}$ known entries. If $\tilde{n} < k$ then both (8.14) and (1.1) have infinitely many solutions. A similar conclusion is derived from (8.15). Let A have a column with $\tilde{m}$ known entries. If $\tilde{m} < k$ then both (8.15) and (1.1) have infinitely many solutions. These observations have simple interpretation in the context of Netflix-like matrices: To determine the "taste" of a viewer we need at least k rates from that user. Similarly, to "characterize" a movie we need at least k ratings of that movie. The next statement summarizes our findings.

Theorem 2. *Let ν_{min} denote the smallest number of known entries in one row or one column of A. If $\nu_{min} < k$ then (1.1) has infinitely many solutions.*

Observe that ν_{min} provides an upper bound on the value of k for which (1.1) has a unique solution. Summing the condition on $\tilde{n}$ over the rows (the columns) of A leads to the following conclusion: If

$$k > \min\{\nu/m,\ \nu/n,\ 2\nu/(m+n)\}$$

then there exists a row, or a column, in which the number of known entries is smaller than k. Hence in this case (1.1) has infinitely many solutions. Recall that ν denotes the number of known entries in A. We see, therefore, that the ratio between ν and $k(m+n)$ affects the possibility to have a unique solution. We will return to discuss this issue in Sections 9 and 10, in the context of exact recovery.

8.6 Regularized least squares problems

In this approach (8.1) is replaced with the problem

$$\begin{aligned} \text{minimize} \quad & R_\lambda(U,V) = F(U,V) + \lambda(\|U\|_F^2 + \|V\|_F^2) \\ \text{subject to} \quad & U \in \mathbb{R}^{m\times k} \quad \text{and} \quad V \in \mathbb{R}^{n\times k}, \end{aligned} \tag{8.17}$$

where $\lambda > 0$ is a preassigned regularization constant. The optimal value of this problem can't exceed the value of $R_\lambda(0,0) = \|P_\Omega(A)\|_F^2$. Therefore the search for a minimizer of (8.17) can be restricted to the ball

$$\{\ (U,V)\ \big|\ \|U\|_F^2 + \|V\|_F^2 \le \|P_\Omega(A)\|_F^2/\lambda\ \},$$

which ensures the existence of a point $(U_\lambda^*, V_\lambda^*)$ that solves (8.17), and that any solution of this problem lies within this ball. The idea of regularization comes from the solution of ill-conditioned linear least squares problems, e.g., [23, 110]. In statistics it is called "ridge regression". Roughly speaking, it is aimed to result in an improved problem which is more convex and less ill-conditioned. Thus when using a descent method to solve (8.17) it has better chances to achieve fast converges toward a local minimizer. Note that in Netflix-like problems the regularized model (8.17) reflects a "parsimonious" attitude toward rating.

A further insight into the nature of regularized solutions is gained from the following considerations. Assume for a moment the existence of a point (U^*, V^*) that solves (8.1), and let the point $(U_\lambda^*, V_\lambda^*)$ solve (8.17). Then, clearly,

$$R_\lambda(U_\lambda^*, V_\lambda^*) \le R_\lambda(U^*, V^*),$$

and

$$F(U_\lambda^*, V_\lambda^*) + \lambda(\|U_\lambda^*\|_F^2 + \|V_\lambda^*\|_F^2) \le F(U^*, V^*) + \lambda(\|U^*\|_F^2 + \|V^*\|_F^2).$$

Furthermore, since (U^*, V^*) solves (8.1),

$$0 \le F(U_\lambda^*, V_\lambda^*) - F(U^*, V^*) \le \lambda[(\|U^*\|_F^2 + \|V^*\|_F^2) - (\|U_\lambda^*\|_F^2 - \|V_\lambda^*\|_F^2)],$$

and

$$\|U_\lambda^*\|_F^2 + \|V_\lambda^*\|_F^2 \le \|U^*\|_F^2 + \|V^*\|_F^2, \tag{8.18}$$

while the last bound implies the limit

$$\lim_{\lambda \to 0^+} F(U_\lambda^*, V_\lambda^*) = F(U^*, V^*). \tag{8.19}$$

The bound (8.18) holds for any point (U^*, V^*) that solves (8.1). It is possible, therefore, to lower this bound by choosing a solution of (8.1) that has the smallest norm. This brings us to consider the following minimum norm problem.

$$\begin{aligned} &\text{minimize} \quad \varphi(U,V) = {}_{1/2}(\|U\|_F^2 + \|V\|_F^2) \\ &\text{subject to} \quad F(U,V) \le F(U^*, V^*). \end{aligned} \tag{8.20}$$

A point that solves this problem is said to be a minimum norm solution of (8.1). The search for such a point can be restricted to the ball

$$\{(U,V)\ |\ \|U\|_F^2 + \|V\|_F^2 \le \|U^*\|_F^2 + \|V^*\|_F^2\},$$

which ensures the existence of a point $(\tilde{U}, \tilde{V})$ that solves both (8.1) and (8.20). Replacing (U^*, V^*) with $(\tilde{U}, \tilde{V})$ in (8.18) leads to the following conclusion.

Theorem 3. *The inequality*

$$\|U_\lambda^*\|_F^2 + \|V_\lambda^*\|_F^2 \le \|\tilde{U}\|_F^2 + \|\tilde{V}\|_F^2 \tag{8.21}$$

holds for any $\lambda > 0$. That is, the norm of a regularized solution is smaller than that of a minimum norm solution.

The last theorem has a useful consequence: We see that for small values of λ a regularized solution $(U_\lambda^*, V_\lambda^*)$ provides a valuable substitute for a minimum norm solution of (8.1). In Section 10.1.1 we shall see that (8.20) is equivalent to the nuclear norm problem

$$\begin{aligned} &\text{minimize} \quad \|B\|_* \\ &\text{subject to} \quad B \in \mathbb{B}_k \quad \text{and} \quad F(B) \le F^*, \end{aligned} \tag{8.22}$$

where $F^* = F(U^*, V^*)$ denotes the optimal value of (1.1). That is, both problems share the same optimal value, and a solution of one problem provides a solution to the other problem. Moreover, as proved in Theorem 11, if k exceeds a certain value then $F^* = 0$ and (8.20) is equivalent to the nuclear problem (10.1). In this case, when k is large enough, solving (8.17) with a small λ provides an approximate solution of (10.1).

The replacement of (8.1) with (8.17) is advocated by several authors. For example, a Newton method for solving (8.17) is proposed in [33], Gradient Descent methods are considered in [92, 152, 209, 212], and the use of ALS to solve this problem is discussed in [33] and [152]. In fact, almost all the methods presented in this section are easily modified to solve (8.17) instead of (8.1). However, we skip the details for the sake of brevity. Different types of regularization are used in OptSpace [145] and JELLYFISH [226], see Section 8.8 below.

8.7 Proximal point methods: ALS and LMaFit

The Iterative SVD method is an example of a proximal point method that is aimed at solving (1.1). Casting (1.1) in the factored form (8.1) suggests that the method can be modified to take advantage of this form by replacing the SVD computation with a less expensive procedure. In the new setting the rank-k TSVD matrix is replaced by the product matrix $U_\ell V_\ell^T$, where $U_\ell \in \mathbb{R}^{m\times k}$ and $V_\ell \in \mathbb{R}^{n\times k}$. As before, the related admissible matrix is denoted by A_ℓ. Recall that the entries of A_ℓ which correspond to unknown entries of A obtain the values of the corresponding entries in the matrix $U_\ell V_\ell^T$. Using the matrix operator P_Ω the last statement can be expressed as

$$A_\ell = P_\Omega(A) + \left[U_\ell V_\ell^T - P_\Omega(U_\ell V_\ell^T)\right] = U_\ell V_\ell + P_\Omega(A - U_\ell V_\ell^T).$$

The proximal problem that corresponds to A_ℓ has the form

$$\begin{aligned} &\text{minimize} \quad \Pi_\ell(U,V) = \|A_\ell - UV^T\|_F^2 \\ &\text{subject to} \quad U \in \mathbb{R}^{m\times k} \ \text{ and } V \in \mathbb{R}^{n\times k}, \end{aligned} \tag{8.23}$$

and the related proximal functions satisfy

$$\Pi_\ell(U_\ell, V_\ell) = F(U_\ell, V_\ell)$$

and

$$\Pi_\ell(U,V) \geq F(U,V) \quad \forall\ U \in \mathbb{R}^{m\times k}, \quad V \in \mathbb{R}^{n\times k}.$$

The ℓth iteration, $\ell = 1, 2, \ldots$, starts with the triplet U_ℓ, V_ℓ, and A_ℓ. Then the matrices $U_{\ell+1}$ and $V_{\ell+1}$ are computed in an attempt to solve (8.23). There is no need in computing an accurate solution, but the computed matrices are restricted to satisfy

$$\Pi_\ell(U_\ell, V_\ell) > \Pi_\ell(U_{\ell+1}, V_{\ell+1}),$$

as this restriction forces the descent property

$$F(U_\ell, V_\ell) = \Pi_\ell(U_\ell, V_\ell) > \Pi_\ell(U_{\ell+1}, V_{\ell+1}) \geq F(U_{\ell+1}, V_{\ell+1}).$$

In practice the proximal problem can be solved by any minimization method. Then using (U_ℓ, V_ℓ) as starting point is likely to ensure the descent property. Once $U_{\ell+1}$ and $V_{\ell+1}$ are computed, $A_{\ell+1}$ is defined to be the related admissible matrix.

The Proximal-ALS algorithm

To illustrate the above ideas we consider a simple algorithm in which the proximal problem is solved by applying one ALS iteration. In this implementation $U_{\ell+1}$ is obtained by solving the problem

$$\underset{U\in\mathbb{R}^{m\times k}}{\text{minimize}}\ \Psi(U) = \|A_\ell - UV_\ell^T\|_F^2.$$

Then $V_{\ell+1}$ is computed by solving the problem

$$\underset{V\in\mathbb{R}^{n\times k}}{\text{minimize}}\ \varphi(V) = \|A_\ell - U_{\ell+1}V^T\|_F^2.$$

Using the pseudoinverses of V_ℓ and $U_{\ell+1}$, the Proximal-ALS iteration is summarized as follows.

$$U_{\ell+1} = A_\ell (V_\ell^\dagger)^T,$$
$$V_{\ell+1}^T = U_{\ell+1}^\dagger A_\ell,$$

and

$$A_{\ell+1} = P_\Omega(A) + \left[U_{\ell+1}V_{\ell+1}^T - P_\Omega(U_{\ell+1}V_{\ell+1}^T)\right].$$

The practical solution of the above pair of problems can be done with two QR factorizations. First a QR factorization of V_ℓ is used to compute the rows of $U_{\ell+1}$. Then a QR factorization of $U_{\ell+1}$ is used to compute the rows of $V_{\ell+1}$. Recall that the ALS iteration (8.12)-(8.13) requires a different QR factorization for nearly every row of $U_{\ell+1}$ and $V_{\ell+1}$. Hence the Proximal-ALS iteration achieves a considerable simplicity and saving.

The LMaFit algorithm

A further saving is achieved in the "LMaFit" algorithm proposed by Wen et al. [277]. Here the basic Proximal-ALS iteration is carried out with only one QR factorization which is used to generate an orthonormal basis to the column space of the matrix $A_\ell V_\ell$. In this iteration $U_{\ell+1}$ is defined to be the resulting orthonormal basis. Then, since $U_{\ell+1}$ has orthonormal columns, $V_{\ell+1}$ is given by the equality $V_{\ell+1}^T = U_{\ell+1}^T A_\ell$. The equivalence of the two iterations follows from the observation that

$$\text{Range}(U_{\ell+1}) = \text{Range}\Big(A_\ell (V_\ell^\dagger)^T\Big) = \text{Range}(A_\ell V_\ell).$$

This equality allows us to replace $U_{\ell+1}$ with an orthonormal basis of Range$(A_\ell V_\ell)$. If V_ℓ has full column rank, then $V_\ell^\dagger = (V_\ell^T V_\ell)^{-1} V_\ell^T$ and the claim is straightforward. Otherwise, when the columns of V_ℓ are linearly dependent, the proof is concluded by using the SVD of V_ℓ, see [277].

Another modification that is introduced in LMaFit [277] is the use of an SOR-like acceleration. Let $w \geq 1$ denote the SOR parameter, then the formal description of the accelerated iteration is as follows:

$$\begin{aligned} U_{\ell+1} &= A_\ell (V_\ell^\dagger)^T, \\ U_{\ell+1}(w) &= w U_{\ell+1} + (1-w) U_\ell, \\ V_{\ell+1} &= U_{\ell+1}^\dagger(w) A_\ell, \\ V_{\ell+1}(w) &= w V_{\ell+1} + (1-w) V_\ell, \end{aligned}$$

and

$$A_{\ell+1}(w) = P_\Omega(A) + \Big[U_{\ell+1}(w) V_{\ell+1}^T(w) - P_\Omega\big(U_{\ell+1}(w) V_{\ell+1}^T(w)\big)\Big].$$

The LMaFit algorithm implements this updating with a single QR factorization. The SOR parameter is adjusted in every iteration by some "trial and error" procedure, where the final value in one iteration serves as the starting value in the next iteration. See [277] for more details.

8.8 Gradient descent methods: OptSpace, Funk's scheme, and JELLYFISH

The gradient vector of $F(U,V)$ at a given point, (U,V), can be expressed as a pair of matrices, $U' = (u'_{ij}) \in \mathbb{R}^{m\times k}$ and $V' = (v'_{ij}) \in \mathbb{R}^{n\times k}$, where

$$u'_{ij} = \frac{\partial F}{\partial u_{ij}}\Big|_{(U,V)} \quad \text{and} \quad v'_{ij} = \frac{\partial F}{\partial v_{ij}}\Big|_{(U,V)}.$$

Using these notations it is easy to verify that

$$U' = -2[P_\Omega(A) - P_\Omega(UV^T)]V$$

and

$$V' = -2[P_\Omega(A) - P_\Omega(UV^T)]^T U.$$

Therefore making a small step in the steepest descent direction, $-(U', V')$, is likely to reduce the objective function value. This idea motivates the gradient descent method. Starting from a given initial point (U_0, V_0), it generates a sequence of points $(U_\ell, V_\ell), \quad \ell = 1, 2, \ldots$, where $(U_{\ell+1}, V_{\ell+1})$ is obtained from (U_ℓ, V_ℓ) by the following rule:

$$U_{\ell+1} = U_\ell + \theta[P_\Omega(A) - P_\Omega(U_\ell V_\ell^T)]V_\ell, \tag{8.24a}$$

and

$$V_{\ell+1} = V_\ell + \theta[P_\Omega(A) - P_\Omega(U_\ell V_\ell^T)]^T U_\ell, \tag{8.24b}$$

where $\theta > 0$ is a step-length parameter.

The basic iteration is often modified to allow a different step-size for each iteration, which is determined by some "line-search" procedure. Below we consider further modifications of the basic scheme. The first method has the flavor of minimization on Grassmann manifolds, but the discussion of this issue is deferred until Section 8.10.

OptSpace: The use of orthogonal factors

The OptSpace algorithm is proposed by Keshavan and Oh in [145]. As we have seen, it is possible to assume that the matrices U and V have orthogonal columns. This converts (8.1) into the problem

$$\begin{aligned} \text{minimize} \quad & F(U,S,V) = \|P_\Omega(A) - P_\Omega(USV^T)\|_F^2 \\ \text{subject to} \quad & U \in \mathbb{R}^{m\times k}, \;\; S \in \mathbb{R}^{k\times k}, \;\; V \in \mathbb{R}^{n\times k}, \\ & U^TU = mI \;\; \text{and} \;\; V^TV = nI, \end{aligned}$$

which is solved by OptSpace. The main idea behind OptSpace is that for a given pair of orthogonal matrices, U and V, the optimal value of S is obtained by solving the problem

$$\underset{S\in\mathbb{R}^{k\times k}}{\text{minimize}} \; \|P_\Omega(A) - P_\Omega(USV^T)\|_F^2.$$

The last problem is essentially a linear least squares problem: The linear system to solve has k^2 unknowns and ν equations. The unknowns are the entries of S and the equations are derived from the desired equalities

$$\mathbf{u}_i^T S \mathbf{v}_j = a_{ij} \quad \forall (i,j) \in \Omega.$$

This way, given U_ℓ and V_ℓ, the OptSpace algorithm computes the related optimal matrix, S_ℓ, by generating and solving the relevant linear least squares problem. Now the gradient descent step (8.24) implies the update

$$U_{\ell+1} = U_\ell + \theta\Big[P_\Omega(A) - P_\Omega(U_\ell S_\ell V_\ell^T)\Big]V_\ell S_\ell^T$$

and

$$V_{\ell+1} = V_\ell + \theta\Big[P_\Omega(A) - P_\Omega(U_\ell S_\ell V_\ell^T)\Big]^T U_\ell S_\ell,$$

which destroys the orthogonality of these matrices. Hence, before starting the next iteration, the algorithm orthogonalizes the columns of $U_{\ell+1}$ and $V_{\ell+1}$. Then $S_{\ell+1}$ is computed from the rule

$$S_{\ell+1} = \arg\min_S F(U_{\ell+1}, S, V_{\ell+1}).$$

The proposed OptSpace iteration includes a bisection type line-search algorithm, as well as safeguards that help to handle ill-conditioned matrices.

The way OptSpace achieves regularization differs from the one discussed in Section 8.6. Here the regularizing term takes the form

$$\lambda \sum_{(i,j)\notin\Omega} (\mathbf{u}_i^T S \mathbf{v}_j)^2$$

where λ is a preassigned positive constant. This way large values of λ force the completed entries to be small. A second way to achieve regularization in OptSpace has the form $\lambda\|S\|_F^2$. If $U^TU = mI$ and $V^TV = nI$, as in OptSpace, then

$$\|S\|_F = \|USV^T\|_F\big/(mn).$$

Hence this regularization seeks a low-rank approximation with small Frobenius norm.

Another feature which characterizes OptSpace is the use of a starting procedure that determines the value of k and provides a starting point (U_0, S_0, V_0). The algorithm achieves these goals by conducting a "trimming" process that converts $P_\Omega(A)$ into a matrix $T \in \mathbb{R}^{m\times n}$. Then the SVD of T is used to determine k, and the resulting rank-k Truncated SVD of T defines the starting point. A row (column) of $P_\Omega(A)$ is said to be over-presented if the number of known entries in this row (column) is more than twice the average. The trimming process set to zero all the entries in over-presented rows and columns. For detailed discussions of the above ideas see [144] and [145].

Funk's scheme: Taking advantage of sparsity

In "Netflix like" problems both $P_\Omega(A)$ and $P_\Omega(UV^T)$ are huge and highly sparse matrices. Hence the basic gradient descent iteration is modified to take advantage of this feature. The idea comes from the equality

$$(U', V') = \sum_{(i,j)\in\Omega} (U'_{ij}, V'_{ij}),$$

where (U'_{ij}, V'_{ij}) denotes the gradient vector of the function

$$f_{ij}(U, V) = (a_{ij} - \mathbf{u}_i^T \mathbf{v}_j)^2.$$

Note that the ith row of U'_{ij} equals $-2(a_{ij} - \mathbf{u}_i^T\mathbf{v}_j)\mathbf{v}_j^T$, while all the other rows of this matrix are null rows. Similarly, the jth row of V'_{ij} equals $-2(a_{ij} - \mathbf{u}_i^T\mathbf{v}_j)\mathbf{u}_i^T$, and all the other rows of this matrix are null rows. Therefore the increment $-\theta(U', V')$ can be computed from that sum. A further saving is achieved by adding the matrices $-\theta(U'_{ij}, V'_{ij})$ directly to the current solution point (U, V). (For the sake of simplicity we omit the iteration index.) This way the basic iteration consists of one sweep over the known entries of A, where for each index pair $(i, j) \in \Omega$ we do as follows.

a) Compute $\eta = \theta(a_{ij} - \mathbf{u}_i^T\mathbf{v}_j)$.
b) Set the ith row of U to be $\mathbf{u}_i + \eta\mathbf{v}_j$.
c) Set the jth row of V to be $\mathbf{v}_j + \eta\mathbf{u}_i$.

This elegant scheme is often attributed to Funk [92]. The modified iteration is very simple and requires minimal amounts of storage and flops. Apart from the basic information on the known entries of A, it uses only the two matrices U and V. The basic iteration is carried out with $3k\nu + 1$ multiplications, and a similar number of additions. The simplicity of Funk's scheme makes it a useful tool for solving large Netflix-like problems, e.g., [92, 151, 152, 209, 212]. A further advantage lies in the ability to achieve parallel computing.

JELLYFISH: Adaptation to parallel computing

The JELLYFISH algorithm [226] is designed to run on a parallel computer with a number of processors. It is based on a modified version of Funk's iteration in which the computational effort is equally partitioned between the processors. On large scale problems this admits a speed-up nearly proportional to the number of processors. The partition of the work between the processors is based on the following idea. Assume for a moment that we have two processors, say P_1 and P_2. Then the data matrix, A, is split into four submatrices,

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix},$$

where all the submatrices have nearly the same number of rows and columns. The submatrix A_{11} has the same number of rows as A_{12}, and the same number of columns as A_{21}. In this setting the basic iteration consists of two "rounds". In the first round P_1 considers the (known) entries of A_{11}, and P_2 considers the entries in A_{22}. In the second round P_1 considers the entries of A_{12}, and P_2 considers the entries in A_{21}. This way the work is equally partitioned between the two processors. Assume now that our computer has three processors. In this case A is partitioned into three rows and three columns of submatrices, $A_{ij},\ \ i = 1, 2, 3, \quad j = 1, 2, 3$. As before, submatrices that belong to the same row have the same number of rows, and submatrices that belong to the same column have the same number of columns. Here the basic iteration consists of three "rounds". The first round considers the matrices A_{11}, A_{22}, A_{33}. The second one considers A_{12}, A_{23}, A_{31}, and the third round considers A_{13}, A_{21}, A_{32}. At each round each processor is assigned a different submatrix. This way, when executing a specific round, entries that "belong" to different processors are located in different rows and columns of A, and relate to different rows of U and V. Hence the update of U and V, according to Funk's

formula, can be done in parallel, sharing the work between the three processors. The extension to p processors is done in a similar way. Note that there is a simple rule for generating the related p rounds. The first round considers the submatrices $A_{11}, A_{22}, \ldots, A_{pp}$. The second round considers the submatrices $A_{12}, A_{23}, \ldots, A_{p-1,p}, A_{p,1}$, and so forth.

In JELLYFISH the partition of the data into submatrices is changing every iteration in a random way. The basic iteration starts by randomly generating two permutations matrices, say $\Pi_{\text{row}} \in \mathbb{R}^{m\times m}$ and $\Pi_{\text{col}} \in \mathbb{R}^{n\times n}$. Then the partition of the data into submatrices is done by replacing A with $\Pi_{\text{row}} A \Pi_{\text{col}}$. The permutation matrices are generated by applying the "Knuth shuffle" algorithm. The use of permutations is aimed to improve the performance of the algorithm. See the discussion of this issue in [226].

The JELLYFISH algorithm is designed to minimize two types of regularized least squares problems. Both ways differ from the standard regularization (8.17). In the first option the regularizing term has the form

$$\lambda\Big(\sum_{i=1}^{m} \|\mathbf{u}_i\|_2^2/\rho_i + \sum_{j=1}^{n} \|\mathbf{v}_j\|_2^2/\eta_j\Big),$$

where $\mathbf{u}_i$ and $\mathbf{v}_j$ are defined as in (8.1), ρ_i denotes the number of known entries in the ith row of A, and η_j denotes the number of known entries in the jth column of A. That is the smaller the number of known entries in a certain row or column the larger its weight. In the second option the regularized problem has the form

$$\begin{aligned} \text{minimize} \quad & F(U,V) = \sum_{(ij)\in\Omega} (a_{ij} - \mathbf{u}_i^T \mathbf{v}_j)^2 \\ \text{subject to} \quad & \|\mathbf{u}_i\|_2^2 \le \beta \ \text{ for } \ i = 1,\ldots,m, \\ \text{and} \quad & \|\mathbf{v}_j\|_2^2 \le \beta \ \text{ for } \ j = 1,\ldots,n, \end{aligned}$$

where $\beta > 0$ is a preassigned bound.

The basic iteration does not include a line-search algorithm. The experiments in [226] are carried out with a variable step-size parameter, α, that changes every iteration. In random test problems the starting value is $\alpha = 0.1$. Then α is reduced by a factor of 0.9 after each iteration. In collaborative filtering experiments the step-size is initialized at $\alpha = 0.05$ and is reduced by a factor of 0.8 after each iteration.

The JELLYFISH algorithm has a unique way to determine k, the number of columns in U and V. The starting value of k is defined as

$$k = \nu/[3(m+n)],$$

where ν denotes the overall number of known entries. This choice is aimed to guarantee a positive definite Hessian at the solution point. Yet the resulting value of k might be "too large". Hence, after a few iterations, the algorithm makes an inspection that estimates the rank of the current approximating matrix, UV^T. If the estimated rank is smaller than k, the value of k

is reduced to this smaller rank, and the algorithm rerun from scratch with the smaller rank. In order to assess the rank of the product matrix UV^T the algorithm computes approximate SVDs of U and V. These approximations are obtained by sampling $2k \log k$ rows of U (of V) and computing exact SVDs of the smaller matrices. The reader is referred to Recht and Re [226] for detailed discussions of the above ideas.

8.9 Fast converging methods: Newton, Gauss-Newton and Wiberg

In this section we briefly overview a number of fast converging methods for solving (8.1). As noted before, the objective function to minimize is the sum of squares of ν nonlinear equations in $k(m+n)$ unknowns. Traditional methods for solving nonlinear least squares problems include Newton's method (for minimization) and the Gauss-Newton method, e.g., [65, 84, 99]. Yet, as we have seen, the related Jacobian and Hessian matrices can be rank-deficient. The usual ways to resolve rank-deficiency are the Levenberg-Marquardt (L-M) modification, the trust-region approach, and regularization.

Explicit formulae for calculating the Hessian matrix of $F(U,V)$ can be found in [33]. The last paper considers the use of an L-M Newton method for solving (8.17). The Hessian matrix of $F(U,V)$ is a symmetric $p \times p$ matrix, where $p = k(m+n)$. So the basic iteration of Newton's method involves the solution of a symmetric $p \times p$ linear system, which requires about $0(p^3/6)$ flops. Thus, Newton's iteration is expected to be far more "expensive" than the ALS iteration. This observation motivates the use of a "hybrid" method which starts by performing a fixed number of ALS iterations, then switch to Newton's iterations, see [33]. The discussion of hybrid methods is deferred to Section 8.13.

We have seen that for any $n \times k$ matrix, V, there exists an "optimal" $m \times k$ matrix, $U = U(V)$, that solves the problem

$$\underset{U \in \mathbb{R}^{m \times k}}{\text{minimize}}\ \Psi(U) = F(U,V).$$

If all the entries of A are known then the solution is simply $U = (V^{\dagger}A^T)^T$, where $V^{\dagger}$ denotes the pseudoinverse of V. Otherwise, when A has missing entries, the solution matrix is computed row after row, as in the ALS iteration. The relation $U = U(V)$ enables us to rewrite (8.1) in the form

$$\underset{V \in \mathbb{R}^{n \times k}}{\text{minimize}}\ G(V) = F(U(V),V),$$

where here the unknowns to find are the entries of V. More precisely, let $\tilde{\mathbf{a}}_i$ be defined as in (8.14) and let the matrix $\tilde{V}_i$ be obtained from V by deleting the rows of V which correspond to missing entries in the ith row of A. Then the ith row of the matrix $U = U(V)$ equals $(\tilde{V}_i^{\dagger}\tilde{\mathbf{a}}_i)^T$ and

$$G(V) = \sum_{(i,j)\in\Omega} \left((\tilde{V}_i^{\dagger}\tilde{\mathbf{a}}_i)^T \mathbf{v}_j - a_{ij} \right)^2, \tag{8.25}$$

where $\mathbf{v}_j$ denotes the jth row of V. A similar reduction is possible by expressing V as a function of U.

Least squares problems of this form, whose variables can be separated into two groups such that the optimal value of one group is easily computed for any given value of the other group, are called "separable". The use of Gauss-Newton algorithms for solving separable nonlinear least squares problems is studied in [23, 108, 109, 142, 234], but these works don't concentrate on problems with missing entries.

Applying the Gauss-Newton method to solve the reduced problem (8.25) results in the Wiberg algorithm, e.g., [43, 208, 232, 278]. In this approach the objective function, $G(V)$, is expressed as the sum of squares of a residual vector function, $\mathbf{r}(V)$, that has ν components; and the method requires the Jacobian matrix of $\mathbf{r}(V)$. For detailed description and discussion of Wiberg's algorithm see [208].

The use of an L-M Newton method for minimizing $G(V)$ is proposed by P. Chen [43]. Since V is an $n\times k$ matrix, the gradient vector of $G(V)$ has nk components, while the Hessian of $G(V)$ is a symmetric $nk \times nk$ matrix. So here Newton's iteration involves the solution of a symmetric $nk \times nk$ linear system. A second approach considered in [43] is the Newton-Grassmann method.

8.10 Newton-Grassmann methods

The set of all k-dimensional subspaces in $\mathbb{R}^n$ is called the Grassmann manifold $\mathbb{G}_k^n$. Let $[V]$ denote the subspace of $\mathbb{R}^n$ which is spanned by the columns of V. Since $V \in \mathbb{R}^{n\times k}$ and V is assumed to have full column rank, $[V]$ is a point on $\mathbb{G}_k^n$. Furthermore, let T be any invertible matrix in $\mathbb{R}^{k\times k}$, then the equalities $G(V) = G(VT)$ and $[V] = [VT]$ imply that our objective function is defined on points of $\mathbb{G}_k^n$.

In this section we consider the use of Newton method to minimize $G(V)$ on the Grassmann manifold $\mathbb{G}_k^n$. We use an iterative algorithm that generates a sequence of matrices $V_\ell \in \mathbb{R}^{n\times k}$, $\ell = 1, 2, \ldots$, such that $G(V_{\ell+1}) \le G(V_\ell)$. The difference between V_ℓ and $V_{\ell+1}$ is denoted by Z_ℓ. That is, $V_{\ell+1} = V_\ell + Z_\ell$ for some matrix $Z_\ell \in \mathbb{R}^{n\times k}$. Let $[V_\ell]_\perp = \text{Null}(V_\ell^T)$ denote the orthogonal complement of $[V_\ell]$. Then Z_ℓ has unique presentation in the form $Z_\ell = \tilde{Z}_\ell + \hat{Z}_\ell$, where the columns of $\tilde{Z}_\ell$ belong to $[V_\ell]$ and the columns of $\hat{Z}_\ell$ belong to $[V_\ell]_\perp$. Therefore, since $[V_\ell + \tilde{Z}_\ell] = [V_\ell]$, the change in the value of $G(V)$ is caused by $\hat{Z}_\ell$ alone. This observation allows us to restrict the choice of Z_ℓ to matrices in $\mathbb{R}^{n\times k}$ whose columns belong to $[V_\ell]_\perp$. Moreover, let $N_\ell \in \mathbb{R}^{n\times(n-k)}$ be a given matrix whose columns form a basis of $[V_\ell]_\perp$. Then it is possible to assume that Z_ℓ has the form $N_\ell L_\ell$ for some matrix $L_\ell \in \mathbb{R}^{(n-k)\times k}$. Therefore, once N_ℓ is computed, the increment matrix, $Z_\ell = N_\ell L_\ell$, is found by searching a matrix $L_\ell \in \mathbb{R}^{(n-k)\times k}$ that attempt to minimize the objective function $H_\ell(L) = G(V_\ell + N_\ell L)$. The second method proposed in [43] defines L_ℓ to be the Newton's direction related to $H_\ell(L)$. This way there is a smaller system to solve, but we need to compute N_ℓ, a matrix whose columns span $\text{Null}(V_\ell^T)$.

A different Newton-Grassmann method is proposed in [77]. This algorithm avoids the computation of N_ℓ. Instead it forces the increment matrix, Z_ℓ, to satisfy $V_\ell^T Z_\ell = 0$. In this version the Newton direction is obtained by solving a symmetric linear system of order $p \times p$ where $p = k(m + n)$.

8.11 Gradient descent on the Grassmann manifold: Moving along a geodesic curve, GROUSE and SET

For large values of m and n Newton's iteration becomes expensive, which motivates the search for simpler iterations. Indeed, as we are about to show, the use of gradient descent methods on the Grassmann manifold gives rise to elegant ingenious schemes. The proposed methods are designed to compute a minimizer of the function

$$G(U) = F\big(U, \varphi(U)\big) \tag{8.26}$$

on the Grassmann manifold $\mathbb{G}_k^m$, where

$$\varphi(U) = \arg \min_{V \in \mathbb{R}^{n\times k}} F(U, V).$$

The matrix $V = \varphi(U)$ is obtained as in the second step of the ALS iteration. To clarify this statement we introduce the following notations. Let the vectors $\mathbf{c}_j \in \mathbb{R}^m$ and $\mathbf{v}_j \in \mathbb{R}^k$ denote the jth columns of A and V^T, respectively. The jth column of A is also used to define a diagonal matrix,

$$D_j = \text{diag}\{d_{11}, \ldots, d_{mm}\} \in \mathbb{R}^{m\times m},$$

whose entries satisfy the rule $d_{ii} = 1$ when a_{ij} is known, and $d_{ii} = 0$ when a_{ij} is missing. Then $\mathbf{v}_j$ is defined to be minimum norm solution of the linear least squares problem

$$\begin{aligned} &\text{minimize} && \|D_j(U\mathbf{v} - \mathbf{c}_j)\|_2^2 \\ &\text{subject to} && \mathbf{v} \in \mathbb{R}^k. \end{aligned}$$

In other words

$$\mathbf{v}_j = \mathbf{v}_j(U) = \arg \min_{\mathbf{v}\in\mathbb{R}^k} \|D_j(U\mathbf{v} - \mathbf{c}_j)\|_2^2.$$

Moreover, define

$$G_j(U) = \|D_j(U\mathbf{v}_j - \mathbf{c}_j)\|_2^2, \quad j = 1, \ldots, n,$$

then

$$G(U) = \sum_{j=1}^{n} G_j(U).$$

Given U and $V = \varphi(U)$ the related steepest descent matrix has the form

$$W = -\nabla G(U) = \big(P_\Omega(A) - P_\Omega(UV^T)\big)V.$$

In classic gradient descent step we proceed from U to a point $U + \theta W$, where $\theta > 0$ is a step-length parameter. Since U presents a point on $\mathbb{G}_k^m$ it is assumed to have full column rank. However, the new point $U + \theta W$ can be a rank-deficient matrix that fails to provide a point on $\mathbb{G}_k^m$. This obstacle invites the use of geodesic curves.

On Grassmann manifolds straight lines are replaced by geodesic curves. Roughly speaking, a geodesic curve between two points on a Grassmann manifold is the path of shortest length in the manifold that connects these points. The basic idea is to move from U into a point that lies on the "steepest" geodesic curve. That is, a geodesic curve that starts at U and proceeds with a constant tangent vector $W = -\nabla G(U)$. The formula that defines this curve uses the SVD of W, see equation (2.65) in [72]. Let

$$W = \tilde{U} S \tilde{V}^T$$

be an SVD of W, where $s_1 \geq s_2 \geq \cdots \geq s_k \geq 0$ denote the singular values of W and

$$S = \text{diag}\{s_1, \cdots, s_k\} \in \mathbb{R}^{k \times k}.$$

Then the corresponding geodesic curve is given by the formula

$$U(\theta) = [U\tilde{V}, \tilde{U}] \begin{bmatrix} \cos S\theta \\ \sin S\theta \end{bmatrix} \tilde{V}^T$$

where cos $S\theta$ and sin $S\theta$ are diagonal matrices:

$$\cos S\theta = \text{diag}\{\cos(s_1\theta), \ldots, \cos(s_k\theta)\} \in \mathbb{R}^{k \times k}$$

and

$$\sin S\theta = \text{diag}\{\sin(s_1\theta), \ldots, \sin(s_k\theta\} \in \mathbb{R}^{k \times k}.$$

This formula enables us to achieve a gradient descent step on the Grassmann manifold $\mathbb{G}_k^m$. However, since the matrices $U, V = \varphi(U)$ and W are not necessarily sparse, for large values of m, n, and k, the use of this formula can be "too expensive". The algorithms described below generate descending geodesic curves without computing a full SVD of W.

The GROUSE algorithm

The GROUSE algorithm of Balzano et al. [18] is based on the following observations. The gradient of $G_j(U)$ is the rank-one matrix $-2\mathbf{r}_j\mathbf{v}_j^T$, where $\mathbf{r}_j$ denotes the related residual vector,

$$\mathbf{r}_j = D_j(U\mathbf{v}_j - \mathbf{c}_j).$$

Note that $U^T\mathbf{r}_j = \mathbf{0}$, since $\mathbf{v}_j$ solves the related least squares problem. Moreover, let $\sigma = 2\|\mathbf{r}_j\|_2 \, \|\mathbf{v}_j\|_2$ denote the singular value of the rank-one gradient matrix and consider the unit vectors

$$\hat{\mathbf{p}}_j = U\mathbf{v}_j/\|U\mathbf{v}_j\|_2, \quad \hat{\mathbf{v}}_j = \mathbf{v}_j/\|\mathbf{v}_j\|_2, \quad \text{and} \quad \hat{\mathbf{r}}_j = \mathbf{r}_j/\|\mathbf{r}_j\|_2.$$

Then $\sigma\hat{\mathbf{r}}_j\hat{\mathbf{v}}_j^T = 2\mathbf{r}_j\mathbf{v}_j^T$, and the geodesic curve that corresponds to this matrix has the form

$$U(\theta) = U + \sin(\sigma\theta)\hat{\mathbf{r}}_j\hat{\mathbf{v}}_j^T + (\cos(\sigma\theta) - 1)\hat{\mathbf{p}}_j\hat{\mathbf{v}}_j^T.$$

The idea of GROUSE is to apply a series of small steps along these curves. More precisely, let $\theta_j > 0$ denote the step-length along the jth curve. Then the basic iteration is composed of n steps. Let U denote the current solution at the beginning of the jth step, $j = 1, \ldots, n$. Then U is updated by using the last formula with $\theta = \theta_j$.

The main effort in one iteration of GROUSE comes form the need to solve n different linear least squares problems. Note that similar linear least squares problems are solved in the second part of the ALS iteration.

The SET algorithm

A different way of building a descending geodesic curve, and moving along it, is applied in the SET algorithm, which is proposed by Dai et al. [51]. Let U denote the current point at the beginning of the basic iteration, and let $W = -\nabla F(U)$ denote the related steepest descent matrix. The SET iteration starts by computing a rank-one matrix, $\sigma \mathbf{x}\mathbf{y}^T$, that approximates W. That is, σ is the largest singular value of W, and the unit vectors $\mathbf{x}$ and $\mathbf{y}$ are the corresponding left and right singular vectors. Let the n vectors $\mathbf{u}_1, \ldots, \mathbf{u}_k$ denote the columns of U. It is shown in [51] that when W is replaced by its rank-one approximation the related geodesic curve can be written as

$$U(\theta) = [\mathbf{u}_1 \cos\theta + \mathbf{x} \sin\theta, \mathbf{u}_2, \ldots, \mathbf{u}_k], \quad \theta \in [0, \pi).$$

The second part of the iteration updates U to be $U(\theta)$ where θ is "proper" step-size. The heart of this part is a "line search" procedure that attempts to find a "satisfactory" local minimizer of the one parameter function $G\big(U(\theta)\big)$. The difficulties in conducting such a search come from the nonconvexity of $G\big(U(\theta)\big)$. This allows the existence of local minimizers which are separated by "barriers". The proposed line search uses the equality

$$G\big(U(\theta)\big) = \sum_{j=1}^{n} G_j\big(U(\theta)\big)$$

to detect the relevant barriers. See [50] and [51] for the details.

8.12 Riemannian optimization: Steepest descent, Conjugate Gradients and Trust-Region methods

Optimization on manifolds, or Riemannian optimization, is a fast growing branch of nonlinear optimization that is aimed at minimizing a smooth objective function on a Riemannian manifold. Roughly speaking, a Riemannian manifold $\mathbb{M}$ has the property that at each point $M \in \mathbb{M}$ there exists a tangent space $\mathbb{T}_M$, which is equipped with an inner product, and the inner product smoothly depends on M. For recent discussions of this methodology see [2, 3, 26, 27, 243].

To illustrate how Riemannian optimization works we consider the set of rank-k matrices

$$\mathbb{B}_k^* = \{B \,\big|\, B \in \mathbb{R}^{m\times n} \quad \text{and} \quad \text{rank}(B) = k\},$$

which is known to be a Riemannian manifold. Let $B \in \mathbb{B}_k^*$ be a given rank-k matrix. Then B has a "short" SVD of the form $B = U\Sigma V^T$, where $U \in \mathbb{R}^{m\times k},\ U^TU = I,\ V \in \mathbb{R}^{n\times k},\ V^TV = I,\ \Sigma = \text{diag}\{\sigma_1 \ldots, \sigma_k\} \in \mathbb{R}^{k\times k}$, and $\sigma_1 \geq \cdots \geq \sigma_k > 0$ are the singular values of B. Let $\mathbb{T}_B$ denote the tangent space of $\mathbb{B}_k^*$ at B. Then this space is composed of all the matrices $T \in \mathbb{R}^{m\times n}$ that can be expressed in the form

$$T = UCV^T + \tilde{U}V^T + U\tilde{V}^T,$$

where the matrices $C, \tilde{U}$ and $\tilde{V}$ satisfy $C \in \mathbb{R}^{k\times k},\ \tilde{U} \in \mathbb{R}^{m\times k},\ \tilde{V} \in \mathbb{R}^{n\times k},\ U^T\tilde{U} = 0$ and $V^T\tilde{V} = 0$. The inner product on $\mathbb{T}_B$ is defined as $\langle X, Y\rangle = \text{trace}(X^TY)$. The problem that we want to solve is the fixed-rank matrix completion problem

$$\begin{aligned} \text{minimize} \quad & F(B) = \|P_\Omega(A) - P_\Omega(B)\|_F^2 \\ \text{subject to} \quad & B \in \mathbb{B}_k^*. \end{aligned} \tag{8.27}$$

We shall start by considering **the Riemannian steepest descent method**. Starting at B, the basic iteration is composed of the following two steps.

Step 1: Projection. Project the steepest descent matrix $R = P_\Omega(A) - P_\Omega(B)$ on $\mathbb{T}_B$. The projected matrix $T \in \mathbb{T}_B$ has the form

$$T = UCV^T + \tilde{U}V^T + \tilde{U}V^T$$

where the matrices $C, \tilde{U}$ and $\tilde{V}$ are computed from the equalities $C = U^TRV,\ \tilde{U} = RV - UC$ and $\tilde{V} = R^TU - VC^T$.

Step 2: Retraction. Moving on $\mathbb{T}_B$ at the steepest descent direction brings us to the point $B + \theta T$, where $\theta > 0$ is a step-length parameter. The retraction operation projects this point back on $\mathbb{B}_k^*$. The projected point serves as starting point for the next iteration.

The projection on $\mathbb{B}_k^*$ is carried out by solving the problem

$$\begin{aligned} \text{minimize} \quad & \xi(X) = \|B + \theta T - X\|_F^2 \\ \text{subject to} \quad & X \in \mathbb{B}_k^*, \end{aligned}$$

whose solution is obtained by computing a rank-k TSVD of the matrix $B + \theta T$. This task becomes simpler by using the equality

$$B + \theta T = [U, \tilde{U}]H(\theta)[V, \tilde{V}]^T$$

where

$$H(\theta) = \begin{bmatrix} \Sigma + \theta C & \theta I \\ \theta I & 0 \end{bmatrix}$$

is a $2k \times 2k$ matrix. This way, after performing QR factorizations of $\tilde{U}$ and $\tilde{V}$, say $\tilde{U} = Q_uR_u$ and $\tilde{V} = Q_vR_v$, the SVD of $B + \theta T$ is obtained by computing the SVD of the $2k \times 2k$ matrix

$$\tilde{H}(\theta) = \begin{bmatrix} \Sigma + \theta C & \theta R_v^T \\ \theta R_u & 0 \end{bmatrix}.$$

The above iteration illustrates two basic techniques in Riemannian optimization. The projection of a direction matrix on $\mathbb{T}_B$ and the use of "retraction curves". Let $B^*(\theta)$ denote the projection of $B+\theta T$ on $\mathbb{B}_k^*$. In Riemannian optimization the "retraction curve" $B^*(\theta)$, $\theta>0$, serves as substitute for the corresponding geodesic curve. The motivation for this replacement comes from the following saving: Computing a point on the retraction curve requires a $2k\times 2k$ SVD, while computing a point on the related geodesic curve requires an $m\times n$ SVD.

Let us turn now to see how the **Conjugate Gradients (CG) method** is implemented to solve (8.27) within the Riemannian framework, as proposed in the **LRGeomCG** algorithm [272]. Here $B_\ell\in\mathbb{B}_k^*$ denotes the current solution at the beginning of the ℓth iteration, $\ell=1,2,\ldots,\mathbb{T}_{B_\ell}$ denotes the tangent space of $\mathbb{B}_k^*$ at B_ℓ, and the matrix $G_\ell\in\mathbb{T}_{B_\ell}$ denotes the projection of the gradient matrix, $P_\Omega(B_\ell)-P_\Omega(A)$, on $\mathbb{T}_{B_\ell}$. The matrix $T_\ell\in\mathbb{T}_{B_\ell}$ is used to denote the CG search direction on $\mathbb{T}_{B_\ell}$. The basic iteration of LRGeomCG is similar to the steepest descent iteration, but has an intermediate step that is called "vector transport", and projects direction matrices from $\mathbb{T}_{B_{\ell-1}}$ to $\mathbb{T}_{B_\ell}$. (The directions which are needed for constructing the CG search direction.) Let $\tilde{G}_{\ell-1}\in\mathbb{T}_{B_\ell}$ and $\tilde{T}_{\ell-1}\in\mathbb{T}_{B_\ell}$ denote the computed projections. Then the new search direction, T_ℓ, is obtained by applying the Polak-Ribiere CG scheme with $\tilde{G}_{\ell-1},\tilde{T}_{\ell-1}$, and G_ℓ. Finally, the retraction step is carried out by projecting the matrix $B_\ell-T_\ell$ back on $\mathbb{B}_k^*$.

Another option is to use a **Riemannian Trust-Region (RTR) method** . Recall that the TR search direction is obtained by minimizing a local quadratic model of the objective function, as in Newton's method, but here the minimizer is restricted to stay within a prescribed "trust-region" around the current solution point. The radius of the trust region is determined by a simple procedure that ensures "suitable" reduction of the objective function. (That is, a reduction that fits the anticipated model reduction.) The radius procedure avoids the need to achieve a line-search. The trust-region modification is helpful in minimization of nonconvex functions, as it prevents convergence toward a nonoptimal stationary point, and increases the possibility of converging toward a local minimizer. Usually the local quadratic model is based on the gradient vector and the Hessian matrix at the current point. Yet is some versions the Hessian is replaced by some other matrix. In particular, replacing the Hessian with the unit matrix, I, results in a trust-region version of the steepest descent method, in which a radius procedure replaces the line-search. The use of trust-region methods on Riemannian manifolds is studied in Absil et al. [2].

The Grassmann manifold $\mathbb{G}_k^m$ is another example of a Riemannian manifold. Recall that a point on $\mathbb{G}_k^m$ is a k dimensional subspace of $\mathbb{R}^m$, which is denoted as [U]. This notation means that $U\in\mathbb{R}^{m\times k}$, rank$(U)=k$, and $[U]=$ range(U). That is, $[U]$ is spanned by the columns of U. The tangent space of $\mathbb{G}_k^m$ at a point $[U]$ has the form

$$\mathbb{T}_{[v]}=\{T\,\big|\,T\in\mathbb{R}^{m\times k},\quad\text{and}\quad U^TT=O\}$$

Note that there is no loss of generality in assuming that U has orthonormal columns. In this case the projection of a matrix $H\in\mathbb{R}^{m\times k}$ on $\mathbb{T}_{[v]}$ is simply $(I-UU^T)H$. Following the methodology proposed in [2], Boumal and Absil [26, 27] develop two RTR methods for minimizing (8.26) on $\mathbb{G}_k^m$. The basic RTR iteration starts at a point $[U]\in\mathbb{G}_k^m$ and builds a local quadratic model on $\mathbb{T}_{[v]}$. The model is built by projecting the gradient and the Hessian on $\mathbb{T}_{[v]}$. See [27]

for the details. Then a direction matrix, $T \in \mathbb{T}_{[v]}$, is computed by minimizing the model within a suitable radius. Finally the matrix $B+T$ is retracted back to $\mathbb{G}_k^m$. The retraction is achieved by computing a QR factorization of this matrix, say $B+T=QR$. Then $[Q]$ serves as starting point for the next iteration. The first algorithm, RTRMC1, is an RTR version of the steepest descent method. The second algorithm, RTRMC2, is an RTR version of Newton's method. See [27] for detailed discussions of these methods.

8.13 Hybrid methods

The asymptotic rates of convergence of Newton's method and the Gauss-Newton method are expected to be faster than those of ALS or Gradient Descent methods. The price paid for this advantage is that the computational effort per iteration is considerably larger. Also, the faster rate of convergence is expected to occur only in close vicinity of the solution point, while far from that point Newton's method and the Gauss-Newton method are not expected to show better progress. On the other hand, as reported by a number of authors, ALS often enjoys a fast initial rate of convergence. These observations suggest the use of a hybrid algorithm which is capable of taking advantage of both types of methods. The hybrid algorithm is based on the following idea: Far from the solution point use "simple" iterations, such as ALS or Proximal-ALS, while close to that point use a fast converging iteration, such as Newton, Wiberg, or Newton-Grassmann. In practice, however, it is difficult to decide whether we are "far" or "close". One way to resolve this difficulty is by dividing the iterative process into two parts. The first part performs a preassigned number of "simple" iterations, while the second part performs "expensive" iterations. The experiments reported in [33], [43], and [77] illustrate the usefulness of this strategy.

A different approach is proposed in [56]. Here the basic iteration is composed of several "simple" iterations followed by one "expensive" iteration. The number of "simple" iterations is determined in a way that balances the computational efforts in the two parts of the basic iteration. This strategy avoids the somewhat arbitrary decision of how many "simple" iterations to apply before starting the "expensive" iterations. Also, combining two (or more) methods in one basic iteration reduces the possibility to "stuck" in a non-optimal point.

8.14 A gradual rank increasing process (GRIP)

The method proposed in this section combines the basic deflation procedure of Section 8.2 with an effective minimization algorithm for solving (1.1). This yields an improved building process that gradually builds a sequence of solutions

$$B_1^*, B_2^*, \ldots, B_k^*, \ldots, \tag{8.28}$$

where B_k^* denotes a computed solution of (1.1). The building process may use any available method for solving (1.1). The first matrix, B_1^*, is a rank-one matrix which is obtained by the

"criss-cross" algorithm. Let $B_k^* = (b_{ij}^{(k)})$ be presented in the form

$$B_k^* = U_k V_k^T = \sum_{p=1}^{k} \mathbf{u}_p^{(k)} \mathbf{v}_p^{(k)^T},$$

where

$$U_k = [\mathbf{u}_1^{(k)}, \ldots, \mathbf{u}_k^{(k)}] \in \mathbb{R}^{m \times k}, \quad \text{and} \quad V_k = [\mathbf{v}_1^{(k)}, \ldots, \mathbf{v}_k^{(k)}] \in \mathbb{R}^{n \times k}.$$

Then the kth building step, $k = 2, 3, \ldots$, starts with B_{k-1}^* and ends with B_k^*. The building of B_k^* is carried out via the following five stages.

Stage 1: Compute the related residual matrix $\tilde{A} = (\tilde{a}_{ij})$ in the following way. $\tilde{a}_{ij} = a_{ij} - b_{ij}^{(k-1)}$ when a_{ij} is known. Otherwise, when a_{ij} is unknown, $\tilde{a}_{ij}$ is considered as unknown. Of course, since B_{k-1}^* is kept as a sum of $k-1$ rank-one matrices, $b_{ij}^{(k-1)}$ is computed from that sum.

Stage 2: Compute a pair of vectors, $\tilde{\mathbf{u}} \in \mathbb{R}^m$ and $\tilde{\mathbf{v}} \in \mathbb{R}^n$, that solve the related rank-one problem (8.7). The computation of these vectors is carried out by the "criss-cross" iteration (8.4)-(8.5), using $\tilde{a}_{ij}$ instead of a_{ij}.

Stage 3: Define the matrices $X_0 \in \mathbb{R}^{m \times k}$, $Y_0 \in \mathbb{R}^{n \times k}$, and $W_0 = X_0 Y_0^T$, from the equalities

$$X_0 = [\mathbf{u}_1^{(k-1)}, \ldots, \mathbf{u}_{k-1}^{(k-1)}, \tilde{\mathbf{u}}] \quad \text{and} \quad Y_0 = [\mathbf{v}_1^{(k-1)}, \ldots, \mathbf{v}_{k-1}^{(k-1)}, \tilde{\mathbf{v}}].$$

Stage 4: Use an iterative minimization algorithm to solve (8.1), with $W_0 = X_0 Y_0^T$ as its starting point.

State 5: Define B_k^* to be the computed solution of (8.1)

One advantage of the above scheme is that the algorithm for solving (1.1) is provided a good starting point. This is likely to reduce the number of required iterations.

Another advantage lies in the information that is gathered from this process: In Section 20 we show that this information can be used to determine the final length (the optimal rank) of the computed solutions sequence (8.28). Also, as the next section shows, this process enables us to solve the rank minimization problem.

9 Rank minimization

This approach achieves imputing by solving the rank minimization problem

$$\begin{array}{ll} \text{minimize} & \text{rank}(B) \\ \text{subject to} & B \in \mathbb{A}. \end{array} \tag{9.1}$$

As before $\mathbb{A}$ denotes the set (6.1) of admissible matrices. Let B^* denote a computed solution of (9.1). Then the missing entries of A attain the values of the corresponding entries in B^*.

The motivation behind this approach comes from imputing problems in which the original data constitute a large low-rank matrix. For example, although the rank of the Netflix matrix is not known in advance, the (unknown) full matrix is expected to be a (nearly) low-rank matrix, because it is commonly believed that only a few factors contribute to an individual's tastes or preferences, e.g., [38, 39, 180, 188]. A second motivation comes from the possibility to achieve "exact recovery". See Theorem 6 below.

The rank minimization problem is known to be NP-hard in general, due to the combinatorial nature of the rank function. This observation is stated in many papers but it is difficult to find a proof of this fact. (It seems to be a consequence of Natarajan's observation [200] that the cardinality problem (10.2) is NP-hard.) The difficulty in solving the minimum rank problem is often resolved by using indirect methods that replace (9.1) with a similar problem which is simpler to solve. For example, replacing (9.1) with the nuclear norm problem (10.1). This approach is discussed in the next section. Other indirect methods for solving (9.1) are described below.

9.1 Indirect methods: Hard-Impute and IRLS

The Hard-Impute algorithm is proposed by Mazumder et al. [188]. It seeks a matrix $B \in \mathbb{R}^{m\times n}$ that solves the "hard-thresholding" problem

$$\text{minimize } \chi(B) = \|P_\Omega(B) - P_\Omega(A)\|_F^2 + \lambda\,\text{rank}(B) \tag{9.2}$$

where $\lambda > 0$ is a preassigned constant. The last problem is solved via the "Hard-Impute" iterative SVD that we described in Section 6.

Another indirect method for solving (9.1) is the IRLS method. The name stands for Iterative Reweighted Least Squares. The origin of the method lies in the field of linear least norm problems, when the residual norm is defined by the ℓ_1 norm and other types of Schatten-p norms. Recently the method has been adapted to solve the cardinaltiy minimization problem (10.2) and the basis pursuit problem (10.3). The adaptation of this approach to solve (9.1) is based on the following observations.

Let $X \in \mathbb{R}^{m\times n}$ be a given matrix, and let $\sigma_j(X),\ j = 1,\ldots,n$, denote the singular values of X. Let $p > 0$ be a given positive number and consider the matrix function

$$\rho(X) = \Big(\sum_{j=1}^{n} \big(\sigma_j(X)\big)^P\Big)^{1/p}.$$

If $p \geq 1$ then this function is a matrix norm, known as the Schatten-p norm. In particular, when $p = 1$ we obtain the nuclear norm. However, when $0 < p < 1$ it is neither a convex function nor a norm. Nevertheless, we can use it to construct smooth approximations to the rank function. In practice it is convenient to minimize the related power function

$$\tau(X) = (\rho(X))^P = \sum_{j=1}^{n} (\sigma_j(X))^P$$

instead of $\rho(X)$. This brings us to consider the problem

$$\begin{aligned} \text{minimize} \quad & \tau(X) \\ \text{subject to} \quad & X \in \mathbb{A}, \end{aligned}$$

with $0 < p \leq 1$, as a substitute for (9.1).

The IRLS method is aimed at solving the last problem. Let the matrix X_ℓ denote the current solution at the beginning of the ℓth iteration, $\ell = 1, 2, \ldots$, and assume for a moment that X_ℓ is a full rank matrix. In this case the matrix

$$W_\ell = (X_\ell^T X_\ell)^{(p-2)/2}$$

is well defined and

$$\tau(X_\ell) = \text{trace}(W_\ell^2 X_\ell^T X_\ell) = \|W_\ell X_\ell^T\|_F^2.$$

Moreover, since W_ℓ is a symmetric positive definite matrix, the equality

$$\text{trace}(W_\ell^2 X^T X) = \|W_\ell X^T\|_F^2$$

holds for any matrix $X \in \mathbb{R}^{m\times n}$. These relations suggest that the proximal function

$$\Pi_\ell(X) = \|W_\ell X^T\|_F^2$$

can be used to approximate $\tau(X)$ in a close vicinity of X_ℓ. However, the definition of W_ℓ should be modified to handle low-rank matrices. The modification which is given below is due to Mohan and Fazel [198]. The resulting IRLS iteration is composed of the following two steps.

Step 1: Define W_ℓ by the rule

$$W_\ell = (X_\ell^T X_\ell + \gamma_\ell I)^{(p-2)/2}$$

where $\gamma_\ell > 0$ is a small positive number.

Step 2: Compute $X_{\ell+1}$ to be a solution (or an approximate solution) of the proximal problem

$$\begin{aligned} \text{minimize} \quad & \Pi_\ell(X) = \|W_\ell X^T\|_F^2 \\ \text{subject to} \quad & X \in \mathbb{A}. \end{aligned}$$

The last problem resembles problem (7.8) in the FRAA algorithm. It is a linear least squares problem whose unknown variables are the unknown entries of X. Consequently , the solution matrix, $X_{\ell+1}$, can be obtained from W_ℓ, row after row, as in FRAA. However, as far as we know, this way of implementing the IRLS method has not yet been tested.

Mohan and Fazel [198] have used a Gradient Projection (GP) method for solving the proximal problem. The proposed inner iteration is based on the observations that the gradient of $\Pi_\ell(X)$ at a point $\hat{X} \in \mathbb{R}^{m\times n}$ points at the direction of $\hat{X}W_\ell$, while the projection of this vector

on the subspace $\{Y|Y \in \mathbb{R}^{m\times n}$ and $P_\Omega(Y) = 0\}$ has the form $\hat{X}W_\ell - P_\Omega(\hat{X}W_\ell)$. For example, if only one GP iteration is used, Step 2 is simplified as follows.

Step 2*: Define a positive step-length $\theta_\ell > 0$ and set

$$X_{\ell+1} = X_\ell - \theta_\ell\big(X_\ell W_\ell - P_\Omega(X_\ell W_\ell)\big).$$

A different version of IRLS is proposed in Fornasier et al. [85], who concentrate on nuclear norm minimization, when $p = 1$. Yet in all versions of IRLS the computation of W_ℓ requires the SVD of X_ℓ.

9.2 Direct methods: GRIP, SVP, and ADMIRA

A further way to resolve the minimum rank problem is opened by the following observation.

Theorem 4. *Let the matrix $B_k^*,\ k = 1,\ldots,n$, solve (1.1). Then there exists a rank index, k^*, such that*

$$F(B_k^*) > 0 \quad \text{for} \quad k = 1,\ldots,k^*-1, \quad \text{and} \quad F(B_k^*) = 0 \quad \text{for} \quad k = k^*,\ldots,n. \tag{9.3}$$

That is, k^* is the smallest rank index for which $F(B_k^*) = 0$. Furthermore, the matrix $B_{k^*}^*$ solves (9.1). Conversely, let B^* solve (9.1) and define k^* to be rank(B^*)*. Then k^* satisfies (9.3) and $F(B^*) = 0$ for $k = k^*,\ldots,n$. In other words, the matrix B^* solves (1.1) for $k = k^*,\ldots,n$.*

Proof. The definition of B_k^* implies the inequalities $F(B_k^*) \geq F(B_{k+1}^*), k = 1,2,\ldots,n-1$, and the equality $F(B_n^*) = 0$. These relations ensure the existence of an index k^* for which (9.3) holds. Now on one hand the equality $F(B_{k^*}^*) = 0$ implies that $B_{k^*}^*$ is an admissible matrix. On the other hand the inequalities $F(B_k^*) > 0,\ k = 1,\ldots,k^*-1$, imply that the admissible set does not contain matrices whose rank is smaller than k^*. This proves that $B_{k^*}^*$ solves (9.1). The converse claim follows from similar arguments. □

Remark: We have seen that (1.1) is not always solvable. In this case B_k^* denotes a computed solution of (1.1).

The last theorem has an important practical consequence: We can use the GRIP algorithm to build the sequence $B_1^*, B_2^*, \ldots, B_{k^*}^*$. If k^* is not expected to be small the search may take larger steps. For example, in Meka et al. [190] the authors suggest to solve (1.1) with the SVP iteration (6.10) - (6.11). Then, starting from some initial rank, the least squares problem (1.1) is repeatedly solved with increasing values of k. Each time k is increased by a fixed number (e.g., 10) until the objective function of (1.1) stops to decrease. A different search for k^* and $B_{k^*}^*$ is conducted in the ADMIRA algorithm of Lee and Bresler [165, 166]. As before the least squares problem (1.1) is solved a number of times, using trial values for k. But here the trial values are determined by a bisection search on k, until k^* is found.

The name ADMIRA stands for Atomic Decomposition for Minimum Rank Minimization, but the basic algorithm is designed to solve (1.1). It is an iterative algorithm that uses "atoms",

where "atom" refers to a rank-one matrix, $\Psi = \mathbf{u}\mathbf{v}^T \in \mathbb{R}^{m\times n}$, such that $\|\mathbf{u}\|_2 = \|\mathbf{v}\|_2 = 1$. The ℓth iteration, $\ell = 1, 2, \ldots$, is composed of the following three steps, starting with a rank-k matrix, B_ℓ, and a set of k atoms, $\Psi_1, \ldots, \Psi_k$ such that $B_\ell \in \text{Span}\,\{\Psi_1, \ldots, \Psi_k\}$.

Step 1: Compute a set of $2k$ atoms, say $\Psi_{k+1}, \ldots, \Psi_{3k}$, that solves the problem

$$\begin{aligned} \text{minimize} \quad & \|P_\Omega(A) - P_\Omega(B_\ell) - X\|_F^2 \\ \text{subject to} \quad & X \in \text{Span}\,\{\Psi_{k+1}, \ldots, \Psi_{3k}\}. \end{aligned}$$

The solution of this problem is achieved by a rank-$2k$ TSVD of the matrix $P_\Omega(A) - P_\Omega(B_\ell)$.

Step 2: Compute a matrix $Y_\ell \in \mathbb{R}^{m\times n}$ that solves the problem

$$\begin{aligned} \text{minimize} \quad & \|P_\Omega(A) - Y\|_F^2 \\ \text{subject to} \quad & Y \in \text{Span}\,\{\Psi_1, \ldots, \Psi_{3k}\}. \end{aligned}$$

The last problem is essentially a linear least squares problem in $3k$ unknowns and ν linear equations.

Step 3: Define $B_{\ell+1}$ to be a rank-k TSVD of Y_ℓ. This TSVD also provides the related k atoms of $B_{\ell+1}$.

Lee and Bresler [165, 166] suggest that for large problems the TSVD in Step 1 should be carried out by using Lanczos method [159, 160] or a random method [70]. For the linear least squares problem in Step 2 they recommend the Richardson iteration or Conjugate Gradients. Assume for a moment that $\Psi_i = \mathbf{u}_i\mathbf{v}_i^T$ for $i = 1, \ldots, 3k$, and let $U \in \mathbb{R}^{m\times 3k}$ and $V \in \mathbb{R}^{n\times 3k}$ be a pair of matrices whose ith rows equal $\mathbf{u}_i^T$ and $\mathbf{v}_i^T$, respectively. Then Y_ℓ has the form $Y_\ell = UDV^T$ for some diagonal matrix, D, whose diagonal entries solve the related least squares problem. Then, after computing QR factorizations of U and V, the task of computing an SVD of Y_ℓ is reduced to that of computing an SVD of a $(3k) \times (3k)$ matrix.

9.3 Uniqueness and exact recovery

A further consequence of the relation with least squares solutions regards the possibility to have a unique minimum rank solution. The importance of the uniqueness property is revealed in Theorem 6, in the context of "exact recovery".

Corollary 5 (Uniqueness of minimum-rank solutions)**.** *Let k^* be defined as in Theorem 4, and consider the least squares problem (1.1) with $k = k^*$. Then any solution of this problem solves (9.1), and vice versa. Consequently (9.1) has a unique solution if and only if the related least squares problem has a unique solution.*

Assume for a moment that A is obtained from a full matrix, $\hat{A}$, by considering some entries of $\hat{A}$ as unknown or missing. In such cases it is common to say that the known entries of A are "sampled" from those of $\hat{A}$. If the computed solution of (9.1) satisfies $B^* = \hat{A}$ then we say that we have "exact matrix completion", or "exact recovery". Now we characterize this possibility.

Theorem 6 (Exact recovery)**.** *Assume that the rank minimization problem (9.1) has a unique solution, B^*, and that*

$$\text{rank}(B^*) = \text{rank}(\hat{A}). \tag{9.4}$$

In this case we have exact recovery. That is,

$$B^* = \hat{A}. \tag{9.5}$$

Furthermore, both the uniqueness condition and the rank condition are necessary to ensure exact recovery.

Proof. Since $\hat{A}$ is admissible, the rank condition (9.4) implies that $\hat{A}$ solves (9.1), while the uniqueness forces B^* to equal $\hat{A}$. If the solution of (9.1) is not unique, we can find a solution that differs from $\hat{A}$. To show that the rank condition (9.4) is necessary we bring the following example. Let the data matrix $\hat{A} = (a_{ij})$ satisfy $a_{11} = a_{22} = 1$ and $a_{ij} = 0$ otherwise. Let A be obtained from $\hat{A}$ by considering a_{11} as unknown. In this case (9.1) has a unique solution $B^* = (b_{ij})$, where $b_{11} = 0, b_{22} = 1$, and $b_{ij} = 0$ otherwise. That is, $\hat{A}$ is a rank-2 matrix, B^* is a rank-one matrix, and $B^* \neq \hat{A}$. □

Theorem 6 shows the potential advantage in using the rank minimization approach. At the same time it exposes a weakness of this approach: If the required conditions fail to hold then we might get a "low quality" solution. (The question of how to assess the quality of a computed solution is discussed in Part III.) In particular, as explained in Section 20, if the solution of (9.1) is not unique, then it might be possible to get a better solution by solving the least squares problem (1.1) with a smaller matrix rank.

These considerations turn our attention to the question of which properties of A are likely to ensure a unique minimum rank solution. From Corollary 5 we know that this question is related to non-uniqueness of least squares solutions. Thus a partial answer is given in Theorem 2. Assume that we have a "defective" row or column in which the number of known entries is smaller than k^*. In this case the rank minimization problem (9.1) has infinitely many solutions.

Another factor that helps to answer the uniqueness question is φ, the number of degrees of freedom that we have in SVD presentation of B^*,

$$B^* = \sum_{j=1}^{k^*} \sigma_j \mathbf{u}_j \mathbf{v}_j^T,$$

where $k^* = \text{rank}(B^*)$. On one hand we are free to choose the $k^*(m+n+1)$ entries of the triplets $(\mathbf{u}_j, \mathbf{v}_j, \sigma_j), j = 1, \cdots, k^*$. On the other hand the singular vectors are forced to be mutually orthogonal and to have unit length. These relations introduce $k^*(k^*+1)$ constraints, so the overall number of degrees of freedom is

$$\varphi = k^*(m+n-k^*), \tag{9.6}$$

As before, ν denotes the overall number of known entries in A. If ν is smaller than φ there are less equations than degrees of freedom, and (9.1) is likely to have many solutions. Conversely, as ν exceeds φ there are more equations than degrees of freedom, and there are better chances of having a unique solution.

However, a large ratio ν/φ is not sufficient to avoid a "defective" row or column. A second helpful assumption is, therefore, that the known entries of A are sampled uniformly at random from $\hat{A}$. This type of sampling is likely to ensure that the known entries spread evenly among the rows and the columns of A. Then, as the ratio ν/m becomes considerably greater than k^* there is small probability to have a "defective" row or column.

The example in the proof of Theorem 6 demonstrates that exact recovery is not guaranteed when the singular vectors of $\hat{A}$ are constructed from rows and columns of identity matrices. Similar difficulties are expected when $\hat{A}$ has "spiky" singular vectors. That is, the "mass" of the vector is concentrated in a small number of entries. One way to avoid this possibility is by forcing the singular vectors of $\hat{A}$ to be spread out across all coordinates. This property of the singular vectors is called "low coherence" or "incoherence", e.g., [37, 38, 39, 85, 190, 224]. In the next section we shall see that the three properties mentioned above (ν/φ sufficiently large, randomly sampled entries, and low coherence) are likely to ensure exact recovery when solving nuclear norm problems.

10 Nuclear norm minimization

In this approach one achieves imputing by solving the minimum norm problem

$$\begin{aligned} &\text{minimize} \quad \|X\|_* \\ &\text{subject to} \quad X \in \mathbb{A}, \end{aligned} \tag{10.1}$$

where $\|X\|_*$ denotes the nuclear norm of the matrix X, which is the sum of its singular values. In Linear Algebra literature this norm is called Ky Fan norm or Schatten 1-norm, e.g., [58]. As with rank-minimization, the method is aimed at solving imputing problems in which the original data matrix, $\hat{A}$, is assumed to be a low-rank matrix.

The idea of replacing (9.1) with (10.1) comes from the field of Compressed Sensing, e.g., [37, 38, 69, 180, 261]. In this area we wish to compute the sparsest solution to highly underdetermined linear system of equations, $Y\mathbf{x} = \mathbf{b}$, where $\mathbf{x} = (x_1, \ldots, x_n)^T \in \mathbb{R}^n,\ Y \in \mathbb{R}^{\ell \times n}$, $\ell < n$, and $\mathbf{b} \in \mathbb{R}^\ell$. The fact that the system to solve is highly underdetermined means that ℓ is much smaller than n. To get a sparse solution we solve the "**cardinality minimization problem**"

$$\begin{aligned} &\text{minimize} \quad \text{card}(\mathbf{x}), \\ &\text{subject to} \quad Y\mathbf{x} = \mathbf{b}, \end{aligned} \tag{10.2}$$

where $\text{card}(\mathbf{x})$ denotes the number of nonzero components in the vector $\mathbf{x}$. Natarajan [200] has proved that the last problem is NP-hard. To get a simpler problem one replaces (10.2)

with the so-called "**basis pursuit problem**"

$$\begin{array}{ll} \text{minimize} & \|\mathbf{x}\|_1 \\ \text{subject to} & Y\mathbf{x} = \mathbf{b}, \end{array} \tag{10.3}$$

where $\|\mathbf{x}\|_1 = |x_1| + \cdots + |x_n|$ denotes the ℓ_1 vector norm. The last problem is a standard convex optimization problem for which there exists effective solution methods, such as Linear Programming or IRLS.

A second motivation for replacing (10.2) by (10.3) comes from the following observation. Let $\|\mathbf{x}\|_\infty = \max\limits_{j=1,\ldots,n} |x_j|$ denote the ℓ_∞ norm of $\mathbf{x}$, and let

$$\mathbb{B}_\infty = \{\mathbf{x} \in \mathbb{R}^n \,\big|\, \|\mathbf{x}\|_\infty \le 1\}$$

denote the corresponding unit norm ball. Then the function $\|\mathbf{x}\|_1$ is the convex envelope of card($\mathbf{x}$) on $\mathbb{B}_\infty$. That is, $\|\mathbf{x}\|_1$ is the largest convex function, $g(\mathbf{x})$, that satisfies $g(\mathbf{x}) \le$ card($\mathbf{x}$) for all $\mathbf{x} \in \mathbb{B}_\infty$, e.g., [81, 82, 180, 225].

However, in general the existence of sparse solutions for (10.2) does not force sparse solutions for (10.3). Consider for example the solution of a linear system which consists of one equation,

$$x_1 + x_2 + \cdots + x_n = 1.$$

In this example optimal solutions of (10.2) have cardinality 1, but (10.3) has many solutions with cardinality n. In contrast, when solving the equation

$$x_1 + (1/2)x_2 + (1/3)x_3 + \cdots + (1/n)x_n = 1$$

the ℓ_1 solution is a sparse solution of cardinality 1. These examples raise the question of which properties of the basis pursuit problem impose sparse solutions. Below we derive a simple condition of this type.

Assume that the basis pursuit problem (10.3) has a unique solution, $\mathbf{x}^* \in \mathbb{R}^n$. Then, as we now show,

$$\text{card}\,(\mathbf{x}^*) \le \ell + 1.$$

Therefore, when ℓ is much smaller than n, the solution is a sparse vector. The proof is based on the following argument. Define k to be the cardinality of $\mathbf{x}^*$. Then there is no loss of generality in assuming that the first k coordinates of $\mathbf{x}^*$ differ from zero, and the last $n-k$ coordinates of $\mathbf{x}^*$ equal zero. Let the matrix $Z \in \mathbb{R}^{\ell \times k}$ be obtained from the first k columns of Y, and let the vector $\mathbf{z} = (z_1, \ldots, z_k)^T \in \mathbb{R}^k$ be defined by the rule

$$z_j = \text{sign}(x_j^*) \;\text{ for } j = 1, \ldots, k.$$

If the linear system

$$Z\mathbf{u} = \mathbf{0} \quad \text{and} \quad \mathbf{z}^T\mathbf{u} = 0$$

has a nontrivial solution, $\mathbf{u}^* \neq \mathbf{0}$, then we can find a small positive constant, θ, for which $\mathbf{x}^* + \theta\mathbf{u}^*$ is another solution of (10.3). But this contradicts our uniqueness assumption. Consequently the columns of the related $(\ell + 1)$ by k matrix must be linearly independent, and k can't exceed $\ell + 1$.

The ability of ℓ_1 solutions to recover sparse solutions has been noticed in certain kinds of highly underdetermined linear systems that arise in geophysics and signal processing. This phenomenon has motivated several papers that attempt to provide mathematical explanations. Recent results suggest that the answer is related to matrix properties such as "incoherence" and "restricted isometry". See [38, 69, 130, 225] and the references therein.

The matrix completion problem can be formulated as underdetermined linear system of the form $Y\mathbf{x} = \mathbf{b}$, where $Y \in \mathbb{R}^{\nu \times mn}$ and $\mathbf{b} \in \mathbb{R}^{\nu}$. The rows of Y are sampled from the rows of the $mn \times mn$ identity matrix, and $\mathbf{b}$ consists of the known entries of A. If the number of known entries, ν, is much smaller than mn, then the system to solve is highly underdetermined. This suggests the use of similar strategies.

Let us return now to consider the nuclear norm problem (10.1). The reasons for replacing (9.1) by (10.1) are similar. First, a number of useful methods for solving the nuclear norm problem have been proposed recently. (See below.) Second, let $\|X\|_2 = \max\limits_{j=1,\ldots,n} \sigma_j$ denote the spectral norm of X, and let $\mathbb{B}_2 = \{X \in \mathbb{R}^{m\times n} \mid \|X\|_2 \leq 1\}$ denote the corresponding unit norm ball. Then the convex envelope of rank(X) on $\mathbb{B}_2$ is the nuclear norm function $\|X\|_*$, e.g., [81, 82, 180, 225]. In other words, the nuclear norm function is the largest convex underestimator of the rank function over the unit ball $\mathbb{B}_2$. Third, it is hoped that a small value of $\|X\|_*$ will force a "sparse" vector of singular values $(\sigma_1, \sigma_2, \ldots, \sigma_n)^T \in \mathbb{R}^n$, which means a low-rank matrix.

However, it should be noted that a solution of (10.1) is not necessarily close to solution of (9.1). Consider for example a 2×2 matrix, $A = (a_{ij})$, with $a_{12} = a_{21} = 4,\ a_{22} = 1$, and a_{11} unknown. In this example (9.1) is solved by a rank-one matrix, (10.1) is solved by a rank-two matrix, and the two solutions differ substantially.

A further motivation for solving (10.1) comes from the observation that exact recovery is expected under certain conditions. As before, the matrix $\hat{A} \in \mathbb{R}^{m\times n},\quad m \geq n$, denotes the original data matrix from which the known entries of A are sampled. It is assumed here that $\hat{A}$ is a low-rank matrix with SVD of the form

$$\hat{A} = \sum_{j=1}^{r} \sigma_j \mathbf{u}_j \mathbf{v}_j^T, \tag{10.4}$$

where $r = \text{rank}(\hat{A})$. A typical exact recovery assertion assumes that $\hat{A}$ and A satisfy the following three conditions.

Condition 1: The known entries of A have locations sampled uniformly at random.

Condition 2: The singular vectors of $\hat{A}$ satisfy certain "low-coherence" requirements.

Condition 3: The number of known entries, ν, is not "too small":

$$\nu \geq Cm^{5/4} r \log n, \tag{10.5}$$

where C is a positive constant.

This type of requirements was introduced by Candès and Recht [38] who proved that under these conditions the nuclear norm problem (10.1) has a unique solution which equals $\hat{A}$ with probability at least $1 - c/m^3$, where c is another positive constant.

It is tempting to conjecture that these conditions also imply a unique solution of (9.1), and that both problems share the same solution. This question and other related issues are discussed in [37, 38, 39, 190, 224, 227].

10.1 Equivalent formulations

In order to solve (10.1) we need to transform this problem into a "solvable" form that can be handled by efficient optimization techniques. Below we present three possible ways to derive alternative formulations. We shall start with two characterizations of the nuclear norm which are based on matrix factorizations. Then we turn to investigate the dual problem of (10.1). Another alternative is to formulate (10.1) as a semidefinite programming problem.

10.1.1 Matrix factorization

The first assertion gives the key argument behind the coming equivalence relations.

Lemma 7. *Let $D = \text{diag}\{d_1, \ldots, d_r\} \in \mathbb{R}^{r\times r}$ be a diagonal matrix with positive diagonal entries, $d_i > 0$, $i = 1, \ldots, r$. Let $d_i^{1/2} > 0$ denote the positive square root of d_i, $i = 1, \ldots, r$, and let $D^{1/2} = \text{diag}\{d_1^{1/2}, \ldots, d_r^{1/2}\} \in \mathbb{R}^{r\times r}$ denote the positive root of D. Let k be a given positive integer that satisfies $k \geq r$ and consider the least norm problem*

$$\begin{aligned} &\text{minimize} \quad \Phi(X,Y) = \frac{1}{2}(\|X\|_F^2 + \|Y\|_F^2) \\ &\text{subject to} \quad X \in \mathbb{R}^{r\times k}, \;\; Y \in \mathbb{R}^{r\times k} \quad \text{and} \quad XY^T = D. \end{aligned} \tag{10.6}$$

Let $\hat{Q} \in \mathbb{R}^{r\times k}$ be any matrix with orthonormal rows. That is $\hat{Q}\hat{Q}^T = I \in \mathbb{R}^{r\times r}$. Then the matrices $\hat{X} = D^{1/2}\hat{Q}$ and $\hat{Y} = D^{1/2}\hat{Q}$ solve (10.6), giving the optimal value of

$$1/2(\|\hat{X}\|_F^2 + \|\hat{Y}\|_F^2) = \sum_{i=1}^{r} d_i = \|D\|_*. \tag{10.7}$$

Conversely, let $\tilde{X} \in \mathbb{R}^{r\times k}$ and $\tilde{Y} \in \mathbb{R}^{r\times k}$ be a pair of matrices that solve (10.6). Then $\tilde{X} = \tilde{Y}$ and there exists an $r \times k$ matrix, $\tilde{Q}$, that has orthonormal rows and satisfies $\tilde{X} = \tilde{Y} = D^{1/2}\tilde{Q}$.

Proof. Let $X \in \mathbb{R}^{r\times k}$ and $Y \in \mathbb{R}^{r\times k}$ be a pair of matrices that satisfy $XY^T = D$. Let $\mathbf{x}_i$ and $\mathbf{y}_i$ denote the ith rows of X and Y, respectively, $i = 1, \ldots, r$. Then, clearly, $d_i = \mathbf{x}_i^T\mathbf{y}_i$ for $i = 1, \ldots, r$. Using the Cauchy-Schwartz inequality we see that

$$d_i = |\mathbf{x}_i^T\mathbf{y}_i| \leq \|\mathbf{x}_i\|_2\|\mathbf{y}_i\|_2.$$

Note also that the inequality $0 \leq (\|\mathbf{x}_i\| - \|\mathbf{y}_i\|)^2$ implies

$$\|\mathbf{x}_i\|_2\|\mathbf{y}_i\| \leq (\|\mathbf{x}_i\|_2^2 + \|\mathbf{y}_i\|_2^2)/2.$$

Combining these relations gives

$$d_i = |\mathbf{x}_i^T\mathbf{y}_i| \leq \|\mathbf{x}_i\|_2\|\mathbf{y}_i\|_2 \leq (\|\mathbf{x}_i\|_2^2 + \|\mathbf{y}_i\|_2^2)/2, \quad i = 1, \ldots, r$$

and

$$\sum_{i=1}^{r} d_i \leq \sum_{i=1}^{r}(\|\mathbf{x}_i\|_2^2 + \|\mathbf{y}_i\|_2^2)/2 = (\|X\|_F^2 + \|Y\|_F^2)/2.$$

That is, the optimal value of (10.6) is bounded from below by $\sum_{i=1}^{r} d_i$. This proves that the matrices $\hat{X} = D^{1/2}\hat{Q}$ and $\hat{Y} = D^{1/2}\hat{Q}$ solve (10.6).

Conversely, let the matrices X and Y solve (10.6). In this case the above inequalities are satisfied as equalities, forcing the equality $X = Y$. Now the equality $XX^T = D$ implies that the rows of X are mutually orthogonal, so $X = D^{1/2}\tilde{Q}$ for some $r \times k$ matrix, $\tilde{Q}$, with orthonormal rows. □

Theorem 8. *Let $B \in \mathbb{R}^{m\times n}$ be a given rank-r matrix with short SVD of the form*

$$B = U_r D V_r^T, \tag{10.8}$$

where

$$D = \text{diag}\{d_1, \ldots, d_r\} \in \mathbb{R}^{r\times r}$$

is a diagonal matrix,

$$d_1 \geq d_2 \geq \cdots \geq d_r > 0,$$

and the matrices $U_r \in \mathbb{R}^{m\times r}$ and $V_r \in \mathbb{R}^{n\times r}$ have orthonormal columns. That is,

$$U_r^T U_r = V_r^T V_r = I \in \mathbb{R}^{r\times r}.$$

Let k be a given integer that satisfies $k \geq r$ and consider the problem

$$\begin{array}{ll} \text{minimize} & \Psi(X,Y) = {}_{1/2}(\|X\|_F^2 + \|Y\|_F^2) \\ \text{subject to} & X \in \mathbb{R}^{m\times k}, \;\; Y \in \mathbb{R}^{n\times k}, \quad \text{and} \quad XY^T = B. \end{array} \tag{10.9}$$

Let $\hat{Q}$ be any $r \times k$ matrix with orthonormal rows. Then the matrices

$$\hat{X} = U_r D^{1/2}\hat{Q} \quad \text{and} \quad \hat{Y} = V_r D^{1/2}\hat{Q}$$

solve (10.9), giving the optimal value

$${}_{1/2}(\|\hat{X}\|_F^2 + \|\hat{Y}\|_F^2) = \sum_{i=1}^{r} d_i = \|B\|_*. \tag{10.10}$$

Conversely, let $\tilde{X} \in \mathbb{R}^{m\times k}$ and $\tilde{Y} \in \mathbb{R}^{n\times k}$ be a pair of matrices that solve (10.9). Then there exists an $r \times k$ matrix, $\tilde{Q}$, with orthonormal rows, such that

$$\tilde{X} = U_r D^{1/2}\tilde{Q} \quad \text{and} \quad \tilde{Y} = V_r D^{1/2}\tilde{Q}. \tag{10.11}$$

Proof. Let us extend the short SVD of B into a full SVD of the form

$$B = U_m \hat{D} V_n^T \tag{10.12}$$

where

$$\hat{D} = \text{diag}\{d_1, \ldots, d_r, 0, \ldots, 0\} \in \mathbb{R}^{m\times n},$$

$$U_m \in \mathbb{R}^{m\times m}, \;\; U_m^T U_m = I \in \mathbb{R}^{m\times m}, \;\; V_n \in \mathbb{R}^{n\times n}, \quad \text{and} \quad V_n^T V_n = I \in \mathbb{R}^{n\times n}.$$

The first r columns of U_m are those of U_r, and the first r columns of V_n are those of V_r. Now let us rewrite problem (10.9) with the matrices $W = U_m^T X$ and $Z = V_n^T Y$ instead of X and Y, respectively. This shows that (10.9) is equivalent to the problem

$$\begin{aligned} &\text{minimize} \quad \hat{\Phi}(W,Z) = \; {}_{1/2}(\|W\|_F^2 + \|Z\|_F^2) \\ &\text{subject to} \quad W \in \mathbb{R}^{m\times k}, \;\; Z \in \mathbb{R}^{n\times k}, \quad \text{and} \quad W Z^T = \hat{D}. \end{aligned} \tag{10.13}$$

That is, both problems share the same optimal value, and a solution to one problem provides a solution to the other problem. Furthermore, let $W \in \mathbb{R}^{m\times k}$ and $Z \in \mathbb{R}^{n\times k}$ be a pair of matrices that solve (10.13). Then the last $m-r$ rows of W have zero entries. Similarly, the last $n-r$ rows of Z have zero entries. This shows that (10.13) is equivalent to (10.6), so the proof follows directly from Lemma 7. □

Corollary 9. *Let the matrices $\tilde{X} \in \mathbb{R}^{m\times k}$ and $\tilde{Y} \in \mathbb{R}^{n\times k}$ solve* (10.9)*. Then this pair of matrices also solves the problem*

$$\begin{aligned} &\text{minimize} \quad \Pi(X,Y) = \|X\|_F \|Y\|_F \\ &\text{subject to} \quad X \in \mathbb{R}^{m\times k}, \;\; Y \in \mathbb{R}^{n\times k}, \quad \text{and} \quad X Y^T = B. \end{aligned} \tag{10.14}$$

giving the optimal value of

$$\|\tilde{X}\|_F \|\tilde{Y}\|_F = \|B\|_*. \tag{10.15}$$

Conversely, let the matrices $\hat{X}$ and $\hat{Y}$ solve (10.14). Then this pair of matrices solves (10.9)*.*

Proof. Let the matrices $X \in \mathbb{R}^{m\times k}$ and $Y \in \mathbb{R}^{n\times k}$ satisfy $XY^T = B$. Then there is no loss of generality in assuming $\|X\|_F = \|Y\|_F$, while the last equality implies

$$\|X\|_F \|Y\|_F = \|X\|_F^2 = \|Y\|_F^2 = \; {}_{1/2}(\|X\|_F^2 + \|Y\|_F^2).$$

□

Corollary 10. *Given a matrix $B \in \mathbb{R}^{m\times n}$ and a pair of matrices $X \in \mathbb{R}^{m\times k}$ and $Y \in \mathbb{R}^{n\times k}$ such that $B = XY^T$. Then*

$$\|B\|_* \leq \; {}_{1/2}(\|X\|_F^2 + \|Y\|_F^2),$$

and

$$\|B\|_* \leq \|X\|_F \|Y\|_F.$$

Furthermore, equality holds in these inequalities if and only if X and Y have the form (10.11)*.*

The observation that the optimal value of (10.9) equals $\|B\|_*$ is not new. Some authors use this property to define the nuclear norm of a matrix, e.g., [229, 249, 250, 251]. Yet, so far, we have not found in the literature a formal proof of this fundamental observation. The next theorem extends the results to matrices with missing entries.

Theorem 11. *Let the matrix $B^* \in \mathbb{A}$ solve the nuclear norm problem (10.1), and let k be a given integer such that $k \geq \text{rank}(B^*)$. Then (10.1) is equivalent to the problem*

$$\begin{array}{ll} \text{minimize} & \Psi(X,Y) = \ {}_{1/2}(\|X\|_F^2 + \|Y\|_F^2) \\ \text{subject to} & X \in \mathbb{R}^{m\times k}, \ \ Y \in \mathbb{R}^{n\times k}, \quad \text{and} \quad XY^T \in \mathbb{A}. \end{array} \tag{10.16}$$

That is, both problems share the same optimal value, and a solution to one problem provides a solution to the other problem.

Proof. Let the matrices $X^* \in \mathbb{R}^{m\times k}$ and $Y^* \in \mathbb{R}^{n\times k}$ solve the problem

$$\begin{array}{ll} \text{minimize} & \Psi(X,Y) = \ {}_{1/2}(\|X\|_F^2 + \|Y\|_F^2) \\ \text{subject to} & X \in \mathbb{R}^{m\times k},\ Y \in \mathbb{R}^{n\times k} \quad \text{and} \quad XY^T = B^*. \end{array}$$

Then

$$ {}_{1/2}(\|X^*\|_F^2 + \|Y^*\|_F^2) = \|B^*\|_*.$$

Similarly, let $\hat{X} \in \mathbb{R}^{m\times k}$ and $\hat{Y} \in \mathbb{R}^{n\times k}$ solve (10.16) and define $\hat{B} = \hat{X}\hat{Y}^T$. Then $\hat{B} \in \mathbb{A}$ and

$$ {}_{1/2}(\|\hat{X}\|_F^2 + \|\hat{Y}\|_F^2) = \|\hat{B}\|_*.$$

Now, since B^* solves (10.1), $\|B^*\|_* \leq \|\hat{B}\|_*$. On the other hand, since $\hat{X}$ and $\hat{Y}$ solve (10.16),

$$\|\hat{B}\|_* = \ {}_{1/2}(\|\hat{X}\|_F^2 + \|\hat{Y}\|_F^2 \leq \ {}_{1/2}(\|X^*\|_F^2 + \|Y^*\|_F^2) = \|B^*\|_*.$$

So the two inequalities forces the equalities

$$ {}_{1/2}(\|\hat{X}\|_F^2 + \|\hat{Y}\|_F^2) = \|\hat{B}\|_* = \|B^*\|_* = \ {}_{1/2}(\|X^*\|_F^2 + \|Y^*\|_F^2).$$

□

10.1.2 Duality relations

Let us turn now to investigate the dual form of (10.1). For this purpose we introduce some further notations. Let $G = (g_{ij})$ and $H = (h_{ij})$ be two matrices in $\mathbb{R}^{m\times n}$. Then the inner product between these matrices is defined as

$$\langle G, H\rangle = \sum_{i=1}^{m}\sum_{j=1}^{n} g_{ij}h_{ij}.$$

Note that

$$\langle G, H\rangle = \langle H, G\rangle = \text{trace}(G^T H) = \text{trace}(H^T G),$$

and $\|G\|_F = (\langle G, G\rangle)^{1/2}$ is the induced matrix norm. Let $E_{ij} \in \mathbb{R}^{m\times n}$ denote a matrix whose (i, j) entry equals 1 and all the other entries equal zero. Then the set $E_{ij}, i = 1, \ldots, m, \quad j = 1, \ldots, n$, form an orthonormal basis for $\mathbb{R}^{m\times n}$. Let

$$\mathbb{N} = \text{Span}\,\{E_{ij} | (i, j) \in \Omega\}$$

denote the subspace of $\mathbb{R}^{m\times n}$ which is spanned by matrices E_{ij} which correspond to known entries of A. Similarly, let

$$\mathbb{U} = \text{Span}\,\{E_{ij} | (i, j) \notin \Omega\}$$

denote the subspace of $\mathbb{R}^{m\times n}$ which is spanned by matrices E_{ij} which correspond to missing entries of A. Then, clearly, $\mathbb{U}$ is an orthogonal complement of $\mathbb{N}$ in $\mathbb{R}^{m\times n}$, and vice versa. Let the matrix operator P_Ω be defined as in problem (1.2). Then P_Ω is an orthogonal projection from $\mathbb{R}^{m\times n}$ to $\mathbb{N}$. Similarly, the matrix operator Q_Ω denotes orthogonal projection from $\mathbb{R}^{m\times n}$ to $\mathbb{U}$. Here the equality $G = Q_\Omega(H)$ implies $g_{ij} = 0$ when $(i, j) \in \Omega$, and $g_{ij} = h_{ij}$ when $(i, j) \notin \Omega$. Recall also that $G = P_\Omega(A)$ implies that $g_{ij} = a_{ij}$ when $(i, j) \in \Omega$, and $g_{ij} = 0$ when $(i, j) \notin \Omega$. Similarly, $Q_\Omega(A)$ is defined to be the null matrix.

With these notations at hand the set of admissible matrices can be expressed in the form

$$\mathbb{A} = \{P_\Omega(A) - H | H \in \mathbb{U}\},$$

which means that $\mathbb{A}$ is a linear variety in $\mathbb{R}^{m\times n}$. This presentation enables us to rewrite the nuclear norm problem (10.1) in the form

$$\begin{aligned} &\text{minimize} \quad \Pi(H) = \|P_\Omega(A) - H\|_* \\ &\text{subject to} \ \ H \in \mathbb{U}. \end{aligned} \tag{10.17}$$

That is, we seek a matrix H in a subspace $\mathbb{U}$ which is "closest" to $P_\Omega(A)$, regarding the nuclear norm.

The dual problem of (10.17) is specified by the Nirenberg-Luenberger Minimum Norm Duality (MND) theorem, e.g., [54, 179, 206]. Let $\|\cdot\|$ denote some (arbitrary) matrix norm on $\mathbb{R}^{m\times n}$ and let $\|\cdot\|'$ denote the corresponding dual norm. Recall that the dual norm is defined by the rule

$$\|G\|' = \sup\{\langle G, H\rangle \mid \|H\| \le 1\}.$$

Let $\mathbb{U}$ be a subspace of $\mathbb{R}^{m\times n}$ and let $\mathbb{U}^\perp$ denote the orthogonal complement of $\mathbb{U}$ in $\mathbb{R}^{m\times n}$. Given a matrix $Z \in \mathbb{R}^{m\times n}$ the MND theorem says that the dual of the primal problem

$$\begin{aligned} &\text{minimize} \quad \Pi(H) = \|Z - H\| \\ &\text{subject to} \quad H \in \mathbb{U} \end{aligned}$$

has the form

$$\begin{aligned} &\text{maximize} \quad \Delta(G) = \langle Z, G \rangle \\ &\text{subject to} \quad G \in \mathbb{U}^{\perp} \quad \text{and} \quad \|G\|' \leq 1, \end{aligned}$$

and that both problems share the same optimal value.

In our case, when searching the dual of (10.17), $\mathbb{U}^{\perp} = \mathbb{N}$, $Z = \mathbb{P}_{\Omega}(A)$, and the dual norm is specified by the rule

$$\|G\|'_* = \sup\{\langle G, H \rangle \;\Big|\; \|H\|_* \leq 1\}.$$

Let $\sigma_1(G) \geq \sigma_2(G) \geq \cdots \geq \sigma_n(G)$ and $\sigma_1(H) \geq \sigma_2(H) \geq \cdots \geq \sigma_n(H)$ denote the singular values of G and H, respectively. Then, as we are about to show, $\|G\|'_* = \sigma_1(G)$. From Von Neumann's trace inequality we know that

$$|\text{Trace } G^T H| \leq \sum_{j=1}^{n} \sigma_j(G)\sigma_j(H).$$

The last inequality implies

$$\langle G, H \rangle \leq \sigma_1(G) \sum_{j=1}^{n} \sigma_j(H) = \sigma_1(G)\|H\|_*$$

and

$$\sup\{\langle G, H \rangle \;\Big|\; \|H\|_* \leq 1\} \leq \sigma_1(G).$$

Now let $\mathbf{u} \in \mathbb{R}^m$ and $\mathbf{v} \in \mathbb{R}^n$ be a pair of dominant singular vectors of G such that $G\mathbf{v} = \sigma_1(G)\mathbf{u}, G^T\mathbf{u} = \sigma_1(G)\mathbf{v}$, and $\mathbf{u}^T\mathbf{u} = \mathbf{v}^T\mathbf{v} = 1$. Then the rank-one matrix $\mathbf{u}\mathbf{v}^T$ satisfies

$$\langle G, \mathbf{u}\mathbf{v}^T \rangle = \text{trace}(G^T\mathbf{u}\mathbf{v}^T) = \text{trace}(\sigma_1(G)\mathbf{v}\mathbf{v}^T) = \sigma_1(G).$$

In other words, the dual norm of the nuclear norm is the "spectral norm" $\|G\|_2 = \sigma_1(G)$.

Summarizing our results we conclude that the dual problem of the nuclear norm problem (10.17) has the form

$$\begin{aligned} &\text{maximize} \quad \Delta(G) = \langle P_{\Omega}(A), G \rangle \\ &\text{subject to} \quad G \in \mathbb{N} \quad \text{and} \quad \sigma_1(G) \leq 1. \end{aligned} \tag{10.18}$$

Note that the dual variables correspond to known entries of A. This can be an advantage in cases when the number of known entries is considerably smaller than the number of missing entries. However, to turn the dual approach into a useful tool we need to resolve two issues. First to have an effective algorithm for solving (10.18). Second, to find a rule for retrieving a primal solution from a dual one. Below we will show that both (10.1) and its dual can be formulated as SDP problems.

10.1.3 Semidefinite Programming (SDP)

The term Semidefinite Programming (SDP) refers to optimization problems in which some of the unknowns are matrices that are forced to be symmetric and positive semidefinite. The last constraint is summarized by the notation $S \geq 0$, which means that S is a symmetric positive semidefinite matrix. In this approach (10.1) is replaced with the following SDP problem.

$$\begin{array}{ll} \text{minimize} & \tau(B,R,S) = \ {}_{1/2}(\text{trace}(R) + \text{trace}(S)) \\ \text{subject to} & B \in \mathbb{A}, \ \ R = R^T \in \mathbb{R}^{m\times m}, \ \ S = S^T \in \mathbb{R}^{n\times n}, \end{array} \tag{10.19}$$

and

$$\begin{bmatrix} R & B \\ B^T & S \end{bmatrix} \geq 0.$$

Observe that the restriction $B \in \mathbb{A}$ can be replaced with the linear constraints

$$\langle E_{ij}, B\rangle = a_{ij} \ \ \forall (i,j) \in \Omega.$$

The next theorem shows that (10.1) and (10.19) are equivalent in the following sense. Both problems share the same optimal value, and a solution to one problem provides a solution to the other problem.

Theorem 12. *Let B^* solve (10.1). Define $r = \text{rank}(B^*)$ and let $B^* = U_r D V_r^T$ be a short SVD of B^*, as in* (10.8)*. Then the matrices B^*,*

$$R^* = U_r D U_r^T, \quad \text{and} \quad S^* = V_r D V_r^T \tag{10.20}$$

solve (10.19)*. Conversely, let the matrices $\hat{B}, \hat{R}$, and $\hat{S}$ solve* (10.19)*. Then $\hat{B}$ solves (10.1), and*

$$\text{trace}(\hat{R}) = \text{trace}(\hat{S}) = \|\hat{B}\|_* = \|B^*\|_*. \tag{10.21}$$

Proof. Let the matrices B, R, and S satisfy the constraints of (10.19), and let k denote the rank of the matrix $\begin{bmatrix} R & B \\ B^T & S \end{bmatrix}$. Then, since the last matrix is psd, there exists a pair of matrices, $X \in \mathbb{R}^{m\times k}$ and $Y \in \mathbb{R}^{n\times k}$, such that

$$\begin{pmatrix} X \\ Y \end{pmatrix} (X^T, Y^T) = \begin{bmatrix} R & B \\ B^T & S \end{bmatrix}. \tag{10.22}$$

The last equality means that

$$R = XX^T \quad \text{and} \quad S = YY^T, \tag{10.23}$$

so

$$\text{trace}(R) = \|X\|_F^2 \quad \text{and} \quad \text{trace}(S) = \|Y\|_F^2. \tag{10.24}$$

Furthermore, since

$$B = XY^T \in \mathbb{A}, \tag{10.25}$$

Theorem 11 implies that

$$\|B^*\|_* \leq \ {}_{1/2}(\|X\|_F^2 + \|Y\|_F^2) = \ {}_{1/2}(\text{trace}(R) + \text{trace}(S)). \tag{10.26}$$

This proves the first claim, as the matrices

$$X^* = U_r D^{\,1/2} \quad \text{and} \quad Y^* = V_r D^{\,1/2}$$

satisfy

$$\begin{pmatrix} X^* \\ Y^* \end{pmatrix} ({X^*}^T, {Y^*}^T) = \begin{bmatrix} R^* & B^* \\ {B^*}^T & S^* \end{bmatrix}$$

and

$$\|B^*\|_* = \ {}_{1/2}(\text{trace}(R^*) + \text{trace}(S^*)).$$

Conversely, let B, R, and S solve (10.19). Then, as we have seen, there exist matrices $X \in \mathbb{R}^{m\times k}$ and $Y \in \mathbb{R}^{n\times k}$ for which (10.22)--(10.26) hold. But now (10.26) is satisfied as equality. Therefore, since the optimal value of (10.16) equals $\|B^*\|_*$, X and Y solve (10.16). Hence the matrix $B = XY^T$ solves (10.1) and

$$\text{trace}(R) = \|X\|_F^2 = \|B^*\|_* = \|Y\|_F^2 = \text{trace}(S).$$

□

Let us consider for a moment the case when A has no missing entries. Here $B = A$ is a given matrix, and Theorem 12 yields the following results.

Corollary 13. *Let $B \in \mathbb{R}^{m\times n}$ be a given matrix. Let the symmetric matrices $\hat{R} \in \mathbb{R}^{m\times m}$ and $\hat{S} \in \mathbb{R}^{n\times n}$ solve the SDP problem*

$$\begin{aligned} &\text{minimize} \quad \rho(R,S) = \text{trace}(R) + \text{trace}(S) \\ &\text{subject to} \quad R = R^T \in \mathbb{R}^{m\times m}, \ \ S = S^T \in \mathbb{R}^{n\times n} \end{aligned} \tag{10.27}$$

$$\text{and} \quad \begin{bmatrix} R & B \\ B^T & S \end{bmatrix} \geq 0.$$

Then

$$\text{trace}(\hat{R}) = \text{trace}(\hat{S}) = \|B\|_*.$$

Corollary 14. *Given a matrix $B \in \mathbb{R}^{m\times n}$ and a real number β, the inequality*

$$\|B\|_* \leq \beta \tag{10.28}$$

holds if and only if there exist symmetric matrices $R \in \mathbb{R}^{m\times m}$ and $S \in \mathbb{R}^{n\times n}$, that satisfy

$$\begin{bmatrix} R & B \\ B^T & S \end{bmatrix} \geq 0 \quad \text{and} \quad \text{trace}(R) + \text{trace}(S) \leq 2\beta. \tag{10.29}$$

Remark: The derivation of the above results can be done in reverse order. That is, Corollary 13 is established from Corollary 14, Theorem 12 is derived from Corollary 13, Theorem 11 from Theorem 12, and so forth. However, a direct proof of Corollary 14 is quite complicated and requires the use of Fejer's theorem (a corollary of Schur Product theorem) as well as Von Neumman's Trace theorem. See [81, 82, 271] for details. In contrast, our method of proof starts with a simple intuitive proof of Lemma 7, from which the other results are easily concluded.

To formulate the dual problem (10.18) as SDP problem we use the following observations. Let $B \in \mathbb{R}^{m\times n}$ be a given matrix of rank r, and let

$$\sigma_1(B) \geq \sigma_2(B) \geq \cdots \geq \sigma_r(B) > 0$$

denote the nonzero singular values of B in decreasing order. Then the $(m+n)\times(m+n)$ matrix $\begin{bmatrix} 0 & B \\ B^T & 0 \end{bmatrix}$ has $2r$ eigenvalues of the form $\pm\sigma_j(B)$, $j = 1,\ldots,r$, and the rest $m+n-2r$ eigenvalues are zeros, e.g., Bjorck [23, p. 12]. Consequently $\sigma_1(B) \leq 1$ if and only if the matrix $\left[\begin{smallmatrix} I & B \\ B^T & I \end{smallmatrix}\right]$ is positive semidefinite. This enables us to write (10.18) as the SDP problem

$$\begin{aligned} &\text{maximize} \quad \langle P_\Omega(A), G\rangle \\ &\text{subject to} \quad G \in \mathbb{N} \quad \text{and} \quad \begin{bmatrix} I & G \\ G^T & I \end{bmatrix} \geq 0. \end{aligned}$$

As in (10.18), the restriction $G \in \mathbb{N}$ means that the dual variables correspond to known entries of A. Yet the question of how to retrieve a primal solution from a dual one remains open.

The experiments reported in [38] and [225] demonstrate the ability of the SDP formulation to solve nuclear norm problems. However, this formulation introduces $\frac{1}{2}(m(m+1)+n(n+1))$ new variables, the entries of R and S. So the overall number of variables in (10.19) is

$$mn + \tfrac{1}{2}(m^2+m+n^2+n) = \tfrac{1}{2}(m+n)(m+n+1).$$

This fact exposes a certain weakness of the SDP approach. Recall that one motivation for solving (10.1) comes from the Netflix problem, in which $m+n$ is about $498,000$. On the other hand, current SDP solvers are unable to solve (10.19) when $m+n$ exceeds a few hundreds, e.g., [35, 85, 176, 188, 261]. This limitation has motivated new types of methods, which are aimed at solving larger nuclear norm problems. This goal is achieved by taking advantage of two features that characterize the Netflix problem. First, the small percentage of known entries. Second, the assumption that these entries are sampled from a low-rank matrix.

10.2 The singular value shrinkage operator

The basic tool that enables the coming methods is called "the singular value shrinkage operator" or, briefly, "the shrinkage operator". Let $Y \in \mathbb{R}^{m\times n}$ be a given matrix with SVD of the form

$$Y = \sum_{j=1}^{n} \sigma_j \mathbf{u}_j \mathbf{v}_j^T, \tag{10.30}$$

and let $\lambda > 0$ be a given real positive number. Then the shrinkage operator, S_λ, is defined as follows:

$$S_\lambda(Y) = \sum_{j=1}^{n} (\sigma_j - \lambda)_+ \mathbf{u}_j \mathbf{v}_j^T. \tag{10.31}$$

Recall that $\theta_+ = \max\{0, \theta\}$ for any real number θ. So the last sum contains only terms with $\sigma_j > \lambda$, while terms with $\sigma_j \leq \lambda$ are vanished. The rank of $S_\lambda(Y)$ equals, therefore, the number of singular values which are greater than λ. The usefulness of the shrinkage operator comes from the observation that $S_\lambda(Y)$ solves the problem

$$\underset{X \in \mathbb{R}^{m \times n}}{\text{minimize}} \quad \varphi(X) = \tfrac{1}{2}\|X - Y\|_F^2 + \lambda \|X\|_*. \tag{10.32}$$

The proof of this feature is based on the following two lemmas.

Lemma 15. *Let $Y = \big(y_{ij}\big) \in \mathbb{R}^{m \times n}$ have the SVD (10.30). Then the diagonal entries of Y satisfy*

$$\sum_{j=1}^{n} |y_{jj}| \leq \sum_{j=1}^{n} \sigma_j.$$

Proof. Let u_{ij} and v_{ij} denote the (i, j) entries of the matrices

$$U = [\mathbf{u}_1, \ldots, \mathbf{u}_n] \in \mathbb{R}^{m \times n} \quad \text{and} \quad V = [\mathbf{v}_1, \ldots, \mathbf{v}_n] \in \mathbb{R}^{n \times n},$$

respectively. Then $U^T U = V^T V = I \in \mathbb{R}^n$, and (10.30) implies the equalities

$$y_{ii} = \sum_{j=1}^{n} \sigma_j u_{ij} v_{ij}, \quad i = 1, \ldots, n.$$

Summing these equalities gives

$$\sum_{i=1}^{n} |y_{ii}| \leq \sum_{i=1}^{n} \sum_{j=1}^{n} \sigma_j |u_{ij}||v_{ij}| = \sum_{j=1}^{n} \sigma_j \Big(\sum_{i=1}^{n} |u_{ij}||v_{ij}|\Big) \leq \sum_{j=1}^{n} \sigma_j,$$

where the last inequality follows from the Cauchy-Schwarz inequality and the fact that U and V have orthonormal columns. □

Lemma 16. *Let $D = \text{diag}\{\sigma_1, \ldots, \sigma_n\} \in \mathbb{R}^{m \times n}$ be a given diagonal matrix with nonnegative entries, and let $\lambda > 0$ be a given positive constant. Let the diagonal matrix $D_\lambda = \text{diag}\{d_1(\lambda), \ldots, d_n(\lambda)\} \in \mathbb{R}^{m \times n}$ be obtained from D by the rule*

$$d_j(\lambda) = (\sigma_j - \lambda)_+, \quad j = 1, \ldots, n.$$

That is, $D_\lambda = S_\lambda(D)$. Then D_λ is the unique solution of the problem

$$\begin{aligned} &\text{minimize} && \Psi(B) = \tfrac{1}{2}\|B - D\|_F^2 + \lambda \|B\|_* \\ &\text{subject to} && B \in \mathbb{R}^{m \times n}. \end{aligned} \tag{10.33}$$

Proof. Let $B = (b_{ij})$ be an arbitrary matrix from $\mathbb{R}^{m\times n}$. Let the diagonal matrix $\tilde{B} \in \mathbb{R}^{m\times n}$ share the same diagonal entries as B. That is, $\tilde{B} = \text{diag}\{b_{11}, \ldots, b_{nn}\}$. Then the former lemma implies

$$\|\tilde{B}\|_* \leq \|B\|_*.$$

Also, since D is a diagonal matrix,

$$\|\tilde{B} - D\|_F^2 \leq \|B - D\|_F^2.$$

Combining the two inequalities gives

$$\tfrac{1}{2}\|\tilde{B} - D\|_F^2 + \lambda\|\tilde{B}\|_* \leq \tfrac{1}{2}\|B - D\|_F^2 + \lambda\|B\|_*,$$

where equality holds if and only if $B = \tilde{B}$. This forces the solution of (10.33) to be a diagonal matrix. The jth diagonal entry of the solution matrix is equal, therefore, to $d_j(\lambda)$, the unique solution of the problem.

$$\text{minimize} \quad \psi_j(\theta) = \tfrac{1}{2}(\theta - \sigma_j)^2 + \lambda|\theta|.$$

□

Theorem 17. *The matrix $S_\lambda(Y)$ solves (10.32).*

Proof. Let $U \in \mathbb{R}^{m\times n}$ and $V \in \mathbb{R}^{n\times n}$ be obtained from the SVD of Y, as in Lemma 15. Then Y has a "long" SVD of the form $Y = \hat{U}DV^T$, where $D = \text{diag}\{\sigma_1, \ldots, \sigma_n\} \in \mathbb{R}^{m\times n}$, and $\hat{U} \in \mathbb{R}^{m\times m}$ is obtained by extending the columns of U to be an orthonormal basis of $\mathbb{R}^{m\times m}$. Then, since both $\|\cdot\|_F$ and $\|\cdot\|_*$ are unitarily invariant norms, the matrix $B = \hat{U}^T XV$ satisfies

$$\varphi(X) = \varphi(\hat{U}^T XV) = \Psi(B) = \tfrac{1}{2}\|B - D\|_F^2 + \lambda\|B\|_*.$$

So the proof follows from Lemma 16. □

The above proof of Theorem 17 differs from the proofs given in [35] and [180]. The former proofs establish that $S_\lambda(Y)$ is a minimizer of $\varphi(X)$ by showing that the null matrix $O \in \mathbb{R}^{m\times n}$ belongs to the subdifferential of $\varphi(X)$ at the point $S_\lambda(Y)$. The next property of S_λ is used in Lagrange Multipliers methods.

Corollary 18. *Let $Y \in \mathbb{R}^{m\times n}$ and $Z \in \mathbb{R}^{m\times n}$ be two given matrices. Then the matrix $S_\lambda(Y + Z)$ solves the problem*

$$\underset{X\in\mathbb{R}^{m\times n}}{\text{minimize}}\ \hat{\varphi}(X) = \lambda\|X\|_* - \langle Y, X\rangle + \tfrac{1}{2}\|X - Z\|_F^2.$$

Proof. The difference between $\hat{\varphi}(X)$ and the function

$$\tilde{\varphi}(X) = \lambda\|X\|_* + \tfrac{1}{2}\|X - (Y + Z)\|_F^2$$

is only a constant. □

10.3 The Singular Value Thresholding (SVT) algorithm

This iterative algorithm is proposed in Cai et al. [35]. It is aimed at solving the problem

$$\begin{array}{ll} \text{minimize} & 1/2\|X\|_F^2 + \lambda\|X\|_* \\ \text{subject to} & X \in \mathbb{A}, \end{array} \tag{10.34}$$

where $\lambda > 0$ is a preassigned "threshold" value. The motivation comes from the observation that for large values of λ the solution of (10.34) approaches a solution of (10.1). The ℓth iteration, $\ell = 1, 2, \ldots,$ starts with the matrices $X_{\ell-1}$ and $Y_{\ell-1}$, and ends with X_ℓ and Y_ℓ.

The SVT algorithm

First compute

$$X_\ell = S_\lambda(Y_{\ell-1}), \tag{10.35}$$

then

$$Y_\ell = Y_{\ell-1} + \delta_\ell P_\Omega(A - X_\ell), \tag{10.36}$$

where $0 < \delta_\ell < 2$ is a positive step-size. The starting matrix, Y_0, is always the null matrix of $\mathbb{R}^{m\times n}$.

As noted in [35], the proposed iteration can be interpreted as a Lagrange multipliers method, known as the Uzawa's algorithm. It is shown there that the sequence $\{X_\ell\}$ converges toward the unique solution of (10.34), provided that the sequence of step-sizes satisfies

$$0 < \inf_\ell \delta_\ell \le \sup_\ell \delta_\ell < 2. \tag{10.37}$$

An empirical observation made in [35] is that the matrices in the sequence $\{X_\ell\}$ have low-rank. This is, perhaps, due to the use of a large λ, to ensure that the computed solution of (10.34) approaches that of (10.1). Another useful property of the SVT algorithm is related to sparsity. Choosing Y_0 to be the null matrix ensures that all the coming matrices, $Y_\ell, \ell = 1, 2, \ldots,$ have the same sparsity pattern as $P_\Omega(A)$. Thus, the number of nonzero entries in Y_ℓ equals the number of known entries in A. The main computational effort is the computation of $S_\lambda(Y_{\ell-1})$ in (10.35). This suggests the use of methods which are able to take advantage of sparsity. The experiments in [35] use the "PROPACK" software package [159, 160], which carries out Lanczos bidiagonalization of a large sparse matrix.

The experience gained with the SVT algorithm reveals slow rate of convergence. See, for example, [35, 42, 176, 180, 186]. However, the SVT algorithm introduces a number of useful ideas: Replacing (10.1) with a "regularized problem", the use of the shrinkage operator S_λ, the use of Lagrange multipliers, and the use of Lanczos methods. The nuclear norm algorithms that follow SVT exploit these ideas to produce faster methods.

10.4 Regularized nuclear norm problems: Soft-Impute, FPC, APGL, and RTR

In this section we consider methods for solving the problem

$$\underset{X\in\mathbb{R}^{m\times n}}{\text{minimize}}\; R_\lambda(X) = \tfrac{1}{2}\|P_\Omega(X) - P_\Omega(A)\|_F^2 + \lambda\|X\|_*, \tag{10.38}$$

where $\lambda > 0$ is a preassigned regularization parameter. Following the discussion of regularized least squares problems it is easy to establish the existence of a matrix $X_\lambda^* \in \mathbb{R}^{m\times n}$ that solves (10.38). Define $\beta = \tfrac{1}{2}\|P_\Omega(X_\lambda^*) - P_\Omega(A)\|_F^2$ then X_λ^* also solves the problem

$$\begin{aligned} &\text{minimize} \quad \|X\|_* \\ &\text{subject to} \;\; X \in \mathbb{R}^{m\times n} \quad \text{and} \quad \|P_\Omega(X) - P_\Omega(A)\|_F^2 \le \beta. \end{aligned} \tag{10.39}$$

The need for solving problems of this form arises in cases when the known entries of A are corrupted with some noise. In such a case $\beta > 0$ is a preassigned parameter whose value reflects the level of noise. Note also that (10.38) can be interpreted as the Lagrange version of (10.39), e.g., [37, 180, 188].

Another closely related problem is (8.17). Assume for a moment that (8.17) is defined with $k \ge \text{rank}(X_\lambda^*)$. In this case (10.38) and (8.17) share the same optimal value, and a solution of one problem provides a solution to the other problem.

The motivation for replacing (10.1) with (10.38) comes from the observation that as λ tends to zero the solution of (10.38) approaches a solution of (10.1). Thus in practice (10.38) is repeatedly solved, using a decreasing sequence of regularization parameters

$$\lambda_1 > \lambda_2 > \cdots > \lambda_p > 0. \tag{10.40}$$

One way to solve (10.38) is by applying the following iterative algorithm. The ℓth iteration, $\ell = 1, 2, \ldots,$ starts with $X_{\ell-1}$ and ends with X_ℓ. In the first step of the basic iteration $X_{\ell-1}$ is used to generate an admissible matrix, $Z_\ell = (Z_{ij})$, using the rule

$$Z_\ell = P_\Omega(A) + Q_\Omega(X_{\ell-1}). \tag{10.41}$$

That is, $z_{ij} = a_{ij}$ when $(i,j) \in \Omega$, and $z_{ij} = x_{ij}$ otherwise. Then, in the second step, X_ℓ is obtained from Z_ℓ by the rule

$$X_\ell = S_\lambda(Z_\ell). \tag{10.42}$$

To justify this iteration we introduce the proximal function

$$\Pi_\ell(X) = \tfrac{1}{2}\|X - Z_\ell\|_F^2 + \lambda\|X\|_*, \tag{10.43}$$

which is strictly convex and satisfies

$$\Pi_\ell(X) \ge R_\lambda(X) \quad \forall X \in \mathbb{R}^{m\times n}$$

and

$$\Pi(X_{\ell-1}) = R_\lambda(X_{\ell-1}).$$

Therefore, since X_ℓ is the unique minimizer of $\Pi_\ell(X)$, the sequence $\{X_\ell\}$ has the decreasing property

$$R_\lambda(X_{\ell-1}) \geq R_\lambda(X_\ell). \tag{10.44}$$

The iteration (10.41) - (10.42) has been proposed in Mazumder et al. [188] under the name "**Soft-Impute**". (Since the matrix operator S_λ is sometimes called "soft-thresholding".) It is proved in [188] that the sequence $\{X_\ell\}$ converges toward a point X^*_λ that solves (10.38).

Observe the similarity between the Soft-Impute algorithm and the Iterative SVD algorithm: The basic iteration of Soft-Impute is obtained from that of Iterative SVD by replacing $T_k(Z_\ell)$ with $S_\lambda(Z_\ell)$. As in Iterative SVD, the updating rule (10.41) can be rewritten in the form

$$Z_\ell = X_{\ell-1} + (P_\Omega(A) - P_\Omega(X_{\ell-1})), \tag{10.45}$$

where the difference matrix, $P_\Omega(A) - P_\Omega(X_{\ell-1})$, points at the steepest descent direction of $F(X) = {}_{1/2}\|P_\Omega(A) - P_\Omega(X)\|^2_F$ at the point $X_{\ell-1}$. The use of (10.45) is advantageous in cases when $X_{\ell-1}$ is a low rank matrix and $P_\Omega(A) - P_\Omega(X_{\ell-1})$ is a sparse matrix. In such cases the Lanczos method is efficiently implemented to compute $S_\lambda(Z_\ell)$, see [188].

A similar method is proposed in Ma et al. [180] under the name **Fixed Point Continuation (FPC)** algorithm. (Both FPC and Soft-Impute were independently proposed at about the same time.) The FPC algorithm modifies (10.45) by allowing a move along the steepest descent direction. Here the basic iteration is composed of the following three steps

Step 1: Select a stepsize $\tau > 0$.
Step 2: Define Z_ℓ by the rule

$$Z_\ell = X_{\ell-1} + \tau(P_\Omega(A) - P_\Omega(X_{\ell-1})). \tag{10.46}$$

Step 3: Compute X_ℓ from the rule

$$X_\ell = S_{\lambda\tau}(Z_\ell). \tag{10.47}$$

It is proved in [180] that under certain conditions on the stepsize parameter, τ, the sequence $\{X_\ell\}$ converges toward a minimizer of (10.38). A modified version of FPC, which is called **FPCA**, computes $S_{\lambda\tau}$ by applying an approximate SVD that is based on the fast Monte Carlo algorithm developed by Drineas et al. [70]. The experiments reported in [180] use $\tau = 1$ for hard problems and $\tau = 2$ for easy problems. In the first case, when $\tau = 1$, the FPC algorithm coincides with the Soft-Impute algorithm.

Recently Toh and Yun [261] proposed a modified version of FPC whose basic iteration has the following form.

Step 0: (Acceleration.) Compute a stepsize $\beta_\ell > 0$ and set

$$\tilde{X}_{\ell-1} = X_{\ell-1} + \beta_\ell(X_{\ell-1} - X_{\ell-2}). \tag{10.48}$$

Step 1: (Line search.) Computes a stepsize $\tau > 0$ that attempts to minimize the one parameter objective function

$$\rho_\ell(\tau) = R_\lambda(S_{\lambda\tau}(\tilde{X}_{\ell-1} + \tau(P_\Omega(A) - P_\Omega(\tilde{X}_{\ell-1})))). \tag{10.49}$$

Step 2: Define Z_ℓ by the rule

$$Z_\ell = \tilde{X}_{\ell-1} + \tau(P_\Omega(A) - P_\Omega(\tilde{X}_{\ell-1})). \tag{10.50}$$

Step 3: Compute X_ℓ from the rule

$$X_\ell = S_{\lambda\tau}(Z_\ell). \tag{10.51}$$

In the first iteration, when $\ell = 1$, both $X_{\ell-2}$ and $X_{\ell-1}$ equal the starting point, X_0. The stepsize β_ℓ is defined in the following way:

$$\beta_\ell = (\alpha_{\ell-1} - 1)/\alpha_\ell, \tag{10.52}$$

where the sequence $\{\alpha_\ell\}$ is generated by the rule $\alpha_0 = 1$ and

$$\alpha_\ell = (1 + (1 + 4(\alpha_{\ell-1})^2)^{1/2})/2. \tag{10.53}$$

The line-search parameter, τ, is determined by inspecting the value of $\rho_\ell(\tau)$ on a sequence of trial values. See [261] for details.

The iteration (10.48) -- (10.51) is called an **Accelerated Proximal Gradient algorithm with Line-search**, or **APGL** in brief. The line-search requires a number of SVD computations. This increases considerably the computational effort per iteration. To reduce this effort the authors provide a shortened iteration, called APG, which avoids the line-search and uses $\tau = 1$. The experiments reported in [261] use the PROPACK [159] implementation of Lanczos method for SVD computations.

A **Riemannian trust-region algorithm (RTR)** for solving (10.38) is proposed in Mishra et al. [194]. Here the matrix of unknowns is factorized in the form $X = USV^T$ where $U \in \mathbb{R}^{m\times k}$ and $V \in \mathbb{R}^{n\times k}$ have orthonormal columns and $S \in \mathbb{R}^{k\times k}$ is a symmetric positive semidefinite matrix. This allows to replace $\|X\|_*$ with trace(S). The value of k is gradually increased in a manner that resembles the GRIP idea. That is, (10.38) is repeatedly solved for increasing values of k until a satisfactory solution is reached.

10.5 Lagrange Multiplier methods: ADM, PPA and ALS

To illustrate the basic ideas we consider the Augmented Lagrangian (AL) method for solving (10.1) The reader is referred to Bertsekas [21, 22] for detailed discussion of Lagrange Multiplier methods. Following this discussion the AL function of (10.1) can be written in the form

$$L(X, Y) = \lambda\|X\|_* - \langle Y, P_\Omega(X) - P_\Omega(A)\rangle + \ {}_{1/2}\|P_\Omega(X) - P_\Omega(A)\|_F^2,$$

where $Y \in \mathbb{N}$ denotes the related matrix of Lagrange multiples. (Recall that the entries of a matrix $Y = (y_{ij}) \in \mathbb{N}$ satisfy $y_{ij} = 0 \ \ \forall (i,j) \notin \Omega$.) The AL method is aimed at finding a saddle point of $L(X,Y)$. The ℓth iteration, $\ell = 1, 2, \ldots$, starts with $X_{\ell-1} \in \mathbb{R}^{m \times n}$ and $Y_{\ell-1} \in \mathbb{N}$, and ends with X_ℓ and Y_ℓ. The basic iteration is composed of two steps. The first one updates the X matrix, the second one updates the Y matrix.

The updating of X attempts to minimize the function

$$L_\ell(X) \ = \ L(X, Y_{\ell-1}) \ = \ \lambda \|X\|_* \ - \ \langle Y_{\ell-1}, P_\Omega(X) \ - \ P_\Omega(A) \rangle \ + \ {}_{1/2} \|P_\Omega(X) \ - \ P_\Omega(A)\|_F^2.$$

That is, we would like to define X_ℓ to be a minimizer of $L_\ell(X)$. However, computing an exact minimizer of $L_\ell(X)$ is not a simple task. Instead we compute an approximate minimizer, using a "proximal approximation" of $L_\ell(X)$. For this purpose we define the matrix

$$Z_\ell = P_\Omega(A) + Q_\Omega(X_{\ell-1}), \tag{10.54}$$

and consider the "proximal function"

$$\Pi_\ell(X) = \lambda \|X\|_* - \langle Y_{\ell-1}, X - Z_\ell \rangle + \ {}_{1/2} \|X - Z_\ell\|_F^2.$$

Then this function satisfies

$$\Pi_\ell(X_{\ell-1}) = L_\ell(X_{\ell-1}) \quad \text{and} \quad \Pi_\ell(X) \geq L_\ell(X) \ \ \forall X \in R^{m \times n}.$$

So a minimizer of $\Pi_\ell(X)$ is likely to provide a good substitute for a minimizer of $L_\ell(X)$. Now the minimizer of $\Pi_\ell(X)$ is given by Corollary 18. Hence in this way X_ℓ is defined by the rule

$$X_\ell = S_\lambda(Y_{\ell-1} + Z_\ell). \tag{10.55}$$

The updating of Y increases the penalty term

$$\eta_\ell(Y) = -\langle Y, P_\Omega(X_\ell) - P_\Omega(A) \rangle$$

by changing Y along the steepest ascent direction of $\eta_\ell(Y)$. That is,

$$Y_\ell = Y_{\ell-1} - \theta_\ell (P_\Omega(X_\ell) - P_\Omega(A)), \tag{10.56}$$

where $\theta_\ell > 0$ is a positive stepsize parameter. This way, starting from $Y_0 \in \mathbb{N}$, the last rule ensures that $Y_\ell \in \mathbb{N}$ for all ℓ.

An extended iteration of this type is proposed in Chen et al. [42] under the name **Alternating Direction Method (ADM)**. Let $\beta \geq 0$ be a given nonnegative bound, and let $\mathbb{W}_\beta$ denote a subset of $\mathbb{R}^{m \times n}$ which is defined by the rule

$$\mathbb{W}_\beta = \{ W \ \big| \ W \in \mathbb{R}^{m \times n} \quad \text{and} \quad \|P_\Omega(W) - P_\Omega(A)\|_F^2 \leq \beta \}.$$

Then the ADM is aimed at solving the problem

$$\begin{aligned} &\text{minimize} \quad \|X\|_* \\ &\text{subject to} \ X \in \mathbb{R}^{m\times n},\ W \in \mathbb{W}_\beta, \\ &\text{and} \qquad\quad X = W. \end{aligned} \tag{10.57}$$

The constraints of (10.57) can be summarized as $X \in \mathbb{W}_\beta$, which shows that (10.57) and (10.39) are essentially the same problem. Yet the recent formulation suggests the following AL method. Here the AL function has the form

$$L(W,X,Y) = \lambda\|X\|_* - \langle Y, X-W\rangle + \ {}_{1/2}\|X-W\|_F^2,$$

and the basic iteration is composed of three steps. The first one improves W, the second step improves X, and the last step improves Y. The change in W reduces the AL function while W is forced to stay in $\mathbb{W}_\beta$. The change in X uses S_λ to reduce the AL function. The change in Y increases the AL function by moving along the related steepest ascent direction. See [42] for the details. If $\beta = 0$ then (10.57) is reduced to the nuclear norm problem (10.1), and the ADM iteration coincides with (10.54) -- (10.56).

Another AL method is considered in Liu et al. [176] under the name primal-dual Proximal Point Algorithm (**primal-dual PPA**). In this iteration X is updated by minimizing a "primal proximal function" that resembles $\Pi_\ell(X)$. Then Y is updated by maximizing a "dual proximal function" that approximates the penalty term.

Let the matrix $X^* \in \mathbb{R}^{m\times n}$ solve the nuclear norm problem (10.1), and let k be a given integer that satisfies

$$\text{rank}(X^*) \le k \le n. \tag{10.58}$$

Then from Theorem 11 we know that (10.1) can be rewritten in the form

$$\begin{aligned} &\text{minimize} \quad \Psi(U,V) = \|U\|_F^2 + \|V\|_F^2 \\ &\text{subject to} \ U \in \mathbb{R}^{m\times k},\ \ V \in \mathbb{R}^{n\times k}, \quad \text{and} \quad UV^T \in \mathbb{A}. \end{aligned} \tag{10.59}$$

The use of an AL method for solving this problem is considered in Recht et al. [225]. In our notations the proposed AL function has the form

$$L(U,V,Y) \ = \ \lambda(\|U\|_F^2 + \|V\|_F^2) - \langle Y, P_\Omega(UV^T) - P_\Omega(A)\rangle + \ {}_{1/2}\|P_\Omega(UV^T) - P_\Omega(A)\|_F^2,$$

and the basic iteration is composed of three steps. The first step improves U by minimizing the AL function with respect to U. The second step minimizes the AL function with respect to V. The last step updates Y, using (10.56) with UV^T instead of X. Note that the first two steps can be implemented as in the ALS iteration for solving (8.17).

10.6 Retrospective Remarks

The observation of Candès and Recht [38] that nuclear norm solutions are able to achieve "exact recovery" is an important theoretical breakthrough, which motivated several authors to

propose new methods for solving (10.1). The interest in the nuclear norm approach is reflected in the following quote from [42]. "This breakthrough achievement reveals that completing a low-rank matrix can be accomplished by solving the convex relaxation problem (10.1), and it has immediately inspired many authors to work on efficient numerical algorithms for (10.1). To mention a few, ... ". Also, as recently noted in [102], "Algorithms for matrix completion seem to sprout like wildflowers in the spring ... ".

However, in spite of the current enthusiasm, there is a place for some reservations. One point regards the motivation behind the nuclear norm approach, which is borrowed from the field of Compressed Sensing. In this field it is desired to compute "sparse solutions" of underdetermined linear systems. But this approach is not necessarily helpful in matrix completion problems. Let $\hat{A}$ denote the original data matrix from which the entries of A are sampled. Then in many practical situations there is no reason to assume that $\hat{A}$ solves (10.1). Moreover, many papers that propose nuclear norm methods mention the solution of large Netflix-like problems as possible application. Yet, even in this specific application, we don't see that the nuclear norm approach has a particular advantage. As Table 3 shows, a number of nuclear norm methods were tested on matrix completion problems which are based on partial data sets of Netflix, Jester, and MovieLens. In these experiments the final RMSE error remains quite large, which means that the tested algorithms failed to solve (10.1). This is not surprising, since the methods in Sections 10.4 and 10.5 are solving regularized least squares problems.

The ability of nuclear norm solutions to achieve exact recovery were tested on "random problems" of the following type. The original matrix, $\hat{A}$, is defined to be a low-rank random matrix that satisfies the "low-coherence" requirement, and the locations of the known entries are sampled "uniformly at random". Let $\hat{r}$ denote the rank of $\hat{A}$. Then, as we have seen, the number of degrees of freedom in SVD presentation of $\hat{A}$ is $\hat{r}(m+n-\hat{r})$. The experience gained in these experiments indicates that the possibility to achieve exact recovery depends on the ratio between the overall number of known entries, ν, and the number of degrees of freedom. If $\nu < \hat{r}(m+n-\hat{r})$ then the computed nuclear norm solution fails to achieve exact recovery. To ensure exact recovery ν should be considerably larger than $\hat{r}(m+n-\hat{r})$. The experiments in [42], [176], [180] and [225] suggest that ν should be at least three times larger than $\hat{r}(m+n-\hat{r})$. In this case the conditions of Theorem 6 are likely to hold, and least squares methods with appropriate matrix rank, k, are also expected to achieve exact recovery. For detailed discussions of this possibility, when using ALS or Iterative SVD, see [136] and [190], respectively.

Another example that illustrates exact recovery is the MIT logo test image [225]. However, it is difficult to find examples of "real" matrix completion problems (problems that come from real applications) in which it is reasonable to expect exact recovery. In many practical situations $\hat{r}$ is not known. So neither the ratio between ν and $\hat{r}(m+n-\hat{r})$ nor a bound of the form (10.5) can be used to predict exact recovery. Similarly, there might be a difficulty to verify the low-coherence condition. For example, it is not known whether the Netflix matrix satisfies these requirements.

Part II: Related Problems and Methods

In this part we briefly consider some closely related topics which have been left outside the main course of our review. We start with two examples of special cases. That is, problems in which the matrix to be completed has a special structure. Then we move to consider two extensions of the matrix completion problem: The problem of imputing missing entries in tensors, and affinely constrained problems. Finally we discuss methods that come from different disciplines: Methods which are based on biological models, and statistical methods.

11 Robust Principal Components Analysis

As before, $\hat{A} \in \mathbb{R}^{m\times n}$ denotes a given data matrix. In this approach we seek a low rank matrix $X \in \mathbb{R}^{m\times n}$ and a sparse matrix $S \in \mathbb{R}^{m\times n}$ such that $\hat{A} = X + S$. Following the motivation behind the nuclear norm approach, the Robust PCA problem is often cast in the form

$$\begin{aligned} \text{minimize} \quad & R(X,S) = \|X\|_* + \eta\|S\|_1 \\ \text{subject to} \quad & X + S = \hat{A}, \end{aligned} \tag{11.1}$$

where η is a positive scalar, and $\|\cdot\|_1$ denotes the sum of absolute values of matrix entries. The modification of this approach to handle problems with missing entries is quite straightforward. In this case (11.1) is replaced with

$$\begin{aligned} \text{minimize} \quad & R(X,S) = \|X\|_* + \eta\|S\|_1 \\ \text{subject to} \quad & P_\Omega(X) + P_\Omega(S) = P_\Omega(A). \end{aligned} \tag{11.2}$$

The treatment of this problem resembles that of the nuclear norm problem. For example, in analogy to (10.38), it is possible to replace (11.2) with its regularized version

$$\text{minimize} \;\; R(X,S) = \lambda\|X\|_* + \lambda\eta\|S\|_1 + \tfrac{1}{2}\|P_\Omega(A) - P_\Omega(X+S)\|_F^2, \tag{11.3}$$

where $\lambda > 0$ is a preassigned regularization parameter. A typical iteration for solving (11.3) is composed of two steps. Starting with X_ℓ and S_ℓ the first step computes $X_{\ell+1}$ in attempt to minimize the proximal function $\Pi(X) = R(X, S_\ell)$. Then, in the second step, $S_{\ell+1}$ is defined as approximate minimizer of the function $\mu(S) = R(X_{\ell+1}, S)$. As before, the first problem is solved by applying the singular values shrinkage operator, $S_\lambda(Y)$. The second problem is solved by applying another matrix operator, $E_\varepsilon(Y)$, which is defined as the unique solution of the problem

$$\underset{Z\in\mathbb{R}^{m\times n}}{\text{minimize}} \;\; \xi(Z) = \tfrac{1}{2}\|Z - Y\|_F^2 + \varepsilon\|Z\|_1, \tag{11.4}$$

where $\varepsilon > 0$ is a given positive scalar. The last problem is solved by separate consideration of each entry. In this way we see that the entries of the solution matrix $Z = E_\varepsilon(Y)$ satisfy the following rule: $z_{ij} = y_{ij} - \varepsilon$ when $y_{ij} > \varepsilon$, $z_{ij} = y_{ij} + \varepsilon$ when $y_{ij} < -\varepsilon$, and $z_{ij} = 0$ when $|y_{ij}| \le \varepsilon$. Note the similarity and the difference between (11.4) and (10.32).

The Augmented Lagrangian (AL) function of (11.2) can be written in the form

$$\begin{aligned} L(X,S,Y) =& \lambda\|X\|_* + \lambda\eta\|S\|_1 + \langle Y, P_\Omega(A) - P_\Omega(X+S)\rangle \\ &+ \ {}_{1/2}\|P_\Omega(A) - P_\Omega(X+S)\|_F^2. \end{aligned} \tag{11.5}$$

Here the basic iteration is composed of three steps: The first step updates X in attempt to reduce the AL function, using the shrinkage operator S_λ. The second step updates S in attempt to reduce the AL function. This reduction uses the shrinkage operator $E_{\lambda\eta}$. The third step increases the AL function by moving Y along the steepest ascent direction. For detailed discussions of Robust PCA methods see [36, 169, 170, 245, 285].

12 Euclidean Distance matrix completion problems

In Linear Algebra literature the term "matrix completion problems" often refers to the task of imputing missing entries of matrices that have special properties, e.g. [138], [167]. This category includes positive semidefinite matrices [114, 139, 161], maximum rank matrices [96], skew-symmetric matrices [102], (Inverse) M-matrices [141], P-matrices [79, 140], Kernel matrices [112], and so forth. The solution of such problems requires special algorithms which take advantage of the specific properties of the matrix at hand.

The Euclidean Distance matrix completion problem is a good example of these types of problems. A matrix $A = (a_{ij}) \in \mathbb{R}^{n\times n}$ is called an r-dimensional Euclidean Distance (ED) matrix if there exist n points in $\mathbb{R}^r$, say $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$, such that

$$a_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|_2^2, \quad i = 1,\ldots,n, \quad j = 1,\ldots,n. \tag{12.1}$$

Let $X = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{r\times n}$ denote the related positions matrix, and let the column vector $\mathbf{g}(X)$ be composed from the diagonal entries of the matrix X^TX. That is,

$$\mathbf{g}(X) = (\|\mathbf{x}_1\|_2^2, \ldots, \|\mathbf{x}_n\|_2^2)^T \in \mathbb{R}^n. \tag{12.2}$$

Let the matrix operator $G(X)$ be defined as

$$G(X) = \mathbf{g}(X)\mathbf{e}^T + \mathbf{e}\mathbf{g}(X)^T - 2X^TX, \tag{12.3}$$

where $\mathbf{e} = (1,1,\ldots,1)^T \in \mathbb{R}^n$. With these notations at hand the definition (12.1) of an ED matrix can be rewritten as

$$A = G(X). \tag{12.4}$$

Observe that for any orthogonal matrix $Q \in \mathbb{R}^{r\times r}$, $Q^TQ = I$, the matrices X and QX share the same ED matrix, $G(X) = G(QX)$, and $\|X\|_F^2 = \|QX\|_F^2$. The Frobenius norm of the positions matrix can be reduced by considering the least squares problem

$$\underset{\mathbf{u}\in\mathbb{R}^r}{\text{minimize}}\ \rho(\mathbf{u}) = \sum_{j=1}^{n} \|\mathbf{x}_j - \mathbf{u}\|_2^2,$$

which has a unique minimizer at the centroid point

$$\hat{\mathbf{u}} = (\mathbf{x}_1 + \cdots + \mathbf{x}_n)/n = X\mathbf{e}/n.$$

Replacing $\mathbf{x}_j$ with $\mathbf{x}_j - \hat{\mathbf{u}}$ is carried out by using the centering matrix

$$C = I - \mathbf{e}\mathbf{e}^T/n,$$

which shifts the origin of $\mathbb{R}^r$ to the centroid point. The matrix XC is row-centered, and $\|XC\|_F^2 \leq \|X\|_F^2$. Furthermore, since shifting the origin does not change Euclidean distances, both X and XC share the same ED matrix, and $G(X) = G(XC)$. Hence there is no loss of generality in assuming that the positions matrix, X, is row-centered. That is

$$X\mathbf{e} = \mathbf{0} \quad \text{and} \quad XC = X. \tag{12.5}$$

A further consequence of the above relations is that the ED matrix is related with symmetric matrices of the form X^TX and $(XC)^T(XC)$. To clarify these relations we use the following notations. Let $\mathbb{S}_n$ denote the set of all $n \times n$ real symmetric matrices. Similarly, $\mathbb{S}_n^+$ denotes the set of all $n \times n$ real symmetric positive semidefinite matrices. Also, as before, $S \geq 0$ means $S \in \mathbb{S}_n^+$. Let $S = (s_{ij}) \in \mathbb{S}_n$ be a given symmetric matrix. Then the column vector

$$\mathbf{h}(S) = (s_{11}, \ldots, s_{nn})^T \in \mathbb{R}^n \tag{12.6}$$

is composed from the diagonal entries of S, and the matrix operator $H(S)$ is defined as

$$H(S) = \mathbf{h}(S)\mathbf{e}^T + \mathbf{e}\mathbf{h}(S)^T - 2S. \tag{12.7}$$

Assume further that

$$S \geq 0 \quad \text{and} \quad \text{rank}(S) \leq r. \tag{12.8}$$

In this case there exists a positions matrix, $X \in \mathbb{R}^{r\times n}$, such that

$$S = X^TX \quad \text{and} \quad G(X) = H(S). \tag{12.9}$$

In other words, if (12.8) holds then $H(S)$ is an r-dimensional ED matrix. Moreover, in this case the matrices S and CSC share the same ED matrix. That is,

$$H(CSC) = H(S). \tag{12.10}$$

Conversely, given an r-dimensional ED matrix, A, there exists a positions matrix, X, such that $A = G(X) = G(XC)$. Hence the matrix $S = (XC)^T(XC)$ satisfies

$$S \geq 0, \quad \text{rank}(S) \leq r, \quad A = H(S), \quad \text{and} \quad S = CSC. \tag{12.11}$$

Note that for any symmetric matrix S the condition $S = CSC$ is equivalent to $S\mathbf{e} = \mathbf{0}$. It is also easy to verify that the relations $A = H(S)$ and $S = CSC$ imply the equality

$$S = -\,{}_{1/2}\,CAC. \tag{12.12}$$

The matrix defined by the last equality is called the "source matrix" of A. The properties of this matrix are summarized below.

Theorem 19 (Uniqueness and minimum-norm property of the source matrix)**.** *Given an r-dimensional ED matrix, A, there exists a unique "source matrix", $S \in \mathbb{S}_n^+$, that satisfies (12.11) and (12.12). Moreover, let $\hat{S} \in \mathbb{S}_n^+$ be some other matrix that satisfies $A = H(\hat{S})$. Then*

$$\| - \tfrac{1}{2} CAC\|_F^2 \leq \|\hat{S}\|_F^2. \tag{12.13}$$

Corollary 20. *Computing a square-root of the source matrix (12.12) provides a row-centered positions matrix, $X \in \mathbb{R}^{r\times n}$, that satisfies*

$$X^T X = - \tfrac{1}{2} CAC, \quad A = G(X), \quad \text{and} \quad X\mathbf{e} = \mathbf{0}. \tag{12.14}$$

Furthermore,

$$\|X\|_F = \| - \tfrac{1}{2} CAC\|_F, \tag{12.15}$$

which means that X enjoys the minimum-norm property: Let $\hat{X}$ be some other positions matrix of A, then

$$\|X\|_F \leq \|\hat{X}\|_F. \tag{12.16}$$

Recall that the square-root of a source matrix is not unique, since $X^T X = (QX)^T(QX)$ for any orthonormal matrix $Q \in \mathbb{R}^{r\times r}$. However, let Y be some matrix that satisfies $Y^T Y = X^T X$. Then, clearly, $\|Y\|_F = \|X\|_F$. This leads to the following conclusion.

Corollary 21. *Any row-centered positions matrix enjoys the minimum-norm property.*

The ED matrix completion problem occurs when some entries of A are missing. In this case we would like to compute an r-dimensional ED matrix, B say, such that

$$P_\Omega(B) = P_\Omega(A), \tag{12.17}$$

where Ω and P_Ω are defined as in (1.2). If r is not specified, then it is often desired to find an ED matrix B, that satisfies (12.17) with the smallest possible r, e.g., [83]. Once B is computed, the missing entries in A attain the values of the corresponding entries in B.

The need for solving ED matrix completion problems arises in a number of important applications. One example comes from multidimensional scaling problems in psychometrics and statistics. Here the matrix entries represent similarities (or dissimilarities) between objects and we want to produce geometric representation of these objects, e.g., [25, 62, 63, 266]. A second example comes from computational chemistry. Here it is desired to determine the structure of a molecule ("molecular conformation") from information about interatomic distances, e.g., [48, 103, 267]. Recent applications include the use of acoustic echoes to reveal room shape, [67], and calibration of ultrasound tomography devices, [210]. Note that in many cases we seek the locations of points in $\mathbb{R}^2$ or $\mathbb{R}^3$. In some applications the problem to solve can be interpreted as graph realization problem. The connections between ED matrix completion problems, positive semidefinite matrices, and graphs are discussed in a number of papers, e.g., [114, 139, 161, 162]. For recent review of ED geometry and its applications see [168].

One way to solve the problem is by computing a (row-centered) positions matrix, X, such that

$$P_\Omega(G(X)) = P_\Omega(A). \tag{12.18}$$

In this approach X is obtained by solving the least-squares problem

$$\underset{X\in\mathbb{R}^{r\times n}}{\text{minimize}}\ \|P_\Omega(G(X)) - P_\Omega(A)\|_F^2, \tag{12.19}$$

which can be rewritten as

$$\begin{aligned} &\text{minimize} \quad \varphi(X) = \sum_{(i,j)\in\Omega} (a_{ij} - \|\mathbf{x}_i - \mathbf{x}_j\|_2^2)^2 \\ &\text{subject to} \quad X = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{r\times n}. \end{aligned} \tag{12.20}$$

Old and new methods for solving this problem are discussed in [80] and [192].

A second approach is to compute a source matrix, $S \in \mathbb{S}_n^+$, that satisfies

$$P_\Omega(H(S)) = P_\Omega(A). \tag{12.21}$$

This task can be achieved by solving an SDP problem of the form

$$\begin{aligned} &\text{minimize } \|P_\Omega(H(S)) - P_\Omega(A)\|_F^2 \\ &\text{subject to} \quad S \geq 0 \quad \text{and} \quad S\mathbf{e} = \mathbf{0}. \end{aligned} \tag{12.22}$$

A primal-dual interior-point algorithm for solving this problem is proposed in [6], while Riemannian optimization methods (gradient descent and trust-region) are discussed in [192].

A third way is proposed by Trosset [268]. It is based on the following observation whose proof is given in [60].

Lemma 22. *Let $\tilde{S} \in \mathbb{S}_n$ be a given symmetric matrix with eigenvalues $\tilde{\lambda}_1 \geq \tilde{\lambda}_2 \geq \cdots \geq \tilde{\lambda}_n$, and spectral decomposition $\tilde{S} = Q\tilde{D}Q^T$, $Q^TQ = I$, and $\tilde{D} = \text{diag}\{\tilde{\lambda}_1, \ldots, \tilde{\lambda}_n\}$. Let the diagonal matrix $\hat{D} = \text{diag}\{\hat{\lambda}_1, \ldots, \hat{\lambda}_n\}$ be obtained from $\tilde{D}$ by the rule: $\hat{\lambda}_i = \max\{\tilde{\lambda}_i, 0\}$ for $i = 1, \ldots, r$, and $\hat{\lambda}_i = 0$ for $i = r+1, \ldots, n$. Then the matrix $\hat{U} = Q\hat{D}Q^T$ solves the problem*

$$\begin{aligned} &\text{minimize} \quad \rho(U) = \|U - \tilde{S}\|_F^2 \\ &\text{subject to } U \in \mathbb{S}_{n,r}^+, \end{aligned} \tag{12.23}$$

where

$$\mathbb{S}_{n,r}^+ = \{U \,\big|\, U \in \mathbb{S}_n^+ \quad \text{and} \quad \text{rank}(U) \leq r\}.$$

As a consequence, the optimal value of this problem equals $\rho(\hat{U})$ and can be expressed by the function

$$\tau_r(\tilde{S}) = \sum_{i=1}^{r} (\tilde{\lambda}_i - \max\{\tilde{\lambda}_i, 0\})^2 + \sum_{i=r+1}^{n} \tilde{\lambda}_i^2. \tag{12.24}$$

Let $\tilde{\mathbb{A}}$ denote the set of matrices which are admissible for the ED matrix completion problem. More precisely, a matrix S belongs to $\tilde{\mathbb{A}}$ if and only if it satisfies the following three conditions: $S \in \mathbb{S}_n$, S has nonnegative entries, and $P_\Omega(S) = P_\Omega(A)$. Trosset [268] has proposed to consider the problem

$$\begin{array}{ll} \text{minimize} & \eta(U,S) = \|U - (-\,{}_{1/2}\,CSC)\|_F^2 \\ \text{subject to} & U \in \mathbb{S}_{n,r}^+ \quad \text{and} \quad S \in \tilde{\mathbb{A}}, \end{array} \tag{12.25}$$

and used Lemma 22 to rewrite it in the form

$$\begin{array}{ll} \text{minimize} & \tau_r(-\,{}_{1/2}\,CSC) \\ \text{subject to} & S \in \tilde{\mathbb{A}}. \end{array} \tag{12.26}$$

Since S must satisfy $P_\Omega(S) = P_\Omega(A)$, the unknown variables in the last problem are nondiagonal entries of S which correspond to missing entries of A. Trosset [267] and Grooms et al. [115] have used a trust-region method to solve this problem.

Another way to solve (12.25) is described below. The ℓth iteration, $\ell = 1, 2, \ldots,$ starts with U_ℓ and S_ℓ, and uses Lemma 22 to compute a matrix $U_{\ell+1} = (u_{ij})$ that solves the problem

$$\begin{array}{ll} \text{minimize} & \varphi(U) = \|U - (-\,{}_{1/2}\,CS_\ell C)\|_F^2 \\ \text{subject to} & U \in \mathbb{S}_{n,r}^+. \end{array} \tag{12.27}$$

Then $S_{\ell+1}$ is defined to be the (minimum norm) solution of the problem

$$\begin{array}{ll} \text{minimize} & \psi(S) = \|U_{\ell+1} - (-\,{}_{1/2}\,CSC)\|_F^2 \\ \text{subject to} & S \in \tilde{\mathbb{A}}. \end{array} \tag{12.28}$$

The last iteration is, clearly, a "block" alternating minimization method. The computation of $S_{\ell+1}$ is based on the observation that (12.28) is a linear least squares problem with simple bounds. The unknown variables here are the unknown entries of S, which are forced to stay nonnegative. The linear equations that define this system are obtained by equating the entries of $-\,{}_{1/2}\,CSC$ against the corresponding entries in $U_{\ell+1}$. Note that the (i,j) entry of $-\,{}_{1/2}\,CSC$ has the form $-\,{}_{1/2}\,\mathbf{c}_i^T S \mathbf{c}_j$, where $\mathbf{c}_j$ denotes the jth column of C, $j = 1, \ldots, n$. Hence the equality $-\,{}_{1/2}\,\mathbf{c}_i^T S \mathbf{c}_j = u_{ij}$ defines a linear equation in the unknown entries of S. Now the symmetry of $U_{\ell+1}$ and S implies that we have $(n+1)n/2$ linear equations in $\tilde{\nu}$ unknowns, where $\tilde{\nu}$ denotes the number of missing entries in the upper triangular part of A. The main point here is that the matrix which defines this system does not change during the iterative process. It is only the r.h.s. vector that changes.

13 Missing values in tensors

A tensor is a multidimensional array. A tensor of order N is an N-way array whose entries are addressed via N indices: A tensor of order one is a vector; a tensor of order two is a matrix;

a tensor of order three is a three-way array $T = (t_{ijk}) \in \mathbb{R}^{\ell \times m \times n}$. (Another name is N-mode array.) To simplify the coming discussion we concentrate on third order tensors. For a recent review of tensors and their applications see Kolda and Bader [150].

A common tool for analyzing a given data tensor, $T = (t_{ijk})$, is the CANDECOMP-PARAFAC (C-P) model. The C-P model was independently proposed by Carroll and Chang [41], and Harshman [122]. In the first paper it appears under the name "Canonical Decomposition" (CANDECOMP), while the second paper calls it "a Parallel Factors procedure", or PARAFAC in brief. In this approach the data tensor, T, is approximated by a sum of p rank-one tensors. The entries of a rank-one tensor $X = (x_{ijk}) \in \mathbb{R}^{\ell \times m \times n}$ satisfy $x_{ijk} = \alpha_i \beta_j \gamma_k$ for some vectors $(\alpha_1, \ldots, \alpha_\ell)^T \in \mathbb{R}^\ell$, $(\beta_1, \ldots, \beta_m)^T \in \mathbb{R}^m$, and $(\gamma_1, \ldots, \gamma_n)^T \in \mathbb{R}^n$. Similarly, a tensor $X = (x_{ijk}) \in \mathbb{R}^{\ell \times m \times n}$ is the sum of p rank-one tensors if there exist three matrices $A = (a_{ij}) \in \mathbb{R}^{\ell \times p}$, $B = (b_{ij}) \in \mathbb{R}^{m \times p}$, and $C = (c_{ij}) \in \mathbb{R}^{n \times p}$, such that

$$x_{ijk} = \sum_{q=1}^{p} a_{iq} b_{jq} c_{kq}. \tag{13.1}$$

Note that (13.1) implies the equality

$$X = X_1 + \cdots + X_p, \tag{13.2}$$

where $X_q, q = 1, \ldots, p$, is the rank-one tensor which is defined by the qth columns of A, B, and C, respectively. The entries of these matrices are determined in an attempt to solve the least squares problem

$$\begin{aligned} &\text{minimize} \quad f(A,B,C) = \sum_{i=1}^{\ell} \sum_{j=1}^{m} \sum_{k=1}^{n} \Big(t_{ijk} - \sum_{q=1}^{p} a_{iq} b_{jq} c_{kq} \Big)^2 \\ &\text{subject to} \quad A = (a_{ij}) \in \mathbb{R}^{\ell \times p}, B = (b_{ij}) \in \mathbb{R}^{m \times p}, \text{ and } C = (c_{ij}) \in \mathbb{R}^{n \times p}. \end{aligned} \tag{13.3}$$

As with matrices, the simplicity of the ALS method makes it a commonly-used tool for solving (13.3). Here the basic iteration of the ALS method runs through the matrices A, B, and C, modifying one matrix at a time. The convergence of the ALS is often improved by using regularization and line-search techniques, e.g., [41, 122, 150, 262, 263, 274]. Further methods are considered in [77] and [263].

The rank of a tensor X can be defined in a number of ways. In one way [150, page 464] it is defined as the smallest number of rank-one tensors whose sum equals X. This definition enables us to rewrite (13.3) in the following way

$$\begin{aligned} &\text{minimize} \quad f(X) = \sum_{i=1}^{\ell} \sum_{j=1}^{m} \sum_{k=1}^{n} (t_{ijk} - x_{ijk})^2 \\ &\text{subject to} \quad \text{rank}(X) \le p. \end{aligned} \tag{13.4}$$

The last problem can be viewed as an extension of the Eckart-Young problem (1.3) to (three-way) tensors. However, there are important differences between the two problems. One difference regards the existence of a tensor X^* that solves (13.4). A second difference is that

a rank-one tensor that solves (13.4) for $p = 1$ does not necessarily participate in a sum that solves (13.4) for a larger value of p. For detailed discussion of these differences see Kolda and Bader [150] and the references therein.

In practical applications it happens that the data tensor, T, has some missing entries. In this case the C-P model of T is often computed with the following modification: The sum that defines the objective function of (13.3) is restricted to known entries of T. (Note the similarity to (8.1).) Once the C-P tensor X is computed, the missing entries of T obtain the values of the corresponding entries in X. The current interest in matrix completion algorithms is followed by a growing interest in tensor completion algorithms. Naturally the treatment of tensors is more complicated but the basic approaches are similar. Below we briefly mention a few examples of tensor completion methods. The adaptation of the ALS to handle missing values is considered in [262]. Another method considered in [262] is a Gauss-Newton algorithm, called INDAFAC, which is aimed at minimizing the related nonlinear least squares problem. The basic iteration of INDAFAC computes the current residual vector, **r**, and the related Jacobian matrix, J. Then a search direction, **u**, is computed by solving the related Levenberg-Marquadt equation

$$(J^T J + \lambda I)\mathbf{u} = J^T \mathbf{r},$$

where λ is a positive parameter. The value of λ is determined in a way that ensures "satisfactory decreasing". (The last iteration can be viewed as trust-region method in which the radius is implicitly determined by λ.) The use of a Nonlinear Conjugate Gradient (NCG) method is proposed in [4] under the name "CP-WOPT". The authors provide two ways for computing the related gradient vector. One for "dense" problems with small percentage of missing entries, and one for "sparse" problems, with small percentage of known entries. In [154] a Riemannian nonlinear CG method is implemented on the manifold of low-rank tensors. (The manifold of tensors of fixed multilinear rank.) The geometric ingredients of this method include "retraction" and "vector transport" as in [272], but the actual computations are more complicated. The extension of the minimum rank approach and the nuclear norm approach to tensor completion problems is discussed in several papers, e.g., [95, 107, 174, 175, 223, 247, 265]. For a recent literature survey of low-rank tensor approximation techniques see [113].

14 Affinely constrained problems

Another extension regards the affine rank minimization problem

$$\begin{aligned} &\text{minimize} \quad \text{rank}(X) \\ &\text{subject to} \ \ \mathcal{A}(X) = \mathbf{b} \end{aligned} \tag{14.1}$$

where $\mathcal{A}$ is an affine transformation from $\mathbb{R}^{m\times n}$ to R^{ν}, $\mathbf{b} \in \mathbb{R}^{\nu}$, and the matrix $X \in \mathbb{R}^{m\times n}$ is the unknown variable to compute. In practice the last problem is often replaced by the related affine nuclear norm problem

$$\begin{aligned} &\text{minimize} \quad \|X\|_* \\ &\text{subject to} \ \ \mathcal{A}(X) = \mathbf{b}. \end{aligned} \tag{14.2}$$

An example of affine transformation is a linear map of the form

$$\mathcal{A}(X) = Y\mathbf{x} \tag{14.3}$$

where $Y \in \mathbb{R}^{\nu \times mn}$ is a given matrix and $\mathbf{x} = vec(X) \in \mathbb{R}^{mn}$ is obtained by unfolding X into an mn-vector. The matrix problems (9.1) and (10.1) are special cases of (14.1) and (14.2), respectively, in which the rows of Y are sampled from the rows of the $mn \times mn$ identity matrix. Indeed several authors consider the solution of (9.1) and (10.1) within this framework. The reader is referred to Recht et al. [225] for recent review of affine rank minimization problems. The analysis of these problems is often based on the assumption that the linear map $\mathcal{A}$ satisfies a certain Restricted Isometry Property (RIP). As noted in [37] and [85], the related matrix completion problem does not satisfy this property in general. Recently Meka et al. [190] have shown that the restricted isometry property holds for matrix completion problems under certain conditions. (Roughly speaking, these conditions require low rank, low coherence, randomly sampled entries, and ν large enough.)

15 Biologically inspired methods

In this section we introduce a new approach that gains popularity these days. The name biological methods refers to computational models that imitate the way a certain biological organ or system works. At the first moment it seems somewhat queer to use a biological model for computing the missing entries of an arbitrary data matrix. Indeed, these methods are quite different from the former imputing methods. However, as explained below, it is possible to harness these models to achieve imputing.

15.1 Artificial Neural Networks (ANN)

Artificial Neural Networks (ANN) are mathematical models, made from layers of "neurons", that imitate the basic structure of a nervous system. A typical ANN model consists of an input layer, one or more hidden layers, and an output layer. The information passes from the input layer to the output layer, through the hidden layers. Each layer contains a number of neurons that receive signals from neurons in the preceding layer, compute a weighted sum of these signals, pass the sum through a "transfer" function, and submit the resulting signal to neurons in the succeeding layer.

Let $x_1, \ldots, x_\ell$ denote the signals of the input neurons, and let $y_1, \ldots, y_n$ denote the signals of the output neurons. Assume for simplicity that the model has one hidden layer, and let $h_1, \ldots, h_m$ denote the signals of the hidden neurons in this layer. Then, for $i = 1, \ldots, m$,

$$h_i = f_i(\sum_{j=1}^{\ell} w_{ij}x_j),$$

where $f_i(\theta)$ denote the transfer function of the ith hidden neuron, and $w_{ij}, j = 1, \ldots, \ell$, are the related weight factors. Similarly, for $k = 1, \ldots, n$,

$$y_k = \tilde{f}_k(\sum_{i=1}^{m} \tilde{w}_{ki} h_i)$$

where $\tilde{f}_k(\theta)$ denotes the transfer function of the kth output neuron, and $\tilde{w}_{ki}$, $i = 1, \ldots, m$, denote the related weight factors.

A nerve cell is activated to send a signal if the sum of received signals exceeds a certain threshold. The transfer functions are designed to mimic this behavior. (These functions are also called activating functions, or logistic functions.)

The weight parameters and the transfer functions are determined by running an iterative process that is called "training". In this process the parameters of the ANN model are adjusted in attempt to follow an observed pair (or pairs) of input and output signals, $\mathbf{x}^* = (x_1^*, \ldots, x_\ell^*)^T$ and $\mathbf{y}^* = (y_1^*, \ldots, y_n^*)^T$ say. Given a set of parameters, the basic iteration starts by propagating forward the input signal, $\mathbf{x}^*$, to produce the related output signal, $\tilde{\mathbf{y}} = (\tilde{y}_1, \ldots, \tilde{y}_n)$ say. Then the related error signal $\mathbf{e} = \mathbf{y}^* - \tilde{\mathbf{y}}$ is propagated back through the network, so that at each layer the associated error is known. Finally, at each layer the associated error is used to correct the network parameters, where the correction is aimed to reduce the error. See, for example, [98, 119, 155, 197, 252] and the references therein.

To illustrate the way ANN models are used to recover missing data we consider the following example. Let the data matrix $A = (a_{ij}) \in \mathbb{R}^{m \times n}$ contain annual records of rain in a group of neighboring monitoring stations. That is, a_{ij} denote the annual rainfall at the jth station during the ith year. Assume further that the first ℓ columns of A are without missing entries, but the rest of the columns have some missing entries. Now let j^* be a column index such that $j^* > \ell$. Then we would like to recover the missing entries in the j^*th column by using the first ℓ columns. For this purpose we build an ANN model with ℓ input neurons, $x_1, \ldots, x_\ell$, and one output neuron, y_1. The model is trained to achieve the following task: Using the input signal $x_j = a_{ij}$, $j = 1, \ldots, \ell$, the output signal y_1 predicts the value of a_{ij^*}. The rows of the matrix in which a_{ij^*} is known are used for the training. If a_{ij^*} is missing then it attains the value of y_1.

It is interesting to compare this method with KNN imputing. In KNN, when the first ℓ columns serve as neighbors of the j^*th column, the missing value of a_{ij^*} is defined to be a weighted average of $a_{i1}, \ldots, a_{i\ell}$. So the KNN method can be viewed as a "degenerate" ANN model that misses its hidden layers. This comparison exposes the relative complexity of the ANN approach. To build an ANN model we need to decide on a proper number of hidden layers, and to fix the number of neurons at each layer. Then we have to achieve the training process, which can be a time consuming task. Nevertheless, despite these difficulties, there is a growing number of papers that report on successful use of ANN models for recovering missing data. See Table 5 for detailed references.

15.2 Genetic Algorithms

A genetic algorithm is a mathematical model that mimics biological evolution. It operates on binary strings which are called "individuals" or "chromosomes", where each chromosome is a vector $\mathbf{b} = (b_1, b_2, \ldots, b_\ell) \in \mathbb{R}^\ell$ whose entries are either 1 or 0. The algorithm is aimed to find a chromosome that minimizes (or maximizes) a given objective function. The objective function is transformed into a "fitness function" and each chromosome is assigned a "fitness value" which measures how good is that chromosome in solving the problem at hand.

The algorithm iterates on a set of N chromosomes which is called "population". The basic iteration starts with a given population and changes that population by applying "genetic operations". Typical genetic operations are "mutation" (random bit flip), "crossover" (exchange of corresponding substrings between two chromosomes), and "selection" (deletion of less successful individuals and making copies of successful ones). To follow the way biological evolution works, these operations are carried out by applying stochastic sampling methods which give preference to better fitted individuals. The basic iteration ends, therefore, with new population which is likely to be better fitted. A common practice is to start with a randomly chosen population, then the algorithm iterates for many generations until either a prespecified iteration limit is exceeded, or a satisfactory fitness level is reached by some individual.

The use of genetic algorithms to solve matrix completion problems can be done in an number of ways. Let us consider for example the nuclear norm problem (10.1) in the special case when the entries of A are nonnegative integers between 0 and $2^\eta - 1$, where η is a given positive integer. Then each entry of A can be represented by a binary string of length η. Assume further that A has μ missing entries which are ordered in some way, and let $x_1, \ldots, x_\mu$, denote the related values of these entries. This way each set of missing values uniquely defines a binary string of length $\ell = \eta\mu$, and vice versa. For each chromosome of that length the objective function value equals the nuclear norm of the related matrix. Now it is possible to use a genetic algorithm to find a well fitted chromosome. For further discussions of genetic algorithms and their use to approximate missing data see the references given in Table 5.

16 Statistical methods

We have seen that several imputing algorithms have evolved from statistical methods for analyzing incomplete sample surveys: Averaging methods are related with the "mean-imputing" method, the Nearest Neighbors approach evolved from "Hot-Deck Imputation", while ICR and Iterative SVD descend from the maximum likelihood EM framework. Another example is the use of Cross-Validation to assess the imputing error, see Section 19. It is interesting, therefore, to note the main features that characterize statistical treatment of missing data.

First, statistical methods often make assumptions on the data. For example, as in maximum likelihood EM methods and in multiple imputing methods, it is common to assume that the values of the data entries are generated by some (known) probabilistic distribution function. Another type of assumptions regards the missingness mechanism that determines the lo-

cations of missing entries. Here it is common to distinguish between "not missing at random", "missing at random", and "missing completely at random", e.g., [10, 173, 182, 236, 241, 270].

Second, and most important, the ultimate goal of statistical analysis is to produce reliable estimates of statistical parameters that characterize the data. (Such as means, variances, correlations, covariances, principal components, etc.) If this task can be satisfied without assigning values to missing entries, then the analysis is carried out without achieving imputing. Consequently, "listwise deletion" is the default option in most of the statistical software packages. In matrix terminology "listwise deletion" means that we simply delete any row that contains missing entries. The simplicity of this method makes it a popular option in many applications. Indeed, as reported in [270], an inspection of 103 research articles shows that listwise deletion is heavily utilized when treating missing data in operations management surveys. However, listwise deletion may lead to inaccurate estimates of statistical parameters, especially when the data fail to satisfy the "missing completely at random" assumption. Thus, from the statistical point of view, the reason for conducting imputation is to gain improved estimates of the desired parameters. For example, Allison [10] gives the following warning (regarding the maximum likelihood method). "The principal output from this algorithm is the set of maximum likelihood estimates of the means, variances and covariances. Although imputed values are generated as part of the estimation process, it is not recommended that these values be used in any other analysis."

In statistical terminology, methods like mean substitution, hot-deck imputation, and expectation-maximization, are classified as "single imputation" methods, since here each missing entry is assigned one value. These methods have certain advantages over listwise deletion, but liable to suffer from certain drawbacks. (For example, mean substitution is likely to produce smaller estimates of the variance.) The "multiple imputing" approach is designed to avoid biased estimates. In this method the values of the missing entries are computed by "random draws" from a given probabilistic distribution. Once all the missing entries attain values, the algorithm computes the desired set of statistical parameters. This imputing process is repeated ℓ times, producing ℓ sets of parameters. (Typically ℓ is a small integer between 3 to 10.) Then the final estimation of the parameters is carried out by taking means over the ℓ sets. This way one obtains an improved set of statistical parameters, as well as some indication on the accuracy of the derived estimates. Yet no concrete values are assigned to the missing entries. For detailed discussions of multiple imputation methods see [10], [230], [231], and [236].

The current review discusses the imputing problem from a Numerical Linear Algebra (NLA) point of view. The differences between the statistical approach and the NLA approach resemble the differences in the treatment of Linear Least Squares (LLS) problems. The statistical treatment of these problems is based on a "standard linear model" that satisfies the assumptions of the Gauss-Markov theorem. In contrast, most of the NLA textbooks are free from statistical terminology and the LLS problem is considered without mentioning the Gauss-Markov theorem. An inspection of our references shows that many of the older papers describe imputing methods within the statistical framework. For example, as noted in Section 6, the Iterative SVD algorithm is often described as an expectation-maximization procedure. This is, perhaps, because the statistical treatment of missing data is a relatively mature area

that preceded most of the other applications. Yet recent papers that are aimed at solving (1.1) or (10.1) follow the NLA approach.

Part III: Assessing the quality of the imputed entries

The last part of our review is dedicated to the question of how to assess the quality of the imputed entries. This issue has several interesting aspects, both from the theoretical and the practical points of views. The Netflix prize competition [203] is based on a certain assessment procedure but there are several other options. Below we describe the basic concepts and the difficulties that characterize each approach.

One way to evaluate the usefulness of a certain imputing method is to run the proposed algorithm on a number of test problems, where each test matrix, A, is generated from a known matrix, $\hat{A}$. In this case we can use direct error measurements. However, in practical situations the "true" values of the missing entries are not known. This brings us to consider implicit methods for estimating the prediction error. The basic idea is to test the ability of the algorithm to predict the values of known entries of A. The simplest version of this approach is based on Model Evaluation Metrics, see Section 18 below. Yet, as we shall see, this approach has certain limitations. A more sophisticated approach is to split the set of known entries into two distinct sets: A "training" set and a "probe" set. Then the algorithm runs on the training set and tested on the probe set. The probe set is used to define a "probe function" that measures the quality of the predicted entries which belong to the probe set. This idea comes from the statistical Cross-Validation method, see Section 19. The Cross-Validation technique is often used for tuning model's parameters. This idea is illustrated in Section 20: Comparing the behavior of the model evaluation metric with that of the probe function gives an effective way to determine an optimal matrix rank when solving (1.1).

17 Direct error measurements

This approach is possible when the test matrix, A, is obtained from a known matrix, $\hat{A}$, by considering some entries of $\hat{A}$ as "missing". Usually the locations of the "known" entries (or the locations of the "missing" entries) are "sampled at random". Also, as we have seen, many methods achieve imputing by computing a low-rank matrix $X = (x_{ij}) \in \mathbb{R}^{m\times n}$ that approximates A in some way. Then the missing entries of A attain the values of the corresponding entries in X. In such cases, when both $\hat{A}$ and X are available, it is possible to measure the error with the ratio

$$\|X - \hat{A}\|_F / \|\hat{A}\|_F. \tag{17.1}$$

The use of this measure is common in the Nuclear Norm literature. See, for example, [35, 37, 38, 42, 106, 176, 180, 225, 261]. Yet another option is to concentrate on the prediction error.

In this way (17.1) is replaced with the ratio

$$\Big(\sum_{(i,j)\in\Omega'} (x_{ij} - a_{ij})^2 \Big)^{1/2} \Big/ \Big(\sum_{(i,j)\in\Omega'} a_{ij}^2 \Big)^{1/2}, \tag{17.2}$$

where

$$\Omega' = \{(i,j) \mid a_{ij} \text{ is missing } \} \tag{17.3}$$

denotes the index set of missing entries. In [188] the authors use a slightly different error normalization of the form

$$\Big(\sum_{(i,j)\in\Omega'} (x_{ij} - a_{ij})^2 \Big)^{1/2} \Big/ \Big(\sum_{(i,j)\in\Omega'} x_{ij}^2 \Big)^{1/2}. \tag{17.4}$$

In problems that come from Collaborative Filtering and Recommender Systems the entries of the matrix lie in a certain range. For example, the known entries of the Netflix matrix are integers between 1 and 5. In this area it is common to compute the average value of the prediction error. The Mean Absolute Error (MAE) criterion is defined as

$$\text{MAE} = \Big(\sum_{(i,j)\in\Omega'} |x_{ij} - a_{ij}| \Big) / \nu', \tag{17.5}$$

where ν' denotes the number of missing entries. The Normalized MAE criterion is

$$\text{NMAE} = \text{MAE} / (\alpha_{max} - \alpha_{min}), \tag{17.6}$$

where

$$\alpha_{max} = max\{a_{ij} | (i,j) \in \Omega'\} \quad \text{and} \quad \alpha_{min} = min\{a_{ij} | (i,j) \in \Omega'\}.$$

Similarly the Root Mean Squared Error (RMSE) criterion is defined as

$$\text{RMSE} = \Big((1/\nu') \sum_{(i,j)\in\Omega'} (x_{ij} - a_{ij})^2 \Big)^{1/2}. \tag{17.7}$$

The error criteria MAE, NMAE, and RMSE are often used to evaluate the performance of recommender systems, e.g., [7, 40, 104, 118, 128, 189, 255]. The use of these criteria is easily extended to methods that achieve imputing without a low-rank matrix X that approximates A. (Such as KNN or ICR.) In this case x_{ij} denotes the predicted value of the (i,j) entry.

18 Model evaluation metrics

In practical problems the "true" values of the missing entries remain unknown. Hence the above definitions of MAE, NMAE, and RMSE need to be modified. One way to resolve this difficulty is by summing the error over the index set of known entries,

$$\Omega = \{(i,j) \mid a_{ij} \text{ is known } \}.$$

As before it is assumed that the tested algorithm achieve imputing by generating a low-rank matrix X that approximates A. The idea is that the ability of X to approximate the known part of A reflects its ability to predict the unknown part. This leads to the following definitions, e.g., [40, 42, 104, 128].

$$\text{MAE} = \Big(\sum_{(i,j)\in\Omega} |x_{ij} - a_{ij}| \Big) / \nu, \tag{18.1}$$

where ν denotes the number of known entries.

$$\text{NMAE} = \text{MAE} \Big/ (\alpha_{max} - \alpha_{min}), \tag{18.2}$$

where

$$\alpha_{max} = max\{a_{ij} \;\big|\; (i,j) \in \Omega\} \quad \text{and} \quad \alpha_{min} = min\{a_{ij} \;\big|\; (i,j) \in \Omega\},$$

and

$$\text{RMSE} = \Big((1/\nu) \sum_{(i,j)\in\Omega} (x_{ij} - a_{ij})^2 \Big)^{1/2} \tag{18.3}$$

Observe that the last three criteria are essentially equivalent. Being related to vector norms on $\mathbb{R}^\nu$ these terms satisfy

$$\text{RMSE}/\nu^{1/2} \le \text{MAE} \le \text{RMSE}. \tag{18.4}$$

Note also that the RMSE term (18.3) is closely related to $F(B)$, the objective function of the least squares problem (1.1). Substituting $B = X$ in (1.1) shows that

$$F(X) = \nu(\text{RMSE})^2. \tag{18.5}$$

However, since any admissible matrix gives a zero RMSE value, a small RMSE value does not necessarily imply high quality imputing. This exposes a certain weakness of the model evaluation approach. The modifications described in the next section attempt to overcome this difficulty.

19 The training set, the probe set, and cross-validation

In this approach the index set of known entries, Ω, is split into two distinct sets: A "training set", $\tilde{\Omega}$, and a "probe set", $\hat{\Omega}$, such that

$$\Omega = \tilde{\Omega} \cup \hat{\Omega} \quad \text{and} \quad \tilde{\Omega} \cap \hat{\Omega} = \emptyset. \tag{19.1}$$

Then the algorithm achieves imputing using $\tilde{\Omega}$ instead of Ω, generating predicted values, p_{ij}, for all entries in positions $(i,j) \notin \tilde{\Omega}$. Therefore, since for $(i,j) \in \hat{\Omega}$ both p_{ij} and a_{ij} are available, the prediction error is estimated from this set. Thus, for example,

$$\text{MAE} = \Big(\sum_{(i,j)\in\hat{\Omega}} |p_{ij} - a_{ij}| \Big) / \hat{\nu}, \tag{19.2}$$

where $\hat{\nu}$ denotes the number of entries in the probe set, and

$$\text{RMSE} = \Big((1/\hat{\nu}) \sum_{(i,j)\in\hat{\Omega}} (p_{ij} - a_{ij})^2 \Big)^{1/2}. \tag{19.3}$$

The implementation of this idea raises two crucial questions: How many entries to include in the probe set, and how to choose these entries. The answers depend on the specific properties of the matrix at hand: The size of the matrix, the percentage of missing entries, the way the missing entries are spread in the matrix, and the applications intended for the matrix. For example, in Collaborative Filtering it is common to take the probe entries from the most recent ratings, e.g., [256], [257]. In the Netflix Prize problem [203] the probe set is chosen in a random way. (But the actual evaluation process is more complicated for security reasons.)

The use of a training set and a probe set plays an important role in the statistical "Cross-Validation" method, e.g., [16, 126, 149, 237, 244]. In this methodology the probe set is often called "test set" or "validation set". The basic procedure is called "**k-fold Cross-Validation**". Here the set of known entries, Ω, is randomly split into k disjoint subsets, $\Omega_p, \;\; p = 1, \ldots, k$ of similar size. Then the imputing process is repeated k times. At the pth round, $p = 1, 2, \ldots, k$, Ω_p is used as probe set, and the union of the other $k-1$ subsets serves as training set. This way, at the end of the k rounds each known entry has a predicted value , p_{ij}, and the error parameters are computed by using (19.2)-(19.3) with Ω instead of $\hat{\Omega}$.

In cross-validation terminology the error criteria (18.1)-(18.3) are called **"Resubstitution Validation"**, while the use of one probe set, as in (19.1)-(19.3) is called **"Hold-out Validation"**. The strategy of using the largest number of folds is called **"Leave-One-Out Cross-Validation"** (LOOCV). This option is often used in linear model selection, e.g., [244], but in matrix completion problems it might be "too expensive", since here each subset Ω_p is a singleton $\{a_{ij}\}$ that refers to a known entry of A, and the number of "rounds" equals the number of known entries. In "Repeated Hold-out Validation" the Hold-out procedure is carried out a number of times, each time with a different probe set. Then an improved MAE value (or RMSE value) is gained by taking the mean value over all the runs. Another common strategy is "**Repeated k-fold cross-validation**", e.g., [273]. The choice of an "optimal" cross-validation strategy is intensively discussed in the statistical literature. For example, in data mining and machine learning applications a 10-fold cross-validation is a common choice, e.g., [16] and [149]. However, it seems that these issues have not yet received proper attention in the context of matrix completion.

Usually cross-validation is used for three related purposes. Assessment of model's prediction error, tuning model's parameters, and model selection, e.g., [237] and [244]. In particular, it is often used to determine the number of neighbors in k-Nearest Neighbors models, e.g., [121] and [199]. In the next section we illustrate the use of Hold-out validation to determine a suitable matrix rank when solving least squares problems.

20 Determining the rank

We have seen that many algorithms achieve imputing by constructing a low-rank matrix that approximates A. Yet the question of how to determine the rank of this matrix is discussed in a small number of papers. In the SVP algorithm [190] the matrix rank is determined by inspecting the singular values of $P_\Omega(A)$. The singular values are computed one after another until a significant gap is found. In OptSpace [145] the rank is determined by inspecting the rank of a "trimmed" version of $P_\Omega(A)$. In JELLYFISH [226] the rank is chosen in two stages. The initial rank is an integer close to the ratio $\nu/[3(m+n)]$. (This choice is aimed to ensure local strong convexity.) Then, after running a few iterations, the SVD of the current approximating matrix is checked in attempt to find a "gap" in the singular values spectrum. If a significant gap is found then the rank is reduced accordingly.

In several nuclear norm methods the rank is implicitly determined by the regularization parameter, λ, as the shrinkage operator S_λ annihilates singular values which are smaller than λ. A similar situation exists in the Hard-Impute algorithm [188].

In the RTR algorithm of Mishra et al. [194], and in GRIP, the matrix rank is gradually increased until a satisfactory solution is reached, but the meaning of "satisfactory" has to be clarified. Below we outline a simple "cross-validation" procedure for determining a suitable matrix rank, k, when solving (1.1).

The idea is to watch the behavior of a "training function" against a "probe function" as k increases. For this purpose we use a splitting of Ω into a "training set", $\tilde{\Omega}$, and a "probe set", $\hat{\Omega}$, as in (19.1). The number of entries in these sets are denoted by $\tilde{\nu}$ and $\hat{\nu}$, respectively. Let the matrix $\tilde{B}_k = (b_{ij}^{(k)})$, $k = 1, \ldots, n$, denote the solution of (1.1) when this problem is defined with $\tilde{\Omega}$ instead of Ω. Let

$$\tau_k = \Big((1/\tilde{\nu}) \sum_{(i,j)\in\tilde{\Omega}} (b_{ij}^{(k)} - a_{ij})^2 \Big)^{1/2} \tag{20.1}$$

and

$$\pi_k = \Big((1/\hat{\nu}) \sum_{(i,j)\in\hat{\Omega}} (b_{ij}^{(k)} - a_{ij})^2 \Big)^{1/2} \tag{20.2}$$

denote the related RMSE values. Then τ_k measures the ability of $\tilde{B}_k$ to approximate the training entries, and π_k measures its ability to approximate the probe entries. That is, π_k reflects the quality of the imputed entries.

The behavior of the "training sequence", $\tau_1, \tau_2, \ldots$, is characterized in Theorem 4:

$$\tau_1 \geq \tau_2 \geq \cdots \geq \tau_n = 0, \tag{20.3}$$

and there exists a "minimum rank" index, k^*, such that $\tau_k > 0$ for $k < k^*$, and $\tau_k = 0$ for $k \geq k^*$.

The first terms of the "probe sequence", $\pi_1, \pi_2, \ldots$, are likely to follow the decreasing behavior of the training sequence. The decreasing is expected as long as the least squares problem that we solve is highly over-determined and has a unique solution. As we have seen, this requires that the number of degrees of freedoms, $k(m + n - k)$, should stay smaller than the

number of known entries, $\tilde{\nu}$. However, as k increases beyond a certain point, the solution of the least squares problem loses its uniqueness. At that point the quality of the imputed entries starts to deteriorate and the probe sequence starts to behave in a somewhat "random" way.

The "optimal" matrix rank attempts to minimize both kinds of errors. That is, we seek a value of k for which both τ_k and π_k attain small values. The sequences $\{\tau_k\}$ and $\{\pi_k\}$ can be computed via the gradual rank increasing process (GRIP), and this process is stopped as soon as the probe sequence $\{\pi_k\}$ stops to decrease. Then a brief inspection of the two sequences is often sufficient to determine a good value for k, e.g., [59, 61].

21 Choosing an algorithm

The choice of a method for solving a specific imputing problem depends on several factors. One factor is the size of the matrix at hand. For example, in very large problems one may consider the use of gradient descent methods that require a minimal amount of extra storage and low computational effort per iteration. In small problems one may consider faster methods, such as Newton's method. The advent of parallel computing suggests its use for solving large problems. As we have seen, some methods are easily adapted to this mode (ALS and JELLYFISH).

A second factor is the percentage of missing entries. Many methods face difficulties when only a small percentage of entries is known. The experience gained with random problems suggests that the rank, k, of the approximating matrix should satisfy an inequality of the form

$$k(m+n-k) < \nu/3.$$

That is, the number of degrees of freedom should be considerably smaller than the number of known entries. Hence a small percentage of known entries forces the use of low-rank matrices. Another common remedy is the addition of a regularization term. See Sections 8.6 and 10.4. On the other hand, a small percentage of known entries causes $P_\Omega(A)$ to be a sparse matrix. This turns out to be an advantage in certain gradient descent methods, and in methods that are based on iterative TSVD.

A third factor regards the origin of the data and known properties of the data matrix. For example, we have seen that Euclidean distance matrices have special solution methods. Similarly, if the rows of the matrix can be grouped into clusters of similar rows, as in DNA microarray data, then KNN imputing is a reasonable choice. In some cases the data has a specific interpretation that suggests the use of certain low-rank models. Also, as noted in Section 8.13, often a hybrid strategy that combines a number of methods gives the best results. (The Netflix winning team [19, 151, 152] has used a hybrid strategy.)

Another important factor is the application intended for the imputed data. For example, if the imputed matrix is needed for Factor Analysis then an iterative SVD method might be suitable. A different type of application arises in recommender systems that achieve imputing in order to suggest the customer items to buy. This is often done under titles like "frequently bought together ...", or "customers who bought this item also bought ...". For this task nearest

neighbors methods, like KNN, might be suitable. (Since there is no need to complete the whole matrix in order to give such a recommendation.)

Finally we present five tables that might be helpful when choosing an algorithm. Tables 1 and 2 list the methods described in Sections 6-10, mentioning the optimization technique that is used and the size of problems that they intend to solve. Table 3 summarizes the experience gained with these methods. Many experiments were carried out by authors who proposed a new algorithm. In this case the name of the new algorithm is mentioned in the first column. The second column of Table 3 specifies the other methods which participated in the experiment, while the third column gives the test problems. Details on common test problems are given in Table 4. Table 5 provides references to various kinds of applications and methods. Codes for several imputing algorithms can be freely downloaded from

http://perception.csl.uiuc.edu/matrix-rank/

The experiments reported in Table 3 compare methods by recording the number of iterations and the computation time which are needed for reducing the RMSE value below a given threshold. However, so far, it is difficult to point on a certain method that has a clear advantage over other competitors.

Table 1: Examples of least squares methods

Algorithm and References	Optimization method	Matrix size	Section
Iterative SVD [125, 156, 190, 269]	Proximal point, dense/sparse SVD	medium/ large	6
FRAA [88, 89]	Minimizing the singular values "tail"	medium	7
SROM [59, 61]	Successive rank-one modifications	large	8.3
ALS [33, 186]	Alternating least squares	large	8.4
LMaFit [277]	Proximal ALS with line search	large	8.7
OptSpace [145]	Gradient descent with orthogonal factors	large	8.8
Funk's scheme [92]	"Sparse" gradient descent	large	8.8
JELLYFISH [226]	"Parallel" gradient descent	huge	8.8
Newton method [33]	L-M Newton method	medium	8.9
Wiberg's algorithm [208, 278]	"Separable" Gauss-Newton	medium	8.9
Separable Newton [43]	"Separable" L-M Newton methods	medium	8.9
Newton-Grassmann [43, 77]	Newton method on the Grassmann manifold	medium	8.10
GROUSE [18]	Gradient descent on the Grassman manifold	large	8.11
SET [51]	Grassmannian gradient descent with line-search	large	8.11
LRGeom CG [272]	Riemannian Conjugate Gradient	large	8.12
RTRMC [26, 27]	Riemannian Trust-Region	large	8.12

Table 2: Examples of rank minimization methods and nuclear norm methods

Algorithm and References	Optimization method	Matrix size	Section
Hard-Impute [188]	Rank minimization via Iterative SVD	medium/ large	6, 9
ADMIRA [165, 167]	Rank min. by selecting optimal sets of atoms		9
IRLS-GP [198]	Rank min. via gradient projection IRLS	large	9
SDP [38]	Nuclear norm min. via Semidefinite Programming	small	10.1.3
SVT [35]	Regularized nuclear norm min. via Uzawa's algorithm	large	10.3
Soft-Impute [188]	Regularized nuc. norm min. via a Proximal Point algorithm	large	10.4
FPC/FPCA [180]	Regularized nuclear norm min. via gradient descent	medium/ large	10.4
APGL [261]	Regularized nuc. norm min. via Accelerated Proximal Gradient with Line-search	large	10.4
RTR [194]	Regularized nuc. norm min. via a Riemannian Trust-Region algorithm	large	10.4
ADM [42]	Augmented Lagrangian via alternating optimization	large	10.5
Primal Dual PPA [176]	Augmented Lagrangian via alternating proximal point approximations	large	10.5
AL [225]	Augmented Lagrangian	large	10.5

Table 3: Examples of reports on numerical comparison of methods

Proposed algorithm and reference	Methods included	Test problems
Hastie et al. [125]	KNN, ICR, Restricted SVD, Iterative SVD	DNA
Troyanskaya et al. [269]	KNN, Iterative SVD, Row averages	DNA
Markovsky [186]	ALS, L-M separable Newton, SVT	R, ML
Michenkova [191]	LMaFit, OptSpace, GROUSE, SET, SDP, SVT, FPCA, APGL, ALM [169], ASALM [258]	R, J, ML
Iterative SVD [190]	ALS, ADMIRA, SVT, OptSpace	R, ML
LMaFit [277]	OptSpace, FPCA, APGL	R, J, ML, IM
OptSpace [145]	ADMIRA, SVT, FPCA	R, J, ML
JELLYFISH [226]	APGL	R, ML, N
L-M Newton [43]	L-M Newton, Newton-Grassmann, Wiberg, ALS	SFM
GROUSE [18]	OptSpace, SDP, SVT, FPCA, APGL	R
SET [51]	OptSpace, ADMIRA, SVT	R
LRGeom CG [272]	LMaFit	R
RTRMC [26, 27]	LMaFit, OptSpace, ADMIRA, SVT	R, J*
Soft/Hard-Impute [188]	OptSpace, SVT	R
IRLS-GP [198]	OptSpace, SVT, FPC, FPCA,	R, ML
FPC/FPCA [180]	SDP, SVT,	R, J*, DNA, IM
APG/APGL [261]	FPC,	R, J, ML
RTR [194]	LMaFit, LRGeom, SVT, Soft Impute, FPCA, APG	R
ADM [42]	SVT, FPCA, APGL, Dual PPA	R, J
Primal/Dual PPA [176]	SVT	R, N*
ALM [169]	SVT, APG, APGL	R

*Using a probe function

Table 4: Common types of test problems

Problems	References	Abbreviation
Random matrices	Section 10.6	R
Netflix data	[20, 92, 203, 256]	N
Jester data	[26, 105, 180, 257, 261]	J
Movi Lense data	[49, 198, 257, 261]	ML
DNA microarray data	[125, 269]	DNA
Structure from Motion data	[33, 43, 44]	SFM
Image inpainting	[180, 277]	IM

Table 5: Examples of applications and related topics

Application	References
Statistical treatment of missing data	[5, 10, 14, 15, 64, 73, 101, 120, 123, 153, 172, 173, 187, 230, 231, 236, 276, 278]
Business reports	[52, 90]
Operations management	[270]
Psychometrika	[86, 146, 266]
Hydrology and Meteorology	[47, 53, 61, 129, 143, 155, 158, 252]
Biology and Medicine	[68, 78, 137, 213, 214, 217, 279]
DNA microarray data, Clustering methods	[9, 11, 12, 13, 45, 74, 111, 148, 216, 259]
DNA microarray data, Imputing methods	[24, 31, 32, 88, 89, 94, 125, 147, 204, 242, 269, 275, 286]
Computer vision and SFM	[28, 30, 33, 44, 117, 134, 135, 208, 264]
Recommender systems and collaborative filtering	[7, 19, 20, 40, 46, 92, 104, 127, 128, 151, 152, 156, 203, 209, 212, 255, 256, 257, 289]
Data mining, information retrieval and machine learning	[97, 157, 182, 219, 220, 221, 222, 246, 290, 291]
Chemometrics	[116, 201, 262, 274, 284]
Principal Components Analysis (PCA) and Factor analysis	[71, 87, 116, 131, 132, 133, 164, 215, 220, 221, 232, 246, 278, 282, 297]
Robust PCA methods	[36, 169, 170, 245, 285]
Euclidean distance matrices and other special matrices	[79, 96, 102, 112, 114, 138, 139, 140, 141, 161, 162, 163, 168, 268]
Missing values in Tensors	[4, 41, 95, 107, 113, 122, 150, 154, 174, 175, 223, 247, 261, 262, 265, 274]
Artificial Neural Networks (ANN)	[1, 78, 98, 119, 143, 155, 183, 184, 195, 196, 197, 202, 252, 294]
Genetics algorithms	[1, 196, 197, 202, 213, 218, 294, 295]

22 Concluding remarks

The need for imputing missing entries of a data matrix arises in many applications. The current list of references is only a sample of the vast literature on this issue. The interest in the problem has been rapidly increasing in recent years, especially in "modern" areas, such as computer vision, DNA microarray data, and recommender systems. The review exposes a surprising variety of imputing techniques. It concentrates on the ideas behind the methods and the ways these ideas are transformed into working algorithms. This enables us to identify a number of basic principles and tools which are shared by several algorithms.

Observe, for example, that KNN, ICR, restricted SVD, and iterative SVD, are based on a similar principle: The missing entries in a certain row (or column) are computed by comparing this row against some other rows (or columns). Similarly, several iterative algorithms proceed by generating a sequence of admissible matrices, $A_\ell \in \mathbb{A}$, $\ell = 1, 2, \ldots$, such that $A_{\ell+1}$ is "better" than A_ℓ in some sense. In many methods A_ℓ is obtained from a low-rank matrix, B_ℓ, that solves a certain "proximal" problem. The sequence $\{B_\ell\}$ is designed to converge toward a solution of (1.1), or a regularized form of this problem. In nuclear norm methods A_ℓ and B_ℓ are replaced by Z_ℓ and X_ℓ, respectively. Note the similarity between the regularized least squares problem (8.17), the regularized nuclear norm problem (10.38), and the Lagrange functions in Section 10.5.

Another idea that repeats a number of times is that of the alternating least squares (ALS) algorithm. It is this feature that makes the problem "separable", a property which is used by several algorithms. Similar least squares problems are solved in FRAA, GROUSE, IRLS, and in the Augmented Lagrangian method for solving (10.59). Further types of linear least squares problems arise in OptSpace and in Trosset's method.

The field of matrix imputing methods is rich in elegant schemes and ingenious ideas. Some of the more popular methods are based on simple intuitive ideas, e.g., KNN, ICR, Iterative SVD, and ALS. On the other side of the spectrum we can find highly sophisticated methods like FRAA, minimization on Grassmann manifolds, or Trosset's algorithm. Artificial Neural Networks and Genetics algorithms are examples of imported tools which are harnessed to solve the imputing problem. Recently the Netflix competition has attracted a lot of attention and motivated several innovations, such as Funk's steepest descent scheme, rank minimization, and nuclear norm minimization.

Indeed the nuclear norm approach has brought several new ideas. Perhaps the most remarkable one regards the possibility to achieve exact recovery. The first attempt to implement this approach relied on Semidefinite Programming, but this method is not suitable for large problems. Recent algorithms are based on the shrinkage operator, S_λ, which is computed via efficient implementations of Lanczos method or a Monte Carlo method. This innovation paves the way for solving large nuclear norm problems. The role of S_λ is similar to that of the Truncated SVD operator, T_k. As we have seen, the basic scheme for solving regularized nuclear norm problems can be viewed as variant of Iterative SVD in which T_k is replaced by S_λ. A third matrix operator, E_ε, is used in Robust PCA methods.

Although there is a plethora of methods, the imputing problem is not always "solvable"

in the sense of achieving satisfactory recovering. Consider for example a permutation matrix from which some ones and zeros are deleted. Then none of the methods mentioned above is able to trace the original matrix. Another example is a "random" matrix whose entries are "random numbers" that obey the same distribution function. In this case the best that we can do is to substitute each missing entry by the average value of the known entries. This solution might be satisfactory in some cases, but there is no way to restore the "true" values of lost entries. We have seen that exact recovery is expected under certain conditions. Yet it is difficult to find real situations in which this property holds.

The fact that the problem is not always solvable stresses the importance of using "probe" functions to assess the quality of the imputed entries. This "cross-validation" idea can be used to determine a proper matrix rank, k, when solving (1.1).

A further conclusion that stems from our survey is that no single algorithm is able to handle all types of problems. The choice of a suitable imputing method depends on the origin of the data, on known properties of the data, and the applications intended for the data.

List of Symbols

$A = (a_{ij})$ a real $m \times n$ matrix with missing entries.

$\Omega = \{(i,j) \mid a_{ij}$ is known$\}$ the set of known entries.

ν the number of known entries in A.

P_Ω a matrix operator that sets zeros in entries $(i,j) \notin \Omega$.

$\mathbb{A} = \{X = (x_{ij}) \in \mathbb{R}^{m \times n} \mid x_{ij} = a_{ij}$ when a_{ij} is known$\}$ the set of admissible matrices.

$\mathbb{B}_k = \{B \mid B \in \mathbb{R}^{m \times n}$ and rank $(B) \leq k\}$ the set of low-rank matrices.

$\|\mathbf{x}\|_2$ the Euclidean norm of a vector **x**.

$\|X\|_F$ the Frobenius norm of a matrix X.

$\|X\|_*$ the nuclear norm of a matrix X. (The sum of singular values.)

$T_k(X)$ the rank-k truncated SVD of a matrix X.

$\tau_k(X)$ the "tail" function of a matrix X.

$S_\lambda(X)$ the "shrinkage" of a matrix X.

$< X, Y >=$ trace$(X^T Y)$ the inner product between matrices.

$\mathbb{G}_k^m$ the Grassmann manifold: The set of all k-dimensional subspaces in $\mathbb{R}^m$.

$[B]$ the k-dimensional subspace of $\mathbb{R}^m$ which is spanned by the columns of a rank-k matrix $B \in \mathbb{R}^{m \times n}$.

$\mathbb{B}_k^* = \{B \in \mathbb{R}^{m \times n}$ and rank$(B) = k\}$ a Riemannian manifold.

$\mathbb{T}_B$ the tangent space of $\mathbb{G}_k^m$ (or $\mathbb{B}_k^*$) at a point $[B] \in \mathbb{G}_k^m$ (or $B \in \mathbb{B}_k^*$).

References

[1] M. Abdella and T. Marwala, *The use of genetic algorithms and neural networks to approximate missing data in database*, Computing and Informatics, **24** (2006), 1001-1013.

[2] P.-A. Absil, C.G. Baker and K.A. Gallivan, *Trust-region methods on Riemannian manifolds*, Foundations of Computational Mathematics **7**(2007), 303-330.

[3] P.-A. Absil, R. Manony and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*, Princeton University Press, Princeton, NJ, 2008.

[4] E. Acar, D.M. Dunlavy, T.G. Kolda and M. Mørup, *Scalable tensor factorizations with incomplete data*, Chemometrics Intelligent Laboratory Systems, **106**(2011), 41-56.

[5] A.A. Afifi and R.M. Elashoff, *Missing observations in multivariate statistics. I: Review of the literature*, J. Amer. Statist. Ass., **61**(1966), 595-604.

[6] A.Y. Alfakih, A. Khandani and H. Wolkowicz, *Solving Euclidean Distance Matrix Completion Problems via Semidefinite Programming*, Computational Optimization and Applications **12**(1999), 13-30.

[7] K. Ali and W. Van Stam, *Making Show Recommendations Using a Distributed Collaborative Filtering Architecture*, in *Proceedings of the Tenth ACVM SIGKDD Conference (KDD'04)*, Seattle, August 2004.

[8] K. Ali and W. Van Stam, *Tivo: making show recommendations using a distributed collaborative filtering architecture*, in W. Kim, R. Kohavi, J. Gehrke and W. DuMouchel, editors, KDD, pps. 394-401. ACM, 2004.

[9] A.A. Alizadeh, et al., *Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling*, Nature **403**(2000), 503-511.

[10] P.D. Allison, *Missing Data*, pp. 72-89 in the SAGE Handbook of quantitative methods in Psychology, edited by R.E. Milesap and A. Maydeu-Olivars, Thousand Oak, 2009.

[11] U. Alon, et al., *Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligo-nucleotide arrays*, Proc. Natl. Acad. Sci. USA **96**(1999), 6745-6750.

[12] Alter, P.O. Brown and D. Botstein, *Singular value decomposition for genome-wide expression data processing and modeling*, Proc. Natl. Acad. Sci. USA **97**(2000), 10101-10106.

[13] O. Alter, P.O. Brown and D. Botstein, *Processing and modeling genome-wide expression data using singular value decomposition*, Proceedings SPIE **4266**(2001), 171-186.

[14] T.W. Anderson, *Maximum likelihood estimates for a multivariate normal distribution when some observations are missing*, J. Amer. Statist. Ass. **52**(1957), 200-203.

[15] R.R. Andridge and R.J.A. Little, *A review of Hot Deck Imputation for survey non-response*, International Statistical Review **78**(2010), 40-64.

[16] S. Arlot and A. Celisse, *A survey of cross-validation procedures for model selection*, Statistics Surveys **4**(2010), 40-79.

[17] M. Bakonyi and C.R. Johnson, *The Euclidean distance matrix completion problem*, SIAM J. Matrix Analysis and Applications **16**(1995), 646-654.

[18] L. Balzano, R. Nowak and B. Recht, *Online identification and tracking of subspaces from highly incomplete information*, in Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on IEEE, 2010, pp. 704-711.

[19] R. Bell and Y. Koren, *Scalable Collaborative Filtering with Jointly Derived Neighborhood Interpolation Weights*, Proc. IEEE Int'l Conf. Data Mining (ICDM 07), IEEE CS Press, 2007, pp. 43-52.

[20] J. Bennet and S. Lanning, *The Netflix Prize*, KDD Cup and Workshop, 2007; www.netflixprize.com.

[21] D.P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*, Academic Press, NY, 1982 (republished in 1996 by Athena Scientific, Belmont, MA).

[22] D.P. Bertsekas, *Nonlinear Programming*, Athena Scientific, Belmont, MA, 1995.

[23] A. Bjorck, *Numerical Methods for Least-squares Problems*, SIAM, Philadelphia, 1996.

[24] T.H. Bó, B. Dysvik and I. Jonassen, *LSimpute: accurate estimation of missing values in microarray data with least squares methods*, Nucleic Acids Res., **32(3)**:34 (2004).

[25] I. Borg and P.J. Groenen, *Modern multi-dimensional scaling theory and application*, Chapman-Hall, 2001.

[26] N. Boumal and P.-A. Absil, *RTRMC: A Riemannian trust-region method for low-rank matrix completion*, in Advances ihn Neural Information Processing Systems 24 (NIPS), J. Shawe-Taylor, R.S. Zemel, P. Bartlett, F.C.N. Pereira and K.Q. Weinberger, eds., 2011, pp. 405-414.

[27] N. Boumal and P.-A. Absil, *Low-rank matrix completion via trust-regions on the Grassmann manifold*, Tech. Report UCL-INMA - 2012.07-v1

[28] M. Brand, *Incremental singular value decomposition of uncertain data with missing values*, in: *Proc. European conf. computer vision*, Lecture notes on computer Sciences, pp. 707-720, Springer-Verlag, 2002.

[29] M. Brand, *Fast online SVD revisions for lightweight recommender systems*, Proceedings, SIAM 3rd International Conference on Data Mining, pp. 37-46, 2003.

[30] S. Brandt, *Closed-form solutions for affine reconstruction under missing data*, in: Statistical methods for video processing workshop, in conjunction with ECCV02, pp. 109-114.

[31] N. Brock, J.R. Shaffer, R.E. Blakesley, M.J. Lotz and G.C. Tseng, *Which missing value imputation method to use in expression profiles: A comparative study and two selection schemes*, BMC Bioinformatics 2008, 9:12.

[32] P. Brown, T. Hastie, R. Tibshirani, D. Botstein and R.B. Altman, *Missing value estimation methods for DNA microarrays*, Bioinformatics **17**(2001), 520-525.

[33] A.M. Buchanan and A.W. Fitzgibbon, *Damped Newton Algorithms for Matrix Factorization with Missing Data*, Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), **2**(2005), 316-322.

[34] S. F. Buck, *A method of estimating missing values in multivariate data suitable for use with an electronic computer*, Journal of the Royal Statistical Society, Series B, **22**(1960), 302-306.

[35] J.-F. Cai, E.J. Candès and Z. Shen, *A Singular Value Thresholding Algorithm for Matrix Completion*. Manuscript, 2008.

[36] E.J. Candès, X. Li, Y. Ma and J. Wright, *Robust principal component analysis?* Journal of the ACM **58**(2011).

[37] E.J. Candès and Y. Plan, *Matrix Completion with Noise*, Manuscript, 2009.

[38] E.J. Candès and B. Recht, *Exact Matrix Completion via Convex Optimization*, Foundations of Computational Mathematics **9** (2009), 717-772.

[39] E.J. Candès and T. Tao, *The Power of Convex Relaxation: Near-Optimal Matrix Completion*. Manuscript, 2009.

[40] J. Canny, *Collaborative filtering with privacy via factor analysis*, in SIGIR '02 Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 238-245, New York, NY, 2002. ACM Press.

[41] J.D. Carrol and J.J. Chang, *Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart-Young" decomposition*, Psychometrika **35**(1970), pp. 283-319.

[42] C. Chen, B. He and X. Yuan, *Matrix completion via alternating direction method*, IMA J. of Numerical Analysis **32**(2011), 227-245.

[43] P. Chen, *Optimization algorithms on subspaces: Revisiting missing data problem in low-rank matrix*, Int. J. Comput. Vis., **80**(2008), 125-142.

[44] P. Chen and D. Suter, *Recovering the missing components in a large noisy low-rank matrix: Application to SFM*, IEEE Transactions on Pattern Analysis and Machine Intelligence **26**(2004), 1051-1063.

[45] H. Chipman, T. Hastie and R. Tibshirani, *Clustering microarray data*, in T. Speed (ed.), Statistical Analysis of Gene Expression Microarray Data, Boca Raton, FL, Chapman and Hall, pp. 159-199, 2003.

[46] M. Clements, A.P. de Vries, J.A. Pouwelse, J. Wang and M.J.T. Reinders, *Evaluation of Neighbourhood Selection Methods in Decentralized Recommendation Systems*, in *Proceedings of SIGIR Workshop on Large Scale Distributed Systems for Information Retrieval (LSDS-IR)*, pp. 38-45, Amsterdam, 2007.

[47] P.A. Conards and M.D. Petkewich, *Estimating of missing water-level data for the Everglades Depth Estimation Network (EDEN)*, U.S. Geological Survey Open-File Report 2009-1102, 2009, 52 pages.

[48] G.M. Crippen and T.F. Havel, *Distance Geometry and Molecular Conformation*, Wiley:New York, 1988.

[49] B.J. Dahlen, J.A. Konstan, J. Herlocker, N. Good, A. Borchers and J. Riedl, *Movie lens data*, 1998. http//www.grouplens.org/node/73.

[50] W. Dai, E. Kerman and O. Milenkovic, *Low-rank matrix completion with geometric performance guarantees*, in *International Conference on Acoustics, Speech, and Signal Ptocessing (ICASSP)*, Prague, Czech Republic, 2011.

[51] W. Dai, E. Kerman and O. Milenkovic, *Subspace evolution and transfer (SET) for low-rank matrix completion*, IEEE Transactions on Signal Processing, 2011.

[52] P. Davies and P. Smith, *Model Quality Reports in Business Statistics*, ONS, UK, 1999.

[53] A. Dax, *Completing missing groundwater observations by interpolation*, Journal of Hydrology **81**(1985), 375-399.

[54] A. Dax, *The distance between the two convex sets*, Linear Algebra and its Applications **416**(2006), 184-213.

[55] A. Dax, *Orthogonalization via delfation: A minimum norm approach for low-rank approximations of a matrix*, SIAM Journal on Matrix Analysis and Applications, **30**(2008), 236-260.

[56] A. Dax, *A hybrid algorithm for solving linear inequalities in a least squares sense*, Numerical Algorithms **50**(2009), 97-114.

[57] A. Dax, *A Minimum Norm Approach for Low-Rank Approximations of a Matrix*, Journal of Computational and Applied Mathematics, **234**(2010), 3091-3103.

[58] A. Dax, *On extremum properties of orthogonal quotient matrices*, Linear Algebra and its Applications, **432**(2010), 1234-1257.

[59] A. Dax, *A gradual rank increasing process for matrix completion*, Tech. Rep., Hydrological Service of Israel, 2014.

[60] A. Dax, *Low-rank positive approximants of symmetric matrices*, Advances in Linear Algebra and Matrix Theory (ALAMT), **4**(2014), 172-185.

[61] A. Dax and M. Zilberbrand, *Imputing missing groundwater observations*, Tech. Rep., Hydrological Service of Israel, 2014.

[62] J. DeLeeuw and W. Heiser, *Theory of multidimensional scaling*, in Handbook of Statistics, P.R. Krishnaiah and L.N. Kanal (Eds.), North-Holland, 1982, vol. 2, pp. 235-316.

[63] J. DeLeeuw, *Multidimensional scaling*, Department of Statistics, UCLA, Tech. Rep. 2001. Available online at: http://preprents.stat.ucla.edu/274/247.pdf

[64] A.P. Dempster, N.M. Laird and D.B. Rubin, *Maximum likelihood from incomplete data via the EM algorithm*, Journal of the Royal Statistical Society, **39**(1977), 1-38.

[65] J.E. Dennis and R.B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, SIAM, Philadelphia, 1996.

[66] J.K. Dixon, *Pattern recognition with partly missing data*, IEEE Trans. Syst. Man Cybern. SMC-9 **10**(1979), 617-621.

[67] I. Dokmanic, R. Parhizkar, A. Walther, Y.M. Lu and M. Vetterli, *Acoustic echoes reveal room shape*, Proceedings of the National Academy of Sciences (PNAS) **110**(2013), 12186-12191.

[68] A.R.T. Donders, G.J.M.G. van der Heijden, T. Stijnen and K.G.M. Moons, *Review: A gentle introduction to imputation of missing values*, Journal of Clinical Edipemiology **59**(2006), 1087-1091.

[69] D.L. Donoho, *Compressed sensing*, IEEE Transactions on Information Theory, **52**(2006), 1289-1306.

[70] P. Drineas, R. Kannan and M.W. Mahoney, *Fast Monte Carlo algorithsm for matrices ii: Computing low-rank approximations to a matrix*, SIAM J. Computing **36**(2006), 158-183.

[71] G. Eckart and G. Young, *The approximation of one matrix by another of lower rank*, Psychometrika, **1**(1936), 211-218.

[72] A. Edelman, T. Arias and S. Smith, *The geometry of algorithms with orthogonality constraints*, SIAM J. Matrix Anal. Appl. **20**(1998), 303-353.

[73] G.L. Edgett, *Multiple regression with missing observations among the independent variables*, J. Amer. Statist. Ass. **51**(1956), 122-131.

[74] M. Eisen, P. Spellman, P. Brown and D. Botstein, *Cluster analysis and display of genome-wide expression patterns*, Proceedings of the National Academy of Sciences of the United States of America, **95**(1998), 14863-14868.

[75] L. Elden, *Partial least squares vs. Lanczos bidiagonalization I: Analysis of a projection methods for multiple regression*, Comput. Statist. and Data Anal. **46**(2004), 11-31.

[76] L. Elden, *Matrix Methods in Data Mining and Pattern Recognition*, SIAM, Philadelphia, 2007.

[77] L. Elden and B. Savas, *A Newton-Grassmann method for computing the best multilinear rank-*(r_1, r_2, r_3) *approximation of a tensor*, SIAM J. Matrix Anal. Appl. **31**(2009), pp. 248-271.

[78] C. M. Ennett, M. Frize and C. R. Walker, *Influence of missing values on artificial neural network performance*, published in *Proceedings of Medinfo, London*, September 2001, 10(Pt 1), 449-53.

[79] S.M. Fallat, C.R. Johnson, J.R. Torregrosa and A.M. Urbano, P*-matrix completions under weak symmetry assumptions*, Linear Algebra Appl. **312**(2000), 73-91.

[80] H. Fang and D.P. O'Leary, *Euclidean Distance Matrix Completion Problems*, Optimization Methods and Software **27**(2012), 695-717.

[81] M. Fazel, *Matrix Rank Minimization with Applications*, PhD thesis, Department of Electrical Engineering, Stanford University, Stanford, CA, 2002.

[82] M. Fazel, H. Hindi and S. Boyd, *A rank minimization heuristic with application to minimum order system approximation*, in *Proceedings of the American Control Conference*, Vol. 6, 2001, pp. 4734-4739.

[83] M. Fazel, H. Hindi and S. Boyd, *Log-det Heuristic for Matrix Rank Minimization with Applications to Hankel and Euclidean Distance Matrices*, in *Proc. American Control Conference*, Denver, Colorado, June 2003.

[84] R. Fletcher, *Practical Methods of Optimization*, Wiley, 1980.

[85] M. Fornasier, H. Rauhut and R. Ward, *Low-rank matrix recovery via iteratively reweighted least squares minimization*, SIAM J. Optimization **21**(2011), 1614-1640.

[86] J.W. Frane, *Some simple procedures for handling missing data in multivariate analysis*, Psychometrica **41**(1976), 409-415.

[87] S.B. Franklin, D.J. Gibson, P.A. Robertson, J.T. Pohlmann and J.S. Fralish, *Parallel analysis: a method for determining significant principal components*, Journal of Vegetation Science, **6**(1995), 99-106.

[88] S. Friedland, A. Niknejad and L. Chihara, *A simultaneous reconstruction of missing data in DNA microarrays*, Linear Algebra and its Applications, **416**(2006), 8-28.

[89] S. Friedland, M. Kaveh, A. Niknejad and H. Zare, *An algorithm for missing value estimation for DNA microarray data*, IEEE Proceedings of ICASSP 2006, II, 1092-1095.

[90] M. Friedman, *The interpolation of time series by related series*, J. Am. Stat. Assoc., **47**(1962), 729-757.

[91] A. Frieze, R. Kannan and S. Vempala, *Fast Monte-Carlo Algorithms for Finding Low-Rank Approximations*, Journal of the ACM **51**(2004), 1025-1041.

[92] S. Funk, *Netflix Update: Try This at Home*, Dec. 2006; http://sifter.org/ simon/journal/20061211.html

[93] K.R. Gabriel and S. Zamir, *Lower rank approximation of matrices by least squares with any choices of weights*, Technometrics, **21**(1970), 489-498.

[94] X. Gan, A.W. Liew and H. Yan, *Microarray missing data imputation based on a set theoretic framework and biological knowledge*, Nucleic Acids Res., **34(5)**(2006), 1608-1619.

[95] S. Gandy, B. Recht and I. Yamada, *Tensor completion and low-n-rank tensor recovery via convex optimization*, Inverse Problems **27**(2011).

[96] J.F. Geelen, *Maximum rank matrix completion*, Linear Algebra Appl. **288**(1999), 211-217.

[97] Z. Ghahramani and M.I. Jordan, *Learning from incomplete data*, C.B.G.L. Paper No. 108, MIT, 1994.

[98] M.K.I. Gill, T. Asefa, Y. Kaheil and M. McKee, *Effect of missing data on performance of learning algorithms for hydrologic predictions: Implication to an imputation technique*, Water Resources Research **43**(2007), 12 pages.

[99] P.E. Gill, W. Murray and M.H. Wright, *Practical Optimization*, Academic Press, London, 1981.

[100] N. Gillis and F. Gilneur, *Low-rank matrix approximation with weights or missing data in NP-hard*, SIAM J. Matrix Analysis Applications **32**(2011), 1149-1165.

[101] M. Glasser, *Linear regression analysis with missing observations among the independent variables*, J. Amer. Statist. Ass., **59**(1964), 834-844.

[102] D.F. Gleich and L. Lim, *Rank aggregation via nuclear norm mininization*, in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, ACM, New York, NY, USA, 2011, pps. 60-68.

[103] W. Glunt, T.L. Hayden and M. Raydan, *Molecular conformation from distance matrices*, J. Comp. Chem. **14**(1993), 114-120.

[104] K. Goldberg, T. Roeder, D. Gupta and C. Perkins, *Eigentaste: A constant time collaborative filtering algorithm*, Inf. Retr., **4(2)**(2001), 133-151.

[105] K. Goldberg, D. Gupta, M. Digiovanni and H. Narita, *Jester 2.0: Evaluation of a new linear time collaborative filtering algorithm*, in *22nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, August 1999.

[106] D. Goldfarb and S. Ma, *Convergence of fixed point continuation algorithms for matrix rank minimization*, Tech. rep., Department of IEOR, Columbia University (2009).

[107] D. Goldfarb and Z. Qin, *Robust Low-rank Tensor Recovery: Models and Algorithms*, to appear in SIAM Journal on Matrix Analysis and Applications, 2013.

[108] G.H. Golub and V. Pereyra, *The differentiation of pseudoinverses and nonlinear least squares problems whose variables separate*, SIAM Journal on Numerical Analysis **10**(1973), 413-432.

[109] G.H. Golub and V. Pereyra, *Separable nonlinear least squares: The variable projection method and its applications*, Inverse Problems **19**(2003), 1-26.

[110] G.H. Golub and C.F. Van Loan, *Matrix Computations*, Johns Hopkins Univ. Press, 1983.

[111] T.R. Golub, et al, *Molecular classification of cancer: class discovery and class prediction by gene expression monitoring*, Science, **286**(1999), 531-537.

[112] T. Graepl, *Kernel Matrix Completion by Semidefinite Programming*, Institute of Computational Science, ETH Zurich, Switzerland.

[113] L. Grasedyck, D. Kressner and C. Tobler, *A literature survey of low-rank tensor approximation techniques*, GAMM-Mitt **36**(2013), 53-78.

[114] R. Grone, C.R. Johnson, E. Sa and H. Wolkowicz, *Positive definite completions of partial Hermitian matrices*, Linear Algebar Appl. **58**(1984), 109-124.

[115] I.G. Grooms, R.M. Lewis and M.W. Trosset, *Molecular embedding via a second order dissimilarity parameterized approach*, SIAM J. Sci. Comput. **31**(2009), 2733-2756.

[116] B. Grung and R. Manne, *Missing values in principal component analysis*, Chemometrics and Intelligent Laboratory System, **42**(1998), 125-139.

[117] R.F.C. Guerreiro and P.M.Q. Aguiar, *Estimation of rank deficient matrices from partial observations: Two-step iterative algorithms*, in *Lecture Notes in Computer Science: Energy Minimization Methods in Computer Vision and Pattern Recognition*, volume 2683, Springer-Verlag, July 2003.

[118] A. Gunawardana and G. Shani, *A Survey of Accuracy Evaluation Metrics of Recommendation Tasks*, Journal of Machine Learning Research (JMLR) **10**(2009), 2935-2962.

[119] A. Gupta and M. S. Lam, *Estimating missing values using neural networks*, Journal of the Operational Research Society **47**(1996), 229-238.

[120] Y. Haitovsky, *Missing data in regression analysis*, J.R. Stat. Soc. B, **30**(1968), 67-82.

[121] G. Hamerly and G. Speegle, *Efficient model selection for large-scale nearest-neighbor data mining*, in *Proceedings of the 2010 British National Conference on Databases (BNCOD 2010)*, pp. 37-54, 2010.

[122] R.A. Harshman, *Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis*, UCLA Working Papers in Phonetics **16**(1970), pp. 1-84. Available at: http://publish.uwo.ca/ harshman/wpppfac0.pdf

[123] H.O. Hartely, *Maximum likelihood estimation from incomplete data*, Biometrics, **14**(1958), 174-194.

[124] R. Hartley and F. Schaffalitzky, *Powerfactorization: 3d reconstruction with missing or uncertain data*, Tech. Rep., Research School of Information Sciences, Australian National University, Australia, 2003.

[125] T. Hastie, R. Tibshirani, G. Sherlock, M. Eisen, P. Brown and D. Botstein, *Imputing missing data for gene expression arrays*, Technical Report, Division of Biostatistics, Stanford University, 1999.

[126] T. Hastie, R. Tibshirani and J. Friedman, *The elements of statistical learning*, Springer Series in Statistics, Springer, New York, Second edition, 2011.

[127] J. Herlocker, J. Konstan, A. Borchers and J. Riedl, *An algorithmic framework for performing collaborative filtering*, in *Proc. ACM SIGIR*, 1999.

[128] J.L. Herlocker, J.A. Konstan, L.G. Terveen and J.T. Riedl, *Evaluating collaborative filtering recommender systems*, ACM Transactions on Information Systems **22**(2004).

[129] R.M. Hirsch, *A comparison of four record extension techniques*, Water Resources Research **15**(1979), 1781-1790.

[130] O.V. Holtz, *Compressive sensing: A paradigm shift in signal processing*, Tech. Rep., University of California, Berkeley, 2008.

[131] H. Hotelling, *Analysis of a complex of statistical variables into principal components*, Journal of Educational Psychology, **24**(1933), 417-441 and 498-520.

[132] K.I. Howard and R.A. Gordon, *Empirical note on the `number of factors' problem in factor analysis*, Psychol. Rep., **12**(1963), 247-250.

[133] D.A. Jackson, *Stopping rules in principal component analysis: a comparison of heuristical and statistical approaches*, Ecology **74**(1993), 2204-2214.

[134] D. Jacobs, *Linear fitting with missing data: Applications to structure-from-motion and to characterizing intensity images*, IEEE Conf. on Computer Vision and Pattern Recognition, 1997, 206-212.

[135] D. Jacobs, *Linear fitting with missing data for structure-from-motion*, Computer Vision and Image Understanding, **82**(2001), 57-81.

[136] P. Jain, P. Netrapalli and S. Sanghavi, *Low-rank matrix completion using alternating minimization*, Tech. Report, arXiv:1212.0467v1, 2012.

[137] J.M. Jerez, I. Molina, J.L. Sabirats and L. Franco, *Missing data imputation in breast cancer prognosis*, Proceeding of the 24th IASTED International Multi-Conference on Biomedical Engineering, Innsbruck, Austria, 2006, 323-328.

[138] C.R. Johnson, *Matrix completion problems: A Survey*, Proceedings of Symposia in Applied Mathematics **40**(1990), 171-198.

[139] C.R. Johnson and P. Tarazaga, *Connections between the real positive semidefinite and distance matrix completion problems*, Linear Algebra Appl. **223/224**(1995), 375-391.

[140] C.R. Johnson and B.K. Kroschel, *The combinatorially symmetric P-matrix completion problem*, Electronic Journal of Linear Algebra **1**(1996), 59-63.

[141] C.R. Johnson and R.L. Smith, *The completion problem for M-matrices and inverse M-matrices*, Linear Algebra and its Applications, 241-243:655-667, 1996.

[142] L. Kaufman, *A variable projection method for solving separable nonlinear least squares problems*, BIT Numerical Mathematics **15**(1975), 49-57.

[143] A.M. Kalteh and P. Hjorth, *Imputation of missing values in a precipitation-runoff process database*, Hydrology Research **40**(2009), 420-432.

[144] R.H. Keshavan, A. Montanari and S. Oh, *Matrix completion from a few entries*, arXiv:0901.3150, January 2009.

[145] R.H. Keshavan and S. Oh, *OPTSPACE: A gradient descent algorithm on the Grassmann manifold for matrix completion*, Tech. Report, 2009. http://arxiv.org/abs/0910.5260v2.

[146] H.A.L. Kiers, *Weighted least squares fitting using ordinary least squares algorithms*, Psychometrika, **62**(1997), 251-266.

[147] H. Kim, G.H. Golub and H. Park, *Missing value estimation for DNA microarray gene expression data: local least squares imputation*, Bioinformatics **21**(2005), 187-198.

[148] D.W. Kim, K.Y. Lee, K.H. Lee and D. Lee, *Toward clustering of incomplete microarray data without the use of imputation*, Bioinformatics, **23**(2007), 107-113.

[149] R. Kohavi, *A study of cross-validation and bootstrap for accuracy estimation and model selection*, International Joint Conference on Artificial intelligence, San Francisco, USA: Morgan Kaufmann Publ. Inc., 1995, pp. 1137-1143.

[150] T.G. Kolda and B.W. Bader, *Tensor decompositions and applications*, SIAM Review **51**(2009), 455-500.

[151] Y. Koren, *Factorization meets the neighborhood: a multifaced collaborative filtering model*, Proc. of KDD'08, August 24-27, Las Vegas, Nevada, USA.

[152] Y. Koren, R. Bell and C. Volinsky, *Matrix factorization techniques for recommender systems*, Computer, published by the IEEE Computer Society, August 2009, pp. 42-49.

[153] R. Kosobud, *A note on a problem caused by assignment of missing data in sample surveys*, Econometrica, **31**(1963), 562-563.

[154] D. Kressner, M. Steinlechner and B. Vandereycken, *Low-rank tensor completion by Riemannian optimization*, BIT Numer. Math., **54**(2014), 447-468.

[155] R.J. Kuligowski and A.P. Barros, *Using artificial neural networks to estimate missing rainfall data*, Journal of the American Water Resources Association **34**(1998), 1437-1447.

[156] M. Kurucz, A.A. Benczúr and K. Csalogány, *Methods for large scale svd with missing values*, in KDD Cup and Workshop in conjunction with KDD 2007, 2007.

[157] K. Lakshminarayan, S.A. Harp, R. Goldman and T. Samad, *Imputation of missing data using machine learning techniques*, KDD-1996, 1966, 140-145.

[158] U. Lall and A. Sharma, *A nearest-neighbor bootstrap for resampling hydrologic time series*, Water Resource. Res., **32**(2001), 679-693.

[159] R.M. Larsen, *PROPACK - Software for large and sparse SVD calculations*, available from http://sun.stanford.edu/~rmunk/ PROPACK/.

[160] R.M. Larsen, *Lanczos bidiagonalization with partial reorthogonalization*, Technical Report DAIMI PB-357, Department of Computer Science, Aarhus University, 1998.

[161] M. Laurent, *The real positive semidefinite completion problem for series-parallel graphs*, Linear Algebra Appl. **252**(1997), 347-366.

[162] M. Laurent, *A connection between positive semidefinite and Euclidean distance matrix completion problems*, Linear Algebra and its Applicaitons **273**(1998), 9-22.

[163] M. Laurent and A. Varvitsiotis, *Positive semidefinite matrix completion, universal rigidity and the strong Arnold property*, arXiv:1301.6616v1 [math.oc], 2013.

[164] R.D. Ledesma and P. Valero-Mora, *Determining the Number of Factors to Retain in EFA: an easy-to-use computer program for carrying out Parallel Analysis*, Practical Assessment Research & Evaluation **12**(2007).

[165] K. Lee and Y. Bresler, *Efficient and guaranteed rank minimization by atomic decomposition*, preprint. Available at: arXiv:0901.1898v1, 2009.

[166] K. Lee and Y. Bressler, *ADMIRA: Atomic decomposition for minimum rank approximation*, IEEE Transactions on Information Theory **56**(2010), 4402-4416.

[167] S.-G. Lee and H.-G. Seol, *A Survey on the Matrix Completion Problem*, Trends in Mathematics, Information Center for Mathematical Sciences Vol. 4, June 2001, 38-43.

[168] L. Liberti, C. Lavor, N. Maculan, A. Mucherino, *Euclidean distance geometry and applications*, SIAM Review, **56**(2014), 3-69.

[169] Z. Lin, M. Chen, L. Wu and Y. Ma, *The augmented lagrange multiplier method for exact recovery of a corrupted low-rank matrices*, Mathematical Programming, submitted 2009.

[170] Z. Lin, A. Ganesh, J. Wright, L. Wu, M. Chen and Y. Ma, *Fast convex optimization algorithms for exact recovery of a corrupted low-rank matrix*, submitted to SIAM J. Optimization, (2009).

[171] G. Linden, B. Smith and J. York, Amazon.com recommendations: *Item-to-item collaborative filtering*, IEEE Internet Computing **7**(2003), pp. 76-80.

[172] R.J.A. Little, *Survey nonresponse adjustments*, Tech. Rep., University of California, Los Angeles.

[173] R.J.A. Little and D.B. Rubin, *Statistical Analysis with Missing Data*, John Wiley and Sons, 1987.

[174] J. Liu, P. Musialski, P. Wonka and J. Ye, *Tensor completion for estimating missing values in visual data*, in Proc. IEEE 12th International Conference on Computer Vision, pp. 2114-2121, 2009.

[175] Y. Liu and F. Shang, *An efficient matrix factorization method for tensor completion*, IEEE Signal Processing Letters **20**(2013), 307-310.

[176] Y. Liu, D. Sun and K.-C. Toh, *An implementable proximal point algorithmic framework for nuclear norm minimization*, preprint, National University of Singapore, 2009.

[177] Z. Liu and L. Vandenberghe, *Interior-point method for nuclear norm approximation with application to system identification*, SIAM Journal on Matrix Analysis and Applications **31**(2009), 1235-1256.

[178] F.M. Lord, *Estimation of parameters from incomplete data*, J. Amer. Statist. Ass., **50**(1955), 870-876.

[179] D.G. Luenberger, *Optimization by Vector Space Methods*, Wiley, New York, 1969.

[180] S. Ma, D. Goldfarb and L. Chen, *Fixed point and Bregman iterative methods for matrix rank minimization*, Technical report, Department of IEOR, Columbia University, 2008.

[181] Y. Ma, J. Wright, A. Ganesh, Z. Zhou, K. Min, S. Rao, Z. Lin, Y. Peng, M. Chen, L. Wu, E. Candés and X. Li, *Low-rank matrix recovery and completion via convex optimization*, Survey website 2013. http://perception.csl.illinois.edu/matrix-rank/.

[182] M. Magnani, *Techniques for dealing with missing data in knowledge discovery tasks*, Technical Report, University of Bologna, Department of Computer Science, 2004. Available from http://magnanim.web.cs.unibo.it/data/pdf/missingdata.pdf.

[183] A. Mahmoudabadi and S. Fakharian, *Estimating missing traffic data using artificial neural networks*, in *Proceedings of the International Conference on Intelligent Network and Computing (ICINC 2010), Kuala Lumpur, Malaysia*, November, 2010, Vol. 2, pp.144-147.

[184] M.A. Malek, S. Harun, S.M. Shamsudin and I. Mohamad, *Imputation of time series data via Kohonen self organizing maps in the presence of missing data*, World Academy of Science, Engineering and Technology **41**(2008), 501-506.

[185] J.H. Manton, R. Mahony and Y. Hua, *The geometry of weighted low-rank approximation*, IEEE Trans. Signal Process **51**(2003), 500-514.

[186] I. Markovsky, *Algorithms and literate programs for weighted low-rank approximation with missing data*. Technical Report 18296, ECS, Univ. of Southampton, http://eprints.ecs.soton.ac.uk/18296/,2009.

[187] A. Matthai, *Estimation of parameters from incomplete data with application to design of sample surveys*, Sankhya, **11**(1951), 145-152.

[188] R. Mazumder, T. Hastie and R. Tibshirani, *Regularization methods for learning incomplete matrices*, Technical Report, arXiv:0906.2034, June 2009.

[189] M.R. McLaughlin and J.L. Herlocker, *A collaborative filtering algorithm and evaluation metric that accurately model the user experience*, in *SIGIR '04: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2004.

[190] R. Meka, P. Jain and I.S. Dhillon, *Guaranteed rank minimization via singular value projection*, Tech. Rep., arXiv:0909.5457v3 [cs.LG] 19 Oct 2009.

[191] M. Michenkova, *Numerical algorithms for low-rank matrix completion problems*, Tech. Report, Dept. of Mathematics, Swiss Federal Institute of Technology. Zurich, Switzerland, 2011.

[192] B. Mishra, G. Meyer and R. Sepulchre, *Low-rank optimization for distance matrix completion*, in proceedings of the 50th IEEE Conference on Decision and Control, December 2011, pp. 4455-4460.

[193] B. Mishra, G. Meyer, S. Bonnabel and R. Sepulchre, *Fixed-rank matrix factorizations and Riemannian low-rank optimization*, 2013. arXiv:1209.0430v2.

[194] B. Mishra and R. Sepulchre, *R3MC: A Riemannian three-factor algorithm for low-rank matrix completion*, 2013. arXiv:1306.2672v1

[195] J. Mistry, F. V. Nelwamondo and T. Marwala, *Using principal component analysis and autoassociative neural networks to estimate missing data in a database*, in *Proceedings of the 12th World Multi-Conference on Systemics, Cybernetics and Informatics: WMSCI*.

[196] A.K. Mohamed, F. V. Nelwamondo and T. Marwala, *Estimation of missing data: Neural networks, principal component analysis and genetic algorithms*, in *Proceedings of the 12th World Multi-Conference on Systemics, Cybernetics and Informatics: WMSCI 2008*, June 29th -July 2nd, Orlando, Florida, USA, pp. 36-41.

[197] S. Mohamed and T. Marwala, *Neural network based techniques for estimating missing data in databases*, in *Proceedings of the 16th Annual Symposium of the Pattern Recognition Society of South Africa*, 2005, Langebaan, South Africa, pp. 27-32.

[198] K. Mohan and M. Fazel, *Iterative reweighted algorithms for matrix rank minimization*, Journal of Machine Learning Research **13**(2012), 3441-2473.

[199] M. Mullin and R. Sukthankar, *Complete Cross-Validation for nearest neighbor classifiers*, Proceedings of the Seventheenth International Conference on Machine Learning, pp. 639-646, 2000.

[200] B.K. Natarajan, *Sparse approximate solutions to linear systems*, SIAM Journal on Computing **24**(1995), 227-234.

[201] P.R.C. Nelson, P.A. Taylor and J.F. MacGregor, *Missing data methods in PCA and PLS: score calculations with incomplete observation*, Chemom. Intell. Lab. Syst., **35**(1996), 45-65.

[202] F. V. Nelwamondo, S. Mohamed and T. Marwala, *Missing Data: A Comparison of Neural Network and Expectation Maximisation Techniques*. Current Science 2007, 93(11), pp. 1507-1526.

[203] Netflix prize website. http://www.netflixprize.com/.

[204] D.V. Nguyen, N. Wang and R.J. Carroll, *Evaluation of missing value estimation for microarray data*, Journal of Data Science, **2**(2004), 347-370.

[205] A. Niknejad and S. Friedland, *Applications of Linear Algebra to DNA Microarrays*, Verlag Dr. Muller, 2009, to appear.

[206] L. Nirenberg, *Functional analysis*, Lectures given in 1960-1961, notes by Lesley Sibner, New York University, 1961.

[207] S. Oba, M. Sato, I. Takemasa, M. Monden, K. Matsubara and S. Ishii, *A Baesian missing value estimation method for gene expression profile data*, Bioinformatics, **19**(2003), 2088-2096.

[208] T. Okatani and K. Deguchi, *On the Wiberg algorithm for matrix factorization in the presence of missing components*, International Journal of Computer Vision, **72**(2006), 329-337.

[209] P. Ott, *Incremental matrix factorization for collaborative filtering*, Tech. Rep., Anhalt University of Applied Sciences, Germany, 2009.

[210] R. Parhizkar, A. Karbasi, S. Oh and M. Vetterli, *Calibration using matrix completion with application to ultrasound tomography*, IEEE Transactions on Signal Processing, July, 2013.

[211] B.N. Parlett, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, NJ, 1980.

[212] A. Paterek, *Improving Regularized Singular Value Decomposition for Collaborative Filtering*, Proc. KDD Cup and Workshop, ACM Press, 2007, pp. 39-42.

[213] D.V. Patil and R. S. Bichkar, *Multiple Imputation of Missing Data with Genetic Algorithm based Techniques*. IJCA Special Issue on *Evolutionary Computation for Optimization Techniques* ECOT, 2010, pp. 74-78.

[214] P.A. Patrician, *Multiple imputation for missing data*, Research in Nursing & Health **25**(2002), 76-84.

[215] P. R. Peres-Neto, D. A. Jackson and K.M. Somers, *How many principal components? stopping rules for determining the number of non-trivial axes revisited*, Computational Statistics & Data Analysis, **49**(2005), 974-997.

[216] C. Perou, et al., *Distinctive gene expression patterns in human mammary epithelial cells and breast cancers*, Proc. Natl. Acad. Sci. USA, **96**(1999), 9212-9217.

[217] E. Pesonen, et al., *Treatment of missing data values in a neural network based decision support system for acute abdominal pain*, Artificial Intelligence in Medicine, **13**(1998), 139-146.

[218] R.D. Priya, S. Kuppuswami and S.M.Kumar, *A Genetic Algorithm Approach for NonIgnorable Missing Data*, International Journal of Computer Applications **20**(2011), 37-41.

[219] Y.S. Qin, S.C. Zhang, X.F. Zhu, J.L. Zhang and C.Q. Zhang, *Semi-parametric Optimization for Missing Data Imputation*, Applied Intelligence, **27**(2007), 79-88.

[220] T. Raiko, A. Ilin and J. Karhunen, *Principal component analysis for large scale problems with lots of missing values*, in Procedings of the 18th European Conference on Machine Learning (ECML 2007), pages 691-698, Springer-Verlag, September 2007.

[221] T. Raiko and H. Valpola, *Missing values in nonlinear factor analysis*, in Proc. of the 8th Int. Conf. on Neural Information Processing (ICONIP/01), pages 822-827, Shanghai, 2001.

[222] M. Ramoni and P. Sebastiani, *Robust Learning with Missing Data*, Machine Learning **45**(2001), 147-170.

[223] H. Rauhut, R. Schneider and Z. Stojanac, *Low rank tensor recovery via iterative hard thresholding*, in Proc. of the 10th International Conference on Sampling Theory and Applications, 2013.

[224] B. Recht, *A Simpler Approach to Matrix Completion*, Journal of Machine Learning Research, to appear.

[225] B. Recht, M. Fazel and P. Parrilo, *Guaranteed minimum rank solutions of matrix equations via nuclear norm minimization*, SIAM Review **52**(2010), 471-501.

[226] B. Recht and C. Re, *Parallel stochastic gradients algorithms for large-scale matrix completion*, Tech. Report, Computer Sciences Department, University of Wisconsin-Madison, 2013.

[227] B. Recht, W. Xu and B. Hassibi, *Necessary and sufficient conditions for success of the nuclear norm heuristic for rank minimization*, California Institute of Technology, Tech. Rep., 2008, preprint available at http://arxiv.org/abs/0809.1260.

[228] B. Recht, W. Xu and B. Hassibi, *Null space conditions and thresholds for rank minimization*, Mathematical Programming **127**(2011).

[229] J.D.M. Rennie and N. Srebro, *Fast maximum margin matrix factorization for collaborative prediction*, in *Proceedings of the 22nd International Conference of Machine Learning*, 2005, pp. 713-719.

[230] D.B. Rubin, *Multiple imputation for nonresponse in surveys*, John Wiley and Sons, 1987.

[231] D.B. Rubin, *An overview of multiple imputation*, in *Survey Research Section*, American Statistical Association, 1988, pp. 79-84.

[232] A. Ruhe, *Numerical computation of principal components when several observations are missing*, Technical report, UMINF-48, Umea, 1974.

[233] A. Ruhe, *Accelerated Gauss-Newton algorithms for nonlinear least squares problems*, BIT, **19**(1979). 356-367.

[234] A. Ruhe and P.A. Wedin, *Algorithms for separable nonlinear least squares problems*, SIAM Review, **22**(1980), 318-337.

[235] B.M. Sarwar, G. Karypis, J.A. Konstan and J. Riedl, *Application of Dimensionality Reduction in Recommender System --- A Case Study*, WEBKDD '2000.

[236] J. Schafer, Analysis of incomplete multivariate data, Chapman Hall, 1997.

[237] C. Schaffer, *Selecting a classification method by cross-validation*, Machine Learning **13**(1993), 135-143.

[238] S. Schechter, *Iteration method for nonlinear problems*, Trans. Amer. Math. Soc., **104**(1962), 179-189.

[239] S. Schechter, *Relaxation methods for convex problems*, SIAM J. Numer. Anal., **5**(1968), 601-612.

[240] S. Schechter, *Minimization of a convex function by relaxation*, Integer and Nonlinear Programming, (J. Abadie, ed.), North-Holland, Amsterdam, 1970.

[241] J. Scheffer, *Dealing with missing data*, Res. Lett. Inf. Math. Sci.. **3**(2002), 153-160.

[242] M.S.B. Sehgal, I. Gondal and L.S. Dooley, *Collateral missing value imputation: A new robust missing value estimation algorithm for microarray data*, Bioinformatics **21**(2005), 2417-1423.

[243] U. Shalit, D. Weinshall and G. Chechik, *Online learning in the embedded manifold of low-rank matrices*, J. of Maching Learning Research **13**(2013), 429-458.

[244] J. Shao, *Linear model selection via cross-validation*, Journal of the American Statistical Association **88**(1993), 486-494.

[245] Y. Shen, Z. Wen and Y. Zhang, *Augmented Lagrangian Alternating Direction Method for Matrix Separation Based on Low-Rank Factorization*, Technical report CAAM TR11-02 (2011).

[246] H. Shum, K. Ikeuchi and R. Rebby, *Principal component analysis with missing data and its applications to polyhedral object modeling*, IEEE Transactions on Pattern Analysis and Machine Intelligence, **17**(1995), 854-867.

[247] M. Signoretto, R. Van de Plas, B. De Moor and J.A.K. Suykens, *Tensor versus matrix completion: A comparison with application to spectral data*, IEEE Signal Processing Letters **18**(2011), 403-406.

[248] L. Simonsson and L. Eldén, *Grassmann algorithms for low rank approxiation of matrices with missing values*, BIT Numerical Mathematics **50**(2010), 173-191.

[249] N. Srebro, *Learning with Matrix Factorization*, Ph.D thesis, Massachusetts Institute of Technology (2004).

[250] N. Srebro, J. Rennie and T. Jaakkola, *Maximum margin matrix factorization*, in *Advances in Neural Information Processing Systems* 17 (2005).

[251] N. Srebro and A. Shraibman, *Rank, trace-norm and max-norm*, Proceedings of the 18th Annual Conference on Learning Theory (COLT), June 2005.

[252] S.K. Starrett, S.K. Starrett, T. Heier, Y. Su, D. Tuan and M. Bandurraga, *Filling in missing peakflow data using artifical neural networks*, ARPN Journal of Engineering and Applied Sciences **5**(2010), 49-55.

[253] G.W. Stewart, *Matrix Algorithms, Vol. I: Basic Decompositions*, SIAM, Philadelphia, 1998.

[254] G.W. Stewart, *Matrix Algorithms, Vol. II: Eigensystems*, SIAM, Philadelphia, 2001.

[255] X. Su and T.M. Khoshgoftaar, *A survey of collaborative filtering techniques*, Advances in Artificial Intelligence, 2009.

[256] G. Takács, I. Pilászy, B. Németh and D. Tikk, *Matrix factorization and neighbor based algorithms for the Netflix Prize problem*, 2nd ACM Conf. on Recommendation Systems, pp. 267-274, Lausanne, Switzerland (2008).

[257] G. Takács, I. Pilászy, B. Németh and D. Tikk, *Scalable collaborative filtering approaches for large recommender systems*, Journal of Machine Learning Research **10**(2009), 623-656.

[258] M. Tao and X. Yuan, *Recovering low-rank and sparse components of matrices from incomplete and noisy observations*, SIAM Journal on Optimization **21**(2011), p. 57.

[259] A. Thalamuthu, I. Mukhopadhyay, X. Zheng and G.C. Tseng, *Evaluation and comparison of gene clustering methods in microarray analysis*, Bioinformatics, **22**(2006), 2405-2412.

[260] R. Tibshirani, *Regression shrinkage and selection via the lasso*, Journal of the Royal Statistical Society, Series B, **58**(1996), 267-288.

[261] K.-C. Toh and S. Yun, *An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems*, Pacific J. Optimization **6**(2010), pp. 615-640.

[262] G. Tomasi and R. Bro, *Parafac and missing values*, Chemometrics and Intelligent Laboratory Systems **75**(2005), 163-180.

[263] G. Tomasi and R. Bro, *A comparison of algorithms for fitting the PARAFAC model*, Comput. Statist. Data Anal. **50**(2006), 1700-1734.

[264] C. Tomasi and T. Kanade, *Shape and motion from image streams under orthography: a factorization method*, International Journal of Computer Vision **9**(1992), 137-154.

[265] R. Tomioka, K. Hayashi and H. Kashima, *Estimation of low-rank tensors via convex optimization*, Tech. Report, 2011.

[266] W.S. Torgerson, *Multidimensional scaling: I. Theory and method*, Psychometrika **17**(1952), 401-419.

[267] M.W. Trosset, *Applications of multidimensional scaling to molecular conformation*, Computing Science and Statistics **29**(1998), 148-152.

[268] M.W. Trosset, *Distance matrix completion by numerical optimization*, Computational Optimization and Applications **17**(2000), 11-22.

[269] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein and R. Altman, *Missing value estimation for DNA microarrays*, Bioinformatics, **17**(2001), 520-525.

[270] N. Tsikriktsis, *A review of techniqes for treating missing data in OM survey research*, Journal of Operations Management **24**(2005), 53-62.

[271] F. Vandenberghe and S. Boyd, *Semidefinite Programming*, SIAM Review, **38**(1996), 49-95.

[272] B. Vandereycken, *Low-rank matrix completion by Riemannian optimization*, SIAM J. on Optimization **23**(2013), 1214-1236.

[273] G. Vanwinckelen and H. Blockeel, *On estimating model accuracy with repeated Cross-Validation*, Proceedings of BeneLearn and PMLS, 2012.

[274] B. Walczak and D.L. Massart, *Dealing with missing data: Part I.* Chemometr. Intell. Lab. **58**(2011), 15-27.

[275] X. Wang, A. Li, Z. Jiang, and H. Feng, *Missing value estimation for DNA microarray gene expression data by Support Vector Regression imputation and orthogonal coding scheme*, BMC Bioniformatics, **7**(2006).

[276] I. Wasito and B. Mirkin, *Nearest neighbour approach in the least-squares data imputation algorithms*, Inform. Sci., **169**(2005)

[277] Z. Wen, W. Yin and Y. Zhang, *Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm*, Math. Prog. Comp. **4**(2012), 333-361.

[278] T. Wiberg, *Computation of principal components when data are missing*, in: Proc. of the Second Symposium Computational Statistics, Berlin, Germany, 1976.

[279] G.N. Wilkinson, *Estimation of missing values for the analysis of incomplete date*, Biometrics, **14**(1958), 257-286.

[280] J.H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1965.

[281] M.F. Villazon and P. Willems, *Filling gaps and daily disaccumulation of precipitation data for rainfall-runoff model*, BALWOIS 2010 - Ochrid, Republic of Macedonia, 25-29 May, 2010.

[282] H. Wold, *Estimation of principal components and related models by iterative least square*, in: P.R. Krishnaiah (ed.), Multivariate Analysis Proceedings of International Symposium in Dayton, Academic Press, New York, 1955, 391-402.

[283] S. Wold, A. Ruhe, H. Wold and W.J. Dunn, *The collinearity problem in linear regression. The partial least squares (PLS) approach to generalized inverses*, SIAM J. Sci. Stat. Comput., **5**(1984), 735-743.

[284] S. Wold, M. Sjöström and L. Eriksson, *PLS-regression: a basic tool of chemometrics*, Chemometrics and Intell. Lab. Systems, **58**(2001), 109-130.

[285] J. Wright, A. Ganesh, S. Rao, Y. Peng and Y. Ma, *Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization*, submitted to Journal of the ACM, 2009.

[286] D. Yoon, E. Lee and T. Park, *Robust imputation method for missing values in microarray data*, BMC Bioinformatics, 2007, 8 (Suppl. 2): S6.

[287] D. Zacharih, M. Sundin, M. Jansson and S. Chatterjee, *Alternating least-squares for low-rank matrix reconstruction*, Tech. Report, ACCESS Linnaeus Center, KTH Royal Institute of Technology, Stockholm.

[288] C.Q. Zhang, et al., *An Imputation Method for Missing Values*, PAKDD, LNAI 4426, 2007, 1080-1087.

[289] S. Zhang, W. Wang, J. Ford, F. Makedon and J. Pearlman, *Using singular value decomposition approximation for collaborative filtering*, in CEC'05: Proceedings of the Seventh IEEE International Conference on E-Commerce Technology (CEC'05), pages 257-264, Washington, DC, USA, 2005. IEEE Computer Society.

[290] S.C. Zhang, C. Zhang and Q. Yang, *Information Enhancement for Data Mining*, IEEE Intelligent Systems, **19**(2004), 12-13.

[291] S.C. Zhang, Z.X. Quin, C.X. Ling and S.L. Sheng, *Missing is useful: Missing values in cost-sensitive decision trees*, IEEE Transactions on Knowledge and Data Engineering, **12**(2005), 1689-1693.

[292] S. Zhang, J. Zhang, X. Zhu, Y. Qin and C. Zhang, *Missing value imputation based on data clustering*, in: Transactions on Computational Science, LNCS Vol. 4750, pp. 128-138, Springer, 2008.

[293] H.-G. Zhao, *A heuristic method for computing the best rank-r approximation to higher-order tensors*, Int. J. Contemp. Math. Sciences **3**(2008), 471-476.

[294] M. Zhong, P. Lingras and S. Sharma, *Estimation of Missing Traffic Counts Using Factor, Genetic, Neural, and Regression Techniques*, Transportation Research Part C: Emerging Technologies **12**(2004), 139-166.

[295] M. Zhong, S. Sharma and P. Lingras, *Genetically Designed Models for Accurate Imputations of Missing Traffic Counts*, Transportation Research Records 1879, Journal of Transportation Research Board, 2004, pp. 71-79.

[296] Z. Zhu, A. M.-C. So and Y. Ye, *Fast and Near-Optimal Matrix Completion via Randomized Basis Pursuit*, Technical Report, Stanford University, May 15, 2009.

[297] R.W. Zwick and W.F. Velicer, *Comparison of five rules for determining the number of components to retain*, Psych. Bull., **99**(1986), 432-442.