Application of Singular Value Decomposition to DNA Microarray

Amir Niknejad

M.S. Claremont Graduate School, 1997

M.S. Northeastern University, 1983

Thesis

Submitted in partial fulfilment of the requirements

for the degree of Doctor of Philosophy

in the Graduate College of the

University of Illinois at Chicago

Chicago, Illinois

August, 2005

Dedication: Dedicated to Dr. Iraj Broomand, Founder of the National Iranian Organization for Gifted and Talented Education, and to the memory of my father.

Acknowledgement: I am grateful to my advisor Professor Shmuel Friedland for his kind supervision, patience and support. I would like to thank my committee members Professor Calixto Calderon, Professor Louis Kauffman, Professor Stanley Sclove and Professor Jan Verschelde, for kindly agreeing to be on my committee despite their busy schedules. I would like to thank Kari Dueball and Darlette Willis for their assistance during my stay at UIC, the UIC mathematics department and the Institute of Mathematics and its Applications (IMA) for their generous support and fellowship, Ali Shaker and Marcus Bishop for kindly helping me with LaTeX, and Dr. Laura Chihara and Hossein Zare for coding and simulation of FRAA and IFRAA. Finally I would like to acknowledge the generous support of Hossein Andikfar, Hossein Basiri, Vali Siadat and Mahyad Zaerpour.

Summary

In chapter 1 we introduce the singular value decomposition (SVD) of matrices and its extensions. We then mention some applications of SVD in analyzing gene expression data, image processing and information retrieval. We also introduce the low rank approximation of matrices and present our Monte Carlo algorithm to achieve this approximation along with other algorithms.

Chapter 2 deals with clustering methods and their application in analysing gene expression data. We introduce the most common clustering methods such as the KNN, SVD, and weighted least square methods.

Chapter 3 deals with imputing missing data in gene expression of micro-arrays. We use some of the methods mentioned in part 2 for these purposes. Then we introduce our fixed rank approximation algorithm (FRAA) for imputing missing data in the DNA gene expression array. Finally, we use simulation to compare FRAA versus other methods and indicate the advantages and its shortcomings, and how to overcome the shortcomings of FRAA.

Contents

1 Introduction and Background

1.1 Missing gene imputation in DNA microarrays

In the last decade, molecular biologists have been using DNA microarrays as a tool for analyzing information in gene expression data. During the laboratory process, some spots on the array may be missing due to various factors (for example, machine error.) Because it is often very costly or time consuming to repeat the experiment, molecular biologists, statisticians, and computer scientists have made attempts to recover the missing gene expressions by some ad-hoc and systematic methods.

In this thesis we introduce various imputation methods for missing data in gene expression matrices. In particular, we review two recent imputation methods, namely Bayesian Principal Component Analysis (BPCA) and Local Least Square Impute (LLSImpute). Then we introduce our fixed rank approximation algorithm (FRAA). Finally in the latter section of this thesis we present improved fixed rank approximation (IFRAA) and use simulation in order to compare IFRAA with BPCA nd LLSimpute.

More recently, microarray gene expression data have been formulated as a gene expression matrix E with n rows, which correspond to genes, and m columns, which correspond to experiments. Typically n is much larger than m. In this setting, the analysis of missing gene expressions on the array would translate to recovering missing entries in the gene expression matrix values.

The most common methods for recovery are [35]:

(a) Zero replacement method;

(b) Row sum mean;

(c) Cluster analysis methods such as K-nearest neighbor clustering , hierarchical clustering;

(d) SVD - Singular Value Decomposition (which is closely related to Principal Component Analysis).

In these methods, the recovery of missing data is done independently, i.e., the estimation of each missing entry does not influence the estimation of the other missing entries. The iterative method using SVD suggested in [35] takes into account implicitly the influence of the estimation of one entry on the other ones. See also [9].

In this thesis we suggest a new method in which the estimation of missing entries is done simultaneously, i.e., the estimation of one missing entry influences the estimation of the other missing entries. If the gene expression matrix E has missing data, we want to complete its entries to obtain a matrix $\hat{E}$, such that the rank of $\hat{E}$ is equal to (or does not exceed) d, where d is taken to be the number of significant singular values of E. The estimation of the entries of E to a matrix with a prescribed rank is a variation of the *problem of communality* (*see* [16, p. 637].) We give an optimization algorithm for finding $\hat{E}$ using the techniques for inverse eigenvalue problems discussed in [14].

1.2 Some biological background

1.2.1 Genes and Gene Expression

Cells and organisms are divided into two classes; procaryotic (such as bacteria) and eucaryotic [10]. The latter have a nucleus. The cell is enclosed by its membrane; embedded in the cell's cytoplasm is its nucleus, surrounded and protected by its own membrane. The nucleus contains DNA, a one dimensional molecule, made of two complementary strands, coiled around each other as a double helix. Each strand consists of a backbone to which a linear sequence of bases is attached. There are four kinds of bases, denoted by C, G, A, T. The two strands contain complementary base sequences and are held together by hydrogen bonds that connect matching pairs of bases; G-C (three hydrogen bonds that connect matching pairs of bases; G-C (three hydrogen bonds) and A-T (two).

A *gene* is a segment of DNA, which contains the formula for the chemical composition of one particular protein. Proteins are the working molecules of life; most biological processes that take place in a cell are carried out by proteins. Topologically, a protein is also a chain; each link is an amino acid, with neighbors along the chain connected by covalent bonds. All proteins are made of 20 different amino acids - hence the chemical formula of a protein of length N is an N-letter word, whose letters are taken from a 20-letter alphabet. A gene is nothing but an alphabetic cookbook recipe, listing the order in which the amino acids are to be strung when the corresponding protein is synthesized. Genetic information is encoded in the linear sequence in which the bases on the two strands are ordered along the DNA

molecule. The genetic code is a universal translation table, with specific triplets of consecutive bases coding for every amino acid.

The *genome* contains the collection of all the genes that code the chemical formulae of all the proteins (and RNA) that an organism needs and produces. The genome of a simple organism such as yeast contains about 6000 genes; the human genome has between 30,000 and 40,000. An overwhelming majority (98%) of human DNA contains non-coding regions , i.e., strands that do not code for any particular protein (but play a role in regulating the level of synthesis of the different proteins).

Here is an amazing fact: every cell of a multicellular organism contains its entire genome! That is, every cell has the entire set of recipes the organism may ever need; the nucleus of each of the reader's cell contains every piece of information needed to make a copy ("clone") of him/her! Even though each cell contains the same set of genes, there is *differentiation*: cells of a complex organism, taken from different organs, have entirely different functions and the proteins that perform these functions are very different. Cells in our retina need photosensitive molecules, whereas our livers do not make much use of these. A gene is expressed in a cell when the protein it codes for is actually synthesized. In an average human cell about 10,000 genes are expressed. The set of (say 10,000) numbers that indicate the expression level of each of theses genes is called the *expression profile* of the cell.

The large majority of abundantly expressed genes are associated with common functions, such as metabolism, and hence are expressed in all cells. However there will be differences between the expression of profiles of different cells, and even

in single cell, expression will vary with time, in a manner dictated by external and internal signals that reflect the state of the organism and the cell itself.

Synthesis of proteins take a place at the *ribosomes*. These are enormous machines (also made of proteins) that read the chemical formulae written on the DNA and synthesize the protein according to the instructions. The ribosomes are in the cytoplasm, whereas the DNA is in the protected environment of the nucleus. This poses an immediate logistic problem - how does the information get transferred from the nucleus to the ribosome?

1.2.2 Transcription and Translation

The obvious solution of information transfer would be to rip out the piece of DNA that contains the gene that is to be expressed, and transport it to the cytoplasm. When a gene receives a command to be expressed, the corresponding segment of the double helix of DNA opens, and a precise copy of the information, as written on one of the strands, is prepared, which is called the *messenger* RNA, and the process of its production is called *transcription*. The subsequent reading of mRNA, deciphering the message (written using base triplets) into amino acids and synthesis of the corresponding protein at the ribosomes is called *translation* [32]. In fact, when many molecules of a certain protein are needed, the cell produces many corresponding mRNAs, which are transferred through the nucleus' membrane to the cytoplasm, and are "read" by several ribosomes. Thus the single master copy of the instructions, contained in the DNA, generates many copies of the protein. This transcription strategy is prudent and safe, preserving the precious master copy; at

the same time it also serves as a remarkable amplifier of the genetic information.

1.2.3 DNA microarrays (chips)

A DNA Chip is the instrument that measures simultaneously the concentration of thousands of different mRNA molecules. It is also referred to as a DNA microarray [32] DNA microarrays, produce by Affymetrix, can measure simultaneously the expression levels of up to 20,000 genes; the less expensive spotted arrays do the same for several thousand. Schematically, the Affymetrix arrays are produced as follows. Divide a chip (a glass plate of about 1 cm across) into ”pixels” - each dedicated to one gene g. Millions of 25 base-pair long pieces (oligonucleotides) of *single strand DNA*, copied from a particular segment of gene g are are photolitigraphically synthesized on the dedicated pixel (these are referred to as ”probes”). The mRNA molecules are extracted from the cells taken from the tissue of interest (such as a tumor tissue) obtained by concentration is enhanced. Next, the resulting DNA is transcribed back into fluorescently marked single strand RNA diffuse over the dense forest of single strand DNA and the labeled RNA diffuse over the dense forest of single strand DNA and the labeled RNA diffuse over the dense forest of single strand DNA probes. When such an mRNA encounters a bit of the probe, of which the RNA is a perfect copy, it hybridizes to this strand - i.e. attaches to it with a high affinity (considerably higher than to a probe of which the target is not a perfect copy). When the mRNA solution is washed off, only those molecules that found their perfect match remain stuck to the chip. Now the chip is illuminated by a laser, and these stuck ”targets” fluoresce; by measuring the light intensity ema-

nating from each pixel, one obtains a measure of the number of targets that stuck, which form a chip on which N_g genes were placed. These N_g numbers represent the expression levels of these genes in that tissue. A typical experiment provides the expression profiles of several tens of samples (say $N_s \approx 100$), over several thousand (N_g) genes. These results are summarized in an $N_g \times N_s$ *expression table*; each row corresponds to one particular gene and each column to a sample. Entry E_{gs} of such an expression table stands for the expression level of gene g in sample s. For example, the experiment on colon cancer, first reported by Alon et al. [3] contains $N_g = 2000$ genes whose expression levels passed some threshold, over $N_s = 62$ samples, 40 of which were taken from tumor and 22 from normal colon tissue.

1.3 Some linear algebra background

Since DNA microarray data involves real values only, we will be restricting our treatment of relevant facts from linear algebra to matrices and vector spaces over $\mathbb{R}$.

1.3.1 Inner Product Spaces (IPS)

Let V be a vector space over $\mathbb{R}$. Then $\langle \bullet, \bullet \rangle : V \times V \to \mathbb{R}$ is a real inner product if

1. $\langle \mathbf{x}, \mathbf{x} \rangle \geq 0$ for all $\mathbf{x} \in V$ and $\langle \mathbf{x}, \mathbf{x} \rangle = 0$ iff $\mathbf{x} = 0$.

2. $\langle \mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{y}, \mathbf{x} \rangle$ for all $\mathbf{x}, \mathbf{y} \in V$.

3. $\langle \mathbf{x}, \alpha\mathbf{y}_1 + \beta\mathbf{y}_2 \rangle = \alpha\langle \mathbf{x}, \mathbf{y}_1 \rangle + \beta\langle \mathbf{x}, \mathbf{y}_2 \rangle$ for all $\mathbf{x}, \mathbf{y}_1, \mathbf{y}_2 \in V$ and for all $\alpha, \beta \in \mathbb{R}$.

1.3.2 Definition and Properties of Positive Definite and Semidefinite Matrices

In the following $\mathbf{x}$ will denote a vector in $\mathbb{R}^n$. A symmetric matrix $A \in \mathbb{R}^{n \times n}$ is

1. **positive definite** if and only if $\mathbf{x}^T A \mathbf{x} > 0$ for all nonzero $\mathbf{x}$. We write $A > 0$.

2. **nonnegative definite** (or **positive semi-definite**) if and only if $\mathbf{x}^T A \mathbf{x} \geq 0$ for all $\mathbf{x}$. We write $A \geq 0$.

3. **negative definite** if $-A$ is positive definite.

4. **nonpositive definite** (or **negative semidefinite**) if $-A$ is nonnegative definite. We write $A \leq 0$.

We have the following facts.

- Any principal submatrix of a positive (nonnegative) definite matrix is positive (nonnegative) definite . ($A \geq 0$ implies any principal submatrix is n.n.d)

- The diagonal elements of a positive definite matrix are positive.

- A is positive definite if and only if its eigenvalues are positive. (It is positive semidefinite if its eigenvalues are nonnegative.)

- If A is positive definite, its determinant is positive. If A is positive semidefinite, its determinant is nonnegative.

- If A is positive definite, then A is nonsingular and A^{-1} is positive definite.

1.3.3 Various Matrix Factorizations

In the following section, we introduce several matrix factorizations useful throughout this thesis.

Theorem 1.1 (Spectral Decomposition) *Let $A = A^T \in \mathbb{R}^{n\times n}$. Then there exists an orthogonal matrix $Q \in \mathbb{R}^{n\times n}$ (whose columns are orthogonal eigenvectors of A) such that*

$$A = QDQ^T = \sum_{i=1}^{n} \lambda_i \mathbf{q}_i \mathbf{q}_i^T$$

where λ_i are the eigenvalues of A and $D = diag(\lambda_1, \ldots \lambda_n)$.

1.3.4 Gram-Schmidt Process

The Gram-Schmidt process is a recursive procedure for generating an orthonormal set of vectors from a given set of linearly independent vectors. Let $\mathbf{x}_1, \ldots \mathbf{x}_n$ be linearly independent vectors in an IPS V. Then generate an orthonormal set $\mathbf{u}_1, \ldots \mathbf{u}_n \in V$ from $\mathbf{x}_1, \ldots \mathbf{x}_n$ such that:

$$\operatorname{span}(\mathbf{x}_1, \ldots, \mathbf{x}_k) = \operatorname{span}(\mathbf{u}_1, \ldots, \mathbf{u}_k), \quad k = 1, \ldots, n,$$

by the following procedure known as the Gram-Schmidt algorithm:

- $\mathbf{u}_1 := \frac{1}{\|\mathbf{x}_1\|}\mathbf{x}_1$
- $\mathbf{p}_1 := \langle \mathbf{x}_2, \mathbf{u}_1 \rangle \mathbf{u}_1, \quad \mathbf{u}_2 := \frac{1}{\|\mathbf{x}_2 - \mathbf{p}_1\|}(\mathbf{x}_2 - \mathbf{p}_1)$
- $\mathbf{p}_2 := \langle \mathbf{x}_3, \mathbf{u}_1 \rangle \mathbf{u}_1 + \langle \mathbf{x}_3, \mathbf{u}_2 \rangle \mathbf{u}_2, \quad \mathbf{u}_3 := \frac{1}{\|\mathbf{x}_3 - \mathbf{p}_2\|}(\mathbf{x}_3 - \mathbf{p}_2)$

$\vdots$

- $\mathbf{p}_k := \langle x_{k+1}, \mathbf{u}_1 \rangle \mathbf{u}_1 + \langle \mathbf{x}_{k+1}, \mathbf{u}_2 \rangle \mathbf{u}_2 + \cdots + \langle \mathbf{x}_{k+1}, \mathbf{u}_k \rangle \mathbf{u}_k$,

  and $\mathbf{u}_{k+1} := \frac{1}{\|\mathbf{x}_{k+1} - p_k\|}(x_{k+1} - \mathbf{p}_k)$

In practise the Gram-Schmidt Process (GSP) is numerically unstable. That is, there is a severe loss of orthogonality of $\mathbf{u}_1, \ldots$ as we proceed to compute $\mathbf{u}_i$. In computations one uses either a Modified GSP or Householder orthogonalization [21].

The Modified Gram-Schmidt Process (MGSP) consisting of the following steps:

- Initialize $j = 1$.
- $\mathbf{u}_j := \frac{1}{\|\mathbf{x}_j\|}\mathbf{x}_j$
- $\mathbf{p}_i := \langle \mathbf{x}_i, \mathbf{u}_j \rangle \mathbf{u}_i$ and replace $\mathbf{x}_i$ by $\mathbf{x}_i := \mathbf{x}_i - \mathbf{p}_i$ for $i = j + 1, \ldots, n$.
- Let $j = j + 1$ and repeat the process.

MGSP is stable, needs mn^2 flops, which is more time consuming then GSP.

1.3.5 Cholesky Decomposition

Let A be a positive definite matrix. Then $A = U^T \cdot U$ where U is an upper triangular matrix with positive diagonal elements. The above factorization is called the *Cholesky decomposition*. The matrix U is called a *Cholesky factor* of A and is not unique.

1.3.6 The QR Decomposition

Let X be an $n \times m$ matrix with $n \geq m$. Then there is a unitary $n \times m$ matrix Q and an upper triangular $n \times n$ matrix R with nonnegative diagonal elements such that

$$X^{n \times m} = Q \begin{pmatrix} R \\ \\ 0 \end{pmatrix}.$$

This factorization is called the QR *decomposition* of X. The matrix Q is called the Q*-factor* of X and the matrix R is the R*-factor* of X.

If we partition $Q = (Q_X, Q_L)$ where Q_X has m columns, then $X = Q_X R$ which is called the QR *factorization* of X.

If X is of rank m, then the QR factorization is unique up to a factor of ± 1 . The columns of Q_X form an orthogonal basis for the column space of X, and those of $Q_\perp$ form an orthonormal basis for the orthogonal complement of the column space of X obtained by the Gram-Schmidt process. If X is of rank m, then $A = X^T X$ is positive definite and R is a Cholesky factor of A. For more details see [34].

1.3.7 An Introduction to the Singular Value Decomposition

In many science and engineering problems, one would like to approximate a given matrix by a lower rank matrix and calculate the distance between them.

Without loss of generality, let E be a $n \times m$ matrix where $n \geq m$ (The following procedure is also valid when $n < m$). Denote by $\mathcal{R}(E)$ and $\mathcal{N}(E)$ the range and the null space of E.

The idea of the *singular value decomposition* is to factor the matrix E into the

product of matrices U, Σ, V such that

$$E^{(n\times m)} = U^{(n\times n)}\Sigma^{(n\times m)}V^{(m\times m)T}. \tag{1.1}$$

Here, U is $n \times n$, Σ is a diagonal $n \times m$ matrix, V is a $m \times m$ matrix, and U and V are orthogonal matrices where

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\min(m,n)} \geq 0.$$

In this procedure, the σ_i's are unique but U and V are not. The σ_i's are called the *singular values* of E and are the square roots of the eigenvalues of EE^T or equivalently E^TE. Also, in this setting the columns of U are eigenvectors of EE^T and the rows of V are eigenvectors of E^TE. The idea is that the rank of the matrix E is the number of the nonzero singular values and the magnitude of nonzero singular values would indicate the proximity of matrix E to the closest lower rank matrix.

Computationally, one brings E to upper bidiagonal form A using Householder matrices. Then one applies implicitly the QR algorithm to $A^{\mathrm{T}}A$ to find the positive eigenvalues $\sigma_1^2, ..., \sigma_r^2$ and the corresponding orthonormal eigenvectors $\mathbf{v}_1, ..., \mathbf{v}_r$ of the matrix $E^{\mathrm{T}}E$ [21]. Next we set

$$\mathbf{u}_q := \frac{1}{\sigma_q}E\mathbf{v}_q \tag{1.2}$$

Theorem 1.2 *If E is an $m \times n$ matrix, then E has a singular value decomposition.*

The proof can be found in [21].

We summarize the facts about the singular value decomposition of a matrix. Let E be a $m \times n$ matrix with singular value decomposition $U\Sigma V^T$.

1. The singular values $\sigma_1, \sigma_2, \ldots, \sigma_n$ of E are unique, however the matrices U and V are not unique. Suppose

$$\sigma_1 > \sigma_2 > \cdots > \sigma_r > 0,$$

then the columns of V are determined up to ± 1.

2. Since V diagonalizes $E^T E$, it follows that $\mathbf{v}_j$'s are eigenvectors of $E^T E$.

3. Since $EE^T = U\Sigma\Sigma^T U^T$, it follows that U diagonalizes EE^T and that the $\mathbf{u}_j$'s are the eigenvectors of EE^T.

4. Comparing the jth columns of each side of the equation

$$EV = U\Sigma$$

we get

$$E\mathbf{v}_j = \sigma_j \mathbf{u}_j \qquad j = 1, \ldots, n.$$

Similarly,

$$E^T U = V\Sigma^T$$

and hence

$$E^T \mathbf{u}_j = \sigma_j \mathbf{v}_j \qquad j = 1, \ldots, n$$

$$E^T \mathbf{u}_j = 0 \qquad j = n+1, \ldots, m.$$

The $\mathbf{v}_j$'s are called the *right singular vectors* of E and the $\mathbf{u}_j$'s are called the *left singular vectors* of E.

5. If E has rank r, then

   (a) $\mathbf{v}_1, \mathbf{v}_2, \ldots \mathbf{v}_r$ form an orthonormal basis for $\mathcal{R}\left(E^T\right)$,

   (b) $\mathbf{v}_{r+1}, \mathbf{v}_{r+2}, \ldots, \mathbf{v}_n$ form an orthonormal basis for $\mathcal{N}(E)$,

   (c) $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_r$ form an orthonormal basis for $\mathcal{R}(E)$,

   (d) $\mathbf{u}_{r+1}, \mathbf{u}_{r+2}, \ldots, \mathbf{u}_m$ form an orthonormal basis for $\mathcal{N}\left(E^T\right)$.

6. The rank of E is equal to the number of its nonzero singular values (where singular values are counted according to their multiplicities). We can rearrange the σ_i's such that

$$\sigma_1 = \cdots = \sigma_{i_1} > \sigma_{i_1} = \cdots = \sigma_{i_1+l_2} > \cdots > \sigma_{i_p} = \cdots = \sigma_{i_p+l_p} = \sigma_r$$

   and for every $1 \leq j \leq p-1$, $\sigma_{i_j+l_j+1} = \sigma_{i_j+1}$ and $\mathbf{v}_{i_j}, \ldots, \mathbf{v}_{i_j+l_j}$ form an orthonormal basis for the eigenspace of $E^T E$ corresponding to σ_{i_j}.

Let U_r, Σ_r, V_r be matrices obtained from U, Σ, V, respectively, as follows: U_r is an $n \times r$ matrix obtained by deleting the last $m-r$ columns of U, V_r is the $m \times r$ matrix obtained by deleting the last $m-r$ columns of V, and Σ_r is obtained by deleting the last $m-r$ columns and rows of Σ . Then

$$E = U_r \Sigma_r V_r^{\mathrm{T}},\ U_r^{\mathrm{T}} U_r = V_r^{\mathrm{T}} V_r = I_r,\ \Sigma_r = \mathrm{diag}(\sigma_1, ..., \sigma_r),\ \sigma_1 \geq ... \geq \sigma_r > 0. \tag{1.3}$$

In this setting U_r, Σ_r, V_r are all rank r matrices: the last $m-r$ columns of U and the last $m-r$ rows of V^{T} are arbitrary, up to the condition that the last $m-r$ columns of U and last $m-r$ rows of V^{T} are orthonormal bases of the orthogonal

complement of the column space and the row space of E, respectively. Hence (1.3) is sometimes called a *reduced* SVD of E.

1.3.8 SVD on inner product spaces

In this section we discuss briefly the standard notion of the SVD decomposition of a linear operator that maps one finite dimensional inner product space to another finite dimensional inner product space. For the utmost generality we consider here the inner products over the complex numbers $\mathbb{C}$. In this section we denote by $\mathrm{M}_{nm}(\mathbb{C})$ the space of $n \times m$ matrices with complex entries.

Let $\mathbf{U}_i$ be an m_i-dimensional inner product space over $\mathbb{C}$, equipped with $\langle \cdot, \cdot \rangle_i$ for $i = 1, 2$. Let $T : \mathbf{U}_1 \to \mathbf{U}_2$ be a linear operator. Let $T^* : \mathbf{U}_2 \to \mathbf{U}_1$ be the adjoint operator of T, i.e. $\langle T\mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{x}, T^*\mathbf{y} \rangle$ for all $\mathbf{x} \in \mathbf{U}_1, \mathbf{y} \in \mathbf{U}_2$. Equivalently, let $[\mathbf{a}_1, ..., \mathbf{a}_{m_1}]$ and $[\mathbf{b}_1, ..., \mathbf{b}_{m_2}]$ be orthonormal bases of $\mathbf{U}_1$ and $\mathbf{U}_2$ respectively. Let $A \in \mathbf{M}_{m_2 m_1}(\mathbb{C})$ be the representation matrix of T in these bases: $[T\mathbf{a}_1, ..., T\mathbf{a}_{m_1}] = [\mathbf{b}_1, ..., \mathbf{b}_{m_2}]A$. Then the matrix $A^* := \overline{A}^{\mathrm{T}}$ represents T^* in these bases. The SVD decomposition of $A = U\Sigma V^*$, where U, V are unitary matrices of dimensions m_2, m_1 respectively, and $\Sigma \in \mathrm{M}_{m_2 m_1}(\mathbb{R})$ is a diagonal matrix with the diagonal entries $\sigma_1 \geq ... \sigma_{\min(m_2, m_1)} \geq 0$, corresponds to the following *base free* concepts of T.

Consider the operators $S_1 := T^*T : \mathbf{U}_1 \to \mathbf{U}_1$ and $S_2 := TT^* : \mathbf{U}_2 \to \mathbf{U}_2$. Then S_1, S_2 are self-adjoint, i.e. $S_1^* = S_1, S_2^* = S_2$ and nonnegative definite: $\langle S_i \mathbf{x}_i, \mathbf{x}_i \rangle \geq 0$ for all $\mathbf{x}_i \in \mathbf{U}_i$ for $i = 1, 2$. The positive eigenvalues of S_1 and S_2, counted with their multiplicities and arranged in a decreasing order are $\sigma_1^2 \geq ... \geq$

$\sigma_r^2 > 0$, where $r = \mathbf{rank}T = \mathbf{rank}T^*$. Let

$$S_1\mathbf{v}_i = \sigma_i^2\mathbf{v}_i,\ i = 1, ..., r, \quad \langle \mathbf{v}_i, \mathbf{v}_j \rangle_1 = \delta_{ij},\ i, 1, ..., r.$$

Then $\mathbf{u}_i := \sigma_i^{-1}T\mathbf{v}_i$ for $i = 1, ..., r$ is an orthonormal set of the eigenvectors of S_2 corresponding to the eigenvalue σ_i^2 for $i = 1, ..., r$. Complete the orthonormal systems $\{\mathbf{v}_1, ..., \mathbf{v}_r\}$ and $\{\mathbf{u}_1, ..., \mathbf{u}_r\}$ to orthonormal bases $[\mathbf{v}_1, ..., \mathbf{v}_{m_1}]$ and $[\mathbf{u}_1, ..., \mathbf{u}_{m_2}]$ in $\mathbf{U}_1$ and $\mathbf{U}_2$ respectively. Then the unitary matrices U, V are the transition matrices from basis $[\mathbf{b}_1, ..., \mathbf{b}_{m_2}]$ to $[\mathbf{u}_1, ..., \mathbf{u}_{m_2}]$ and basis $[\mathbf{a}_1, ...\mathbf{a}_{m_1}]$ to $[\mathbf{v}_1, ..., \mathbf{v}_{m_1}]$:

$$[\mathbf{u}_1, ..., \mathbf{u}_{m_2}] = [\mathbf{b}_1, ..., \mathbf{b}_{m_2}]U, \quad [\mathbf{v}_1, ..., \mathbf{v}_{m_1}] = [\mathbf{a}_1, ..., \mathbf{a}_{m_1}]V.$$

Let $A \in \mathrm{M}_{m_2 m_1}(\mathbb{C})$. Then A can be viewed as a linear operator $A : \mathbb{C}^{m_1} \to \mathbb{C}^{m_2}$, where $\mathbf{x} \to A\mathbf{x}$ for any $\mathbf{x} \in \mathbb{C}^{m_1}$. Let P_i be an $m_i \times m_i$ hermitian positive definite matrix for $i = 1, 2$. We define the following inner products on $\mathbb{C}^{m_1}$ and $\mathbb{C}^{m_2}$:

$$\langle \mathbf{x}, \mathbf{y} \rangle_i := \mathbf{y}^* P_i \mathbf{x},\ \mathbf{x}, \mathbf{y} \in \mathbb{C}^{m_i}, \quad i = 1, 2. \tag{1.4}$$

It is straightforward to show that the SVD decomposition of A, viewed as the above operator, with respect to the inner products given by (1.4) is

$$A = U\Sigma V^*, \quad U^* P_2 U = I_{m_2},\ V^* P_1^{-1} V = I_{m_1}, \tag{1.5}$$

where Σ is an $m_2 \times m_1$ diagonal matrix with nonnegative diagonal entries in a decreasing order. A simple way to deduce this decomposition is to observe that $\langle \mathbf{x}, \mathbf{y} \rangle_i = (P_i^{\frac{1}{2}}\mathbf{y})^*(P_i^{\frac{1}{2}}\mathbf{x})$, where $P_i^{\frac{1}{2}}$ is the unique hermitian positive definite square

root of P_i. The decomposition (1.5) is called the *extended singular value decomposition* of A, ESVD for short. The diagonal entries of Σ are called the *extended singular values* of A. ESVD of A corresponds to the standard SVD decomposition of $P_2^{\frac{1}{2}} A P_1^{-\frac{1}{2}}$.

1.4 Extended Singular Value decomposition

In its utmost generality, the SVD deals with a linear operator $T : \mathbf{V}_s \to \mathbf{V}_t$, where $\mathbf{V}_s$ (the source space) and $\mathbf{V}_t$ (the target space) are finite dimensional vector spaces over the real numbers $\mathbb{R}$ (or more generally over the complex numbers $\mathbb{C}$), which are endowed with the inner products $\langle \cdot, \cdot \rangle_s$ and $\langle \cdot, \cdot \rangle_t$ respectively. Choose bases $\mathbf{e}_1, ..., \mathbf{e}_M$ and $\mathbf{f}_1, ..., \mathbf{f}_N$. Then $\mathbf{V}_s$ and $\mathbf{V}_t$ can be identified with the standard vector spaces $\mathbb{R}^M$ and $\mathbb{R}^N$ respectively, and T is represented by the $n \times m$ matrix $A = (a_{ij})_{i=1,j=1}^{NM}$ as explained later on. *Usually* one chooses the inner products $\langle \cdot, \cdot \rangle_s$ and $\langle \cdot, \cdot \rangle_t$ to be given by the standard inner products on $\mathbb{R}^M$ and $\mathbb{R}^N$. That is, one assumes

$$\langle \mathbf{e}_j, \mathbf{e}_j \rangle_s = 1, \ \langle \mathbf{e}_j, \mathbf{e}_k \rangle_s = 0 \quad \text{for } j \neq k, j, k = 1, ..., M, \tag{1.6}$$

$$\langle \mathbf{f}_i, \mathbf{f}_i \rangle_t = 1, \ \langle \mathbf{f}_i, \mathbf{f}_l \rangle_t = 0 \quad \text{for } i \neq l, i, l = 1, ..., N. \tag{1.7}$$

Then the SVD of T is the singular value decomposition $A = U\Sigma V^{\mathrm{T}}$: U, Σ, V are $N \times d, d \times d, M \times d$ matrices respectively, where $U^{\mathrm{T}}U = V^{\mathrm{T}}V$ are equal to the $d \times d$ identity matrix I_d, and Σ is a diagonal matrix with the diagonal entries $\sigma_1 \geq \sigma_2 \geq ...\sigma_d \geq 0$ called the singular values of A. Furthermore

$$\text{rank } A \leq d \leq \min(M, N), \quad 0 < \sigma_{\text{rank } A} \text{ and } \sigma_m = 0 \text{ for } m > \text{rank } A. \tag{1.8}$$

In general, one does not have to assume that $\mathbf{e}_1, ..., \mathbf{e}_M$ and $\mathbf{f}_1, ..., \mathbf{f}_N$ are orthonormal bases. By letting $P_s := (\langle \mathbf{e}_j, \mathbf{e}_k \rangle_s)_{j,k=1}^M$ and $P_t := (\langle \mathbf{f}_i, \mathbf{f}_l \rangle_t)_{i,l=1}^N$ be any positive definite matrices, one obtains the corresponding SVD of T:

$$A = U\Sigma V^{\mathrm{T}}, \quad U^{\mathrm{T}} P_t U = V^{\mathrm{T}} P_s^{-1} V = I_d, \ \Sigma = \mathrm{diag}\,(\sigma_1, ..., \sigma_d)\,, \tag{1.9}$$

where U and V are $N \times d$ and $M \times d$ matrices whose columns form orthonormal sets of d vectors with respect to P_t and P_s^{-1} respectively. Σ is a diagonal matrix with the singular values $\sigma_1 \geq ... \geq \sigma_d \geq 0$ of T satisfying (1.8). We call (1.9) the singular value decomposition of A with respect to P_s, P_t, or *Extended Singular Value Decomposition* (ESVD) of A. We call $\sigma_1, ..., \sigma_d$ the *extended* singular values of A or singular values when no ambiguity arises. Let $\mathbf{a}_1, ..., \mathbf{a}_M$ and $\mathbf{b}_1, ..., \mathbf{b}_N$ be orthonormal bases in $\mathbf{V}_S$ and $\mathbf{V}_t$ respectively. Let $T : \mathbf{V}_s \to \mathbf{V}_t$. Then the conjugate operator $T^* : \mathbf{V}_t \to V_s$ is defined by the equalities

$$\langle T\mathbf{a}_j, \mathbf{b}_i \rangle_t = \langle \mathbf{a}_j, T^* \mathbf{b}_i \rangle_s\,, \quad \text{for } i = 1, ..., N, \ j = 1, ..., M. \tag{1.10}$$

Assume that

$$T\mathbf{a}_j = \sum_{j=1}^{N} c_{ij} \mathbf{b}_i, \quad i = 1, ..., M. \tag{1.11}$$

Then

$$T^* \mathbf{b}_i = \sum_{j=1}^{M} c_{ij} \mathbf{a}_j, \quad i = 1, ..., N. \tag{1.12}$$

That is, the $N \times M$ matrix $C := (c_{ij})_{i=1,j=1}^{N,M}$ represents T in the orthonormal bases $\mathbf{a}_1, ..., \mathbf{a}_M$ and $\mathbf{b}_1, ..., \mathbf{b}_N$, and C^{T} represents T^* in these bases. In particular

$$T^*T : \mathbf{V}_s \to \mathbf{V}_s, \quad TT^* : \mathbf{V}_t \to \mathbf{V}_t \tag{1.13}$$

are presented by $C^{\mathrm{T}}C$ and CC^{T} in the bases $\mathbf{a}_1, ..., \mathbf{a}_M$ and $\mathbf{b}_1, ..., \mathbf{b}_N$ respectively. Clearly

$$\operatorname{rank} T = \operatorname{rank} A = \operatorname{rank} A^{\mathrm{T}} = \operatorname{rank} T^* = \operatorname{rank} A^{\mathrm{T}}A = \operatorname{rank} AA^{\mathrm{T}} = \operatorname{rank} A^*T = \operatorname{rank} TT^{\mathbf{ast}}$$

Since T^*T is self-adjoint ($(T^*T)^* = T^*T$) and nonnegative definite ($\langle T^*T\mathbf{x}, \mathbf{x}\rangle_s \geq 0$ for any $\mathbf{x} \in \mathbf{V}_s$) it follows that there exists an orthonormal basis $\mathbf{c}_1, ..., \mathbf{c}_M$ in $\mathbf{V}_s$ such that

$$T^*T\mathbf{c}_i = \sigma_i^2\mathbf{c}_i,\ \sigma_i \geq 0, \quad \langle \mathbf{c}_i, \mathbf{c}_k\rangle_s = \delta_{ik},\ i, k = 1, ..., M. \tag{1.14}$$

Let $\mathbf{d}_i := \frac{1}{\sigma_i}T\mathbf{c}_i,\ i = 1, ..., \operatorname{rank} T$. Then $\mathbf{d}_1, ..., \mathbf{d}_{\operatorname{rank} T}$ is an orthonormal system of vectors. Extend this system to an orthonormal basis $\mathbf{d}_1, ..., \mathbf{d}_N$ of $\mathbf{V}_t$. Then SVD of T is given by

$$T\mathbf{c}_i = \sigma_i\mathbf{d}_i,\ i = 1, ..., M,\ \langle \mathbf{d}_j, \mathbf{d}_l\rangle_t = \delta_{jl},\ j, l = 1, ..., N,\ \mathbf{d}_i = 0 \text{ if } i > N. \tag{1.15}$$

Lemma 1.3 *Let $\mathbf{V}_s, \mathbf{V}_t$ be finite dimensional vector spaces over $\mathbb{R}$ of dimension M, N and with the inner products $\langle\cdot,\cdot\rangle_s, \langle\cdot,\cdot\rangle_t$ respectively. Let $\mathbf{e}_1, ..., \mathbf{e}_M$ and $\mathbf{f}_1, ..., \mathbf{f}_N$ be bases in $\mathbf{V}_s$ and $\mathbf{V}_t$ with the corresponding positive matrices $P_s = (\langle \mathbf{e}_j, \mathbf{e}_k\rangle_s)_{j,k=1}^M$ and $P_t = (\langle \mathbf{f}_i, \mathbf{f}_l\rangle_t)_{i,l=1}^N$ respectively. Let $T : \mathbf{V}_s \to \mathbf{V}_t$ be a linear transformation given by (1.11). Then the extended singular value decomposition of $A = (a_{ij})_{i,j=1}^{N,M}$ with respect to the positive definite matrices P_s, P_t, corresponding to the singular value decomposition of T in bases $\mathbf{e}_1, ..., \mathbf{e}_M$ and $\mathbf{f}_1, ..., \mathbf{f}_N$ is given by (1.9). Furthermore the operator $T^* : \mathbf{V}_t \to \mathbf{V}_s$ represented by the matrix*

$A^\dagger = (a^\dagger_{ji})_{j,i=1}^{M,N}$ *in the bases* $\mathbf{e}_1, ..., \mathbf{e}_M$ *and* $\mathbf{f}_1, ..., \mathbf{f}_N$*:*

$$T^*\mathbf{f}_i = \sum_{j=1}^{M} a^\dagger_{ji}\mathbf{e}_j,\ i = 1, ..., N, \quad A^\dagger = P_s^{-1}A^{\mathrm{T}}P_t. \tag{1.16}$$

Proof. Let

$$\mathbf{e}_j = \sum_{k=1}^{M} v_{jk}\mathbf{c}_k,\ \mathbf{d}_l = \sum_{i=1}^{N} u_{il}\mathbf{f}_i,\ j = 1, ..., M,\ l = 1, ..., N. \tag{1.17}$$

Hence

$$T\mathbf{e}_j = \sum_{k=1}^{M} v_{jk}T\mathbf{c}_k = \sum_{k=1}^{d} v_{jk}\sigma_k\mathbf{d}_k = \sum_{k=1,i=1}^{d,N} v_{jk}\sigma_k u_{ik}\mathbf{f}_i.$$

Compare the above equalities to (1.11) to deduce the the first part of the equality (1.9) $A = U\Sigma V^{\mathrm{T}}$, where

$$\begin{aligned} &U = (u_{ik})_{i,k=1}^{N,d},\ \tilde{U} = (u_{il})_{i,l=1}^{N} = (U, U_r), \\ &\hat{U} = (\hat{u}_{il})_{i,l=1}^{N} := (\tilde{U})^{-1}, U_0 := (\hat{u}_{il})_{i,l=1}^{d,N},\ \hat{U}^{\mathrm{T}} = (U_0^{\mathrm{T}}, U_d^{\mathrm{T}}) \\ &\\ &V = (v_{jk})_{j,k=1}^{M,d},\ \tilde{V} = (v_{jk})_{j,k=1}^{M} = (V, V_r), \\ &\hat{V} = (\hat{v}_{jk})_{j,k=1}^{M} := (\tilde{V})^{-1},\ V_0 := (\hat{v}_{jk})_{j,k=1}^{d,M},\ \hat{V}^{\mathrm{T}} = (V_0^{\mathrm{T}}, V_d^{\mathrm{T}}) \end{aligned} \tag{1.18}$$

The second part of (1.9) $U^{\mathrm{T}}P_tU = I_d$ is equivalent to the orthonormality of the system $\mathbf{d}_1, ..., \mathbf{d}_d$. We now show the third part of (1.9) $V^{\mathrm{T}}P_s^{-1}V = I_d$. Since $\mathbf{c}_1, ..., \mathbf{c}_M$ is an orthonormal basis in $\mathbf{V}_s$ it follows that $P_s = \tilde{V}\tilde{V}^{\mathrm{T}}$. Hence $P_s^{-1} = (\tilde{V}^{\mathrm{T}})^{-1}(\tilde{V})^{-1}$. Note that since $(\tilde{V})^{-1}\tilde{V} = I_M$ it follows that $(\tilde{V})^{-1}V = Q$ is the $M \times d$ matrix whose d columns are the first d columns of the identity matrix I_M. Hence $V^{\mathrm{T}}P_s^{-1}V = Q^{\mathrm{T}}Q = I_d$.

From the definition of T^* (1.12) it follows that

$$T^*\mathbf{d}_i = \sigma_i\mathbf{c}_i,\ i = 1, ..., N,\ \sigma_i = 0 \text{ and } \mathbf{c}_i = 0 \text{ for } i > M. \tag{1.19}$$

As $\mathbf{f}_i = \sum_{l=1}^{N} \hat{u}_{li}\mathbf{d}_l$ we obtain

$$T^*\mathbf{f}_i = \sum_{l=1}^{N} \hat{u}_{li}T^*\mathbf{d}_l = \sum_{l=1}^{d} \hat{u}_{li}\sigma_l\mathbf{c}_l = \sum_{l,j=1}^{d,M} \hat{u}_{li}\sigma_l\hat{v}_{lj}\mathbf{e}_j, \ .$$

Compare the above equalities with the definition of $A^\dagger$ in (1.16) to deduce that

$$A^\dagger = V_0^{\mathrm{T}}\Sigma U_0. \tag{1.20}$$

Since $\mathbf{d}_1, ..., \mathbf{d}_N$ is an orthonormal basis it follows that $P_t = \hat{U}^{\mathrm{T}}\hat{U}$. As $P_s^{-1} = \hat{V}^{\mathrm{T}}\hat{V}$ and $A = U\Sigma V^{\mathrm{T}}$ a straightforward calculation shows that $A^\dagger = P_s^{-1}A^{\mathrm{T}}P_t$.

1.5 Some Applications of SVD

1.5.1 Analysing DNA gene expression data via SVD

We now give another form of (1.3) which has a significant interpretation in microarray data. Let $\mathbf{u}_1, \cdots, \mathbf{u}_m$ denote the columns of U and $\mathbf{v}_1, \ldots, \mathbf{v}_m$ denote the columns of V. Then (1.1) and (1.3) can be written as

$$E = \sum_{q=1}^{m} \sigma_q\mathbf{u}_q\mathbf{v}_q^{\mathrm{T}} = \sum_{q=1}^{r} \sigma_q\mathbf{u}_q\mathbf{v}_q^{\mathrm{T}}. \tag{1.21}$$

If $\sigma_1 > ... > \sigma_r$ then $\mathbf{u}_q$ and $\mathbf{v}_q$ are determined up to the sign ± 1 for $q = 1, ..., r$. Namely $\mathbf{u}_q$ and $\mathbf{v}_q$ are length 1 eigenvectors of EE^{T} and $E^{\mathrm{T}}E$, respectively, corresponding to the common eigenvalue σ_q^2. (Note the choice of a sign in $\mathbf{v}_q$ forces the unique choice of the sign in $\mathbf{u}_q$.) The vectors $\mathbf{u}_1, \ldots, \mathbf{u}_r$ are called *eigengenes*, the vectors $\mathbf{v}_1, \ldots, \mathbf{v}_r$ are called *eigenarrays* and $\sigma_1, \ldots, \sigma_r$ are called *eigenexpressions*. The rank r can be viewed as the number of different biological functions of n genes observed in m experiments. The eigenarrays $\mathbf{v}_1, \ldots, \mathbf{v}_r$ give the

principal r orthogonal directions in $\mathbb{R}^m$ corresponding to $\sigma_1, \ldots, \sigma_r$. The eigengenes $\mathbf{u}_1, \ldots, \mathbf{u}_r$ give the principal r orthogonal directions in $\mathbb{R}^n$ corresponding to $\sigma_1, \ldots, \sigma_r$. The eigenexpressions describe the relative significance of each biofunction. From the data given in [4], it seems that the number of significant singular values never exceeds $\frac{m}{2}$. See the discussion on the number of significant singular values in the beginning of §3. The essence of the FRAA algorithm, suggested in this thesis, is based on this observation.

1.5.2 Low rank approximation of matrices

A well-known technique for dimension reduction is the low rank approximation by the singular value decomposition [21].

Let E be the data matrix, and let $E = U\Sigma V^T$ be the SVD of E, where U and V are orthogonal and Σ is diagonal. Then, for a given κ, the optimal rank κ approximation of E is given by

$$\hat{E}_\kappa = U_\kappa \Sigma_\kappa V_\kappa^T,$$

where U_κ and V_κ are the matrices formed by the first κ columns of the matrices U and V respectively, and Σ_κ is the κ-th principal submatrix of Σ. A key property of this rank κ approximation is that it achieves the best possible approximation with respect to the Frobenius norm, (and more generally with respect to any unitarily-invariant norm), among all matrices with rank κ, .

Denote by $||E||_{\mathcal{F}}$ the Frobenius (ℓ_2) norm of E. It is the Euclidean norm of E viewed as a vector with nm coordinates. Each term $\mathbf{u}_q\mathbf{v}_q^{\mathrm{T}}$ in (1.21) is a rank one

matrix with $||\mathbf{u}_q\mathbf{v}_q^{\mathrm{T}}||_{\mathcal{F}} = 1$. Let $\mathcal{R}(n, m, k)$ denote the set of $n \times m$ matrices of at most rank k ($m \geq k$). Then for each k, $k \leq r$, the SVD of E gives the solution to the following approximation problem:

$$\min_{F \in \mathcal{R}(n,m,k)} ||E - F||_{\mathcal{F}} = ||E - \sum_{q=1}^{k} \sigma_q \mathbf{u}_q \mathbf{v}_q^{\mathrm{T}}||_{\mathcal{F}} = \sqrt{\sum_{q=k+1}^{r} \sigma_q^2}. \tag{1.22}$$

If $\sigma_k > \sigma_{k+1}$ then $\sum_{q=1}^{k} \sigma_q \mathbf{u}_q \mathbf{v}_q^{\mathrm{T}}$ is the unique solution to the above minima problem. For our purposes, it will be convenient to assume that $\sigma_q = 0$ for any $q > m$.

2 Randomized Low Rank Approximation of a Matrix

In many applied settings where one is dealing with a very large data set, it is important to reduce the size of data in order to make an inference about the features of data set, in a timely manner. Assume that the data set is represented by an $m \times n$ matrix $A \in \mathbb{R}^{m \times n}$. It is important to find an approximation $B \in \mathbb{R}^{m \times n}$ of a specified rank k to A, where k is much smaller than m and n.

Here are several motivations to obtain such B. First, the storage space needed for B is $k(m + n)$, which is much smaller than the storage space mn needed for A. Indeed, B can be represented as

$$B = \mathbf{x}_1\mathbf{y}_1^{\mathrm{T}} + \mathbf{x}_2\mathbf{y}_2^{\mathrm{T}} + \ldots \mathbf{x}_k\mathbf{y}_k^{\mathrm{T}}, \quad \mathbf{x}_1, \ldots, \mathbf{x}_k \in \mathbb{R}^m, \ \mathbf{y}_1, \ldots, \mathbf{y}_k \in \mathbb{R}^n, \tag{2.23}$$

where $\mathbf{x}_1, \ldots, \mathbf{x}_k$ and $\mathbf{y}_1, \ldots, \mathbf{y}_k$ are k column vectors with m and n coordinates respectively. We store the vectors $\mathbf{x}_1, \ldots, \mathbf{x}_k, \mathbf{y}_1, \ldots, \mathbf{y}_k$, which need the storage $k(m+n)$, and compute the entries of B, when needed, using the above expression for B.

The second most common application is clustering algorithms as in [1], [2], [11], [13] and [30]. Assume that our data represents n points and each point has m coordinates. That is each point represented by a column of the matrix A. We want to cluster the points in such a way that the distance between points in the same cluster is much smaller than the distance between any two points from different clusters. One way to do this is to project all the points on the k main orthonormal directions encoded by the first k left singular vectors of A. Then cluster using either k one dimensional subspaces or the whole k dimensional subspace. The k-rank approximation B gives the approximation to this k dimensional subspace and the approximation to the first k singular vectors. Another way to cluster is to use the projective clustering. It is known that a fast SVD, i.e. fast k-rank approximation is a main tool in this area [2] and [11].

The third application is in DNA microarrays, in particular in data imputation [18]. Let A be a gene expression matrix. The m rows of the matrix A are indexed by the genes, while the n columns are indexed by the number of experiments. It is known that the effective rank of A, k, is usually much less than n. The SVD decomposition is the main ingredient of the FRAA, (Fixed Rank Approximation Algorithm), which is successfully implemented in [18] to impute the corrupted entries of A. The fast k- approximation algorithm suggested in this paper, combined

with the clustering of similar genes, can be implemented to improve the FRAA algorithm.

One way to find a *fast* k-rank approximation is to choose at random $l \geq k$ columns or rows of A and obtain from them k-rank approximations of A. This is basically the spirit of the algorithm suggested by Frieze, Kannan and Vempala in [19]. We call this algorithm the FKV algorithm. Assuming a statistical model for the distribution of the entries of A, the authors give some error bounds on their k-rank approximation.

In order to grasp the nature of the FKV algorithm, we need to discuss the random projection method. Santosh S. Vempala has a thorough discussion of the random projection method. [36]

2.1 Random Projection Method

Let $\mathbf{u} = (u_1, \ldots, u_n)$. In order to obtain the projection of $\mathbf{u}$, we consider an orthonormal matrix R, whose entries are uniformly distributed. Then we scale $R^T\mathbf{u}$ with $\sqrt{n/k}$ to obtain the projection $\mathbf{v}$, $\mathbf{v} = (\sqrt{n/k})R^T\mathbf{u}$. The rational for choosing a scaling factor of $\sqrt{n/k}$ is to have the expected value $E(\|v\|^2) = \|u\|^2$. There are various strategies for coming up with the random matrix R, which are data and application dependent.

One of the interesting properties of random projection is that while it is useful for reducing dimensionality, it preserves pairwise distances with high probability. These properties can be represented by the following lemma.

Lemma 2.1 (Johnson and Lindenstrauss) *For any $0 < \epsilon < 1/2$ and any set of points S in $\mathbb{R}^n$ with $|S| = m$, upon projection to a uniform rank κ-dimensional subspace where $\kappa \geq 1 + \frac{9 \ln m}{\epsilon^2 - (2/3)\epsilon^3}$, the following property holds:*
With probability at least $1/2$, for every pair $\mathbf{u}, \mathbf{u}'$ in S,

$$(1-\epsilon)\|\mathbf{u}-\mathbf{u}'\|^2 \leq \|f(\mathbf{u}) - f(\mathbf{u}')\|^2 \leq (1+\epsilon)\|\mathbf{u}-\mathbf{u}'\|^2$$

where $f(\mathbf{u})$ and $f(\mathbf{u}')$ are projections of $\mathbf{u}$ and $\mathbf{u}'$.

Random Projection Algorithm. Random projection preserves distances approximately, while reducing dimensionality. We could use the following steps to achieve these goals.

1. Project an l-dimensional data point to k-dimensional space. ($k \geq 0$)
2. Find the k largest singular values for obtaining the low-rank approximation of the original data matrix.

Suppose we project an $m \times n$ matrix A using an $m \times l$ matrix B, $B = \sqrt{\frac{n}{l}} R^T A$. Then SVD of A and B could be written as

$$A = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T, \quad B = \sum_{i=1}^{t} \lambda_i \mathbf{u}_i' \mathbf{v}_i'^T$$

where $\mathbf{u}_i$ and $\mathbf{u}_i'$ are left singular vectors and $\mathbf{v}_i$ and $\mathbf{v}_i'$ are right singular values of A and B respectively. The following lemma indicates that the singular values are approximately preserved.

Lemma 2.2 *Let $\epsilon > 0$. If $l \geq C\frac{\log n}{\epsilon^2}$ for sufficiently large constant C, then*

$$\sum_{i=1}^{n} \lambda_i^2 \geq (1-\epsilon) \sum_{i=1}^{\kappa} \sigma_i^2$$

We close this discussion on the random projection by the following theorem which emphasizes that the matrix obtained by random projection is as good as the least κ-rank approximation.

Theorem 2.3 *Let A_κ be the best k-rank approximation of A, and let $\tilde{A}_\kappa$ be the matrix obtained by random projection. If $l > C \log n/\epsilon^2$ for large enough constant C, then*

$$\|A - \tilde{A}_\kappa\|_F^2 \leq \|A - A_\kappa\|_F^2 + 2\epsilon\|A_\kappa\|_F^2$$

Low rank approximation (FKV method).

Given an $m \times n$ matrix A, we would like to approximate a few of the top singular values and their associated singular vector. Here we present the FKV algorithm [12].

The FKV algorithm starts by choosing l rows of A and forming an $l \times n$ matrix L. Let $p_1, \ldots, p_m$ be real numbers such that $0 \leq p_i \leq 1$ and $\sum_{i=1}^m p_i = 1$. The algorithm will input a matirx $A_{m\times n}$ and integers $l \leq m$ and $k \leq l$.

We then construct a matrix L in the following way: Choose $i \in \{1, \ldots, m\}$ where the probability of choosing i is equal to p_i. Then scale the i-th row $A_{(i)}$ of A by $\sqrt{lp_i}$, to obtain $A_{(i)}/\sqrt{lp_i}$, which will constitute the i-th row of L.

In the next step we compute LL^T and its singular decomposition

$$LL^T = \sum_{t=1}^{l} \sigma_t^2 \mathbf{u}_{(t)} \mathbf{u}_{(t)}^T$$

Let H be the matrix whose rows are

$$h_{(t)} := L^T \mathbf{u}_{(t)}/\|L_{34}^T \mathbf{u}_{(t)}\|, t = 1, \ldots, k$$

and whose singular values are σ_t^2.

Notice that $h_{(t)}$ are the right singular values of L, and SVD of L is

$$L = \sum_{t=1}^{l} \sigma_t \mathbf{u}_{(t)} h_{(t)}^T$$

The weak point of FKV algorithm is its inability to improve iteratively FKV approximation by incorporating additional parts of A. In the following section, we present our algorithm [17], which overcomes this shortcoming.

2.2 Fast Monte-Carlo Rank Approximation

The weak point of the FKV algorithm is its inability to improve iteratively FKV approximation by incorporating additional parts of A. In fact in the recent paper [11], which uses FKV random algorithm, this point is mentioned specifically in the end of the paper: "..., it would be interesting to design an algorithm that improves this approximation by accessing A (or parts of A) again."

We provide here a sampling framework for iterative updates of k-rank approximations of A, by reading iteratively additional columns (rows) of A, which improves *for sure* the approximation B each time it is updated. The quality of the approximation of B is given by the Frobenius norm $||B||_{\mathcal{F}}$, the successive values of which are a nondecreasing sequence under these iterations. The rate of increase of the norms $||B||_{\mathcal{F}}$ can be used to give a stopping rule for terminating the algorithm. Also, the updating algorithm of the k-rank approximation gives approximation to the first k singular values of A, and the approximations to the first k left and right singular vectors of A. We believe that this algorithm will have many applications

in data mining, data storage and data analysis.

The following set of theorems ([17]) will be useful in the construction of the algorithm.

Theorem 2.4 *Let* $A \in \mathbb{R}^{m\times n}$ *and* $k \in [1, \min(m,n)]$. *Let* $\mathbf{x}_1, \ldots, \mathbf{x}_k$ *and* $\mathbf{y}_1, \ldots, \mathbf{y}_k$ *be two sets of orthonormal vectors in* $\mathbb{R}^m$ *and* $\mathbb{R}^n$ *respectively. Then*

$$\sum_{i=1}^{k}(A^{\mathrm{T}}\mathbf{x}_i)^{\mathrm{T}}(A^{\mathrm{T}}\mathbf{x}_i) \leq \sum_{i=1}^{k}\sigma_i^2, \quad \sum_{i=1}^{k}(A\mathbf{y}_i)^{\mathrm{T}}(A\mathbf{y}_i) \leq \sum_{i=1}^{k}\sigma_i^2. \tag{2.24}$$

Equality in the first or second inequality occurs if $\mathrm{span}(\mathbf{x}_1, \ldots, \mathbf{x}_k)$ *or* $\mathrm{span}(\mathbf{y}_1, \ldots, \mathbf{y}_k)$, *contains* k *linearly independent left or right singular vectors of* A, *corresponding the* k *maximal singular values of* A *respectively.*

The above characterization follows from the maximal, (Ky Fan), characterization, of the sum of the first largest eigenvalues of a real symmetric matrix:

Theorem 2.5 *Let* $S \in S_p(\mathbb{R})$ *be a real* $p \times p$ *symmetric matrix. Let* $\lambda_1 \geq \ldots \geq \lambda_p$ *be the eigenvalues* S *arranged in a decreasing order and listed with their multiplicities. Let* $\mathbf{w}_1, \ldots, \mathbf{w}_p \in \mathbb{R}^p$ *be an orthonormal set of the corresponding eigenvectors of* S*:* $S\mathbf{w}_i = \lambda_i\mathbf{w}_i, i = 1, \ldots, p$. *Let* $k \in [1, p]$ *be an integer. Then for any orthonormal set* $\mathbf{x}_1, \ldots, \mathbf{x}_k \in \mathbb{R}^p$

$$\sum_{i=1}^{k}\mathbf{x}_i^{\mathrm{T}}S\mathbf{x}_i \leq \sum_{i=1}\lambda_i = \sum_{i=1}^{k}\mathbf{w}_i^{\mathrm{T}}S\mathbf{w}_i. \tag{2.25}$$

Equality holds if and only if the subspace $\mathrm{span}(\mathbf{x}_1, \ldots, \mathbf{x}_k)$ *contains* k *linearly independent eigenvectors of* S *corresponding to the eigenvalues* $\lambda_1, \ldots, \lambda_k$.

See for example [14] for proofs and the references. Note that $\mathbf{x}_1, \ldots, \mathbf{x}_k \in \mathbb{R}^m$ is a system of orthonormal vectors in $\mathbb{R}^m$ if and only if the $m \times k$ matrix $(\mathbf{x}_1, \ldots, \mathbf{x}_k)$ is in $\mathrm{O}_{mk}(\mathbb{R})$.

To obtain Theorem 2.4 from Theorem 2.5 we let $p = m$, (or n) and S be equal to AA^{T}, (or $A^{\mathrm{T}}A$). In (2.24) we emphasized the complexity of the computations of the left-hand side of the inequalities. See also [18] for applications of Theorem 2.5 to data imputation in DNA microarrays.

Corollary 2.6 *Let $A \in \mathbb{R}^{m \times n}$ and $k \in [1, \min(m, n)]$ be an integer. Then for any two k-orthornormal systems $\mathbf{x}_1, \ldots, \mathbf{x}_k \in \mathbb{R}^m$ and $\mathbf{y}_1, \ldots, \mathbf{y}_k \in \mathbb{R}^m$ the following equalities hold:*

$$||A - \sum_{i=1}^{k} \mathbf{x}_i(\mathbf{x}_i^{\mathrm{T}}A)||_{\mathcal{F}}^2 = ||A||_{\mathcal{F}}^2 - \sum_{i=1}^{k} (A^{\mathrm{T}}\mathbf{x}_i)^{\mathrm{T}}(A^{\mathrm{T}}\mathbf{x}_i), \tag{2.26}$$

$$||A - \sum_{i=1}^{k} (A\mathbf{y}_i)\mathbf{y}_i^{\mathrm{T}}||_{\mathcal{F}}^2 = ||A||_{\mathcal{F}}^2 - \sum_{i=1}^{k} (A\mathbf{y}_i)^{\mathrm{T}}(A\mathbf{y}_i). \tag{2.27}$$

In particular the best k-rank approximation $\hat{E}_k$ of A is given by $\sum_{i=1}^{k} \mathbf{u}_i(\mathbf{u}_i^{\mathrm{T}}A)$ and $\sum_{i=1}^{k} (A\mathbf{v}_i)\mathbf{v}_i^{\mathrm{T}}$, where $\mathbf{u}_1, \ldots, \mathbf{u}_k \in \mathbb{R}^m$ and $\mathbf{v}_1, \ldots, \mathbf{v}_k \in \mathbb{R}^n$ are an orthonormal sets of the left and right singular vectors of A corresponding to $\sigma_1, \ldots, \sigma_k$.

The next theorem is the key theorem for *updating* the k-rank approximation for $\sum_{i=1}^{k} \mathbf{x}_i(\mathbf{x}_i^{\mathrm{T}}A)$, (or $\sum_{i=1}^{k} (A\mathbf{y}_i)\mathbf{y}_i^{\mathrm{T}}$), for some $(\mathbf{x}_1, \ldots, \mathbf{x}_k) \in \mathrm{O}_{mk}(\mathbb{R})$, (or some $(\mathbf{y}_1, \ldots, \mathbf{y}_k) \in \mathrm{O}_{nk}(\mathbb{R})$).

Theorem 2.7 *Let $\mathbf{x}_1, \ldots, \mathbf{x}_k \in \mathbb{R}^m$, (or $\mathbf{y}_1, \ldots, \mathbf{y}_k \in \mathbb{R}^n$), be an orthonormal system in $\mathbb{R}^m$, (or in $\mathbb{R}^n$). Let $\mathbf{w}_1, \ldots, \mathbf{w}_l \in \mathbb{R}^m$, (or $\mathbf{z}_1, \ldots, \mathbf{z}_l \in \mathbb{R}^n$),*

be a given set in $\mathbb{R}^m$*, (or in* $\mathbb{R}^n$*). Perform the Modified Gram-Schmidt process on* $\mathbf{x}_1, \ldots, \mathbf{x}_k, \mathbf{w}_1, \ldots, \mathbf{w}_l$*, (or* $\mathbf{y}_1, \ldots, \mathbf{y}_k, \mathbf{z}_1, \ldots, \mathbf{z}_l$*), to obtain an orthonormal set* $\mathbf{x}_1, \ldots, \mathbf{x}_p \in \mathbb{R}^m$*, (or* $\mathbf{y}_1, \ldots, \mathbf{y}_p \in \mathbb{R}^n$*), where* $k \le p \le k + l$*. Assume that* $k < p$*, i.e.* $\mathrm{span}(\mathbf{w}_1, \ldots, \mathbf{w}_l) \not\subseteq \mathrm{span}(\mathbf{x}_1, \ldots, \mathbf{x}_k)$*, (or* $\mathrm{span}(\mathbf{z}_1, \ldots, \mathbf{z}_l) \not\subseteq \mathrm{span}(\mathbf{y}_1, \ldots, \mathbf{x}_k)$*). Form* $p \times p$ *real symmetric matrix* $S := ((A^{\mathrm{T}}\mathbf{x}_i)^{\mathrm{T}}(A^{\mathrm{T}}\mathbf{x}_j))_{i,j=1}^p$*, (or* $S := ((A\mathbf{y}_i)^{\mathrm{T}}(A\mathbf{y}_j))_{i,j=1}^p$*), and assume that* $\lambda_1 \ge \ldots \ge \lambda_k$ *are the* k*-largest eigenvalues of* S *with the corresponding* k*-orthonormal vectors* $\mathbf{o}_1, \ldots, \mathbf{o}_k \in \mathbb{R}^p$*. Let* $O := (\mathbf{o}_1, \ldots, \mathbf{o}_k) \in \mathrm{O}_{pk}(\mathbb{R})$ *and define* k*-orthonomal vectors* $\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_k \in \mathbb{R}^m$*, (or* $\tilde{\mathbf{y}}_1, \ldots, \tilde{\mathbf{y}}_k \in \mathbb{R}^n$*), as follows:*

$$(\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_k) = (\mathbf{x}_1, \ldots, \mathbf{x}_p)O, \quad (or \ (\tilde{\mathbf{y}}_1, \ldots, \tilde{\mathbf{y}}_k) = (\mathbf{y}_1, \ldots, \mathbf{y}_p)O). \tag{2.28}$$

Then

$$\sum_{i=1}^k (A^{\mathrm{T}}\mathbf{x}_i)^{\mathrm{T}}(A^{\mathrm{T}}\mathbf{x}_i) \le \sum_{i=1}^k (A^{\mathrm{T}}\tilde{\mathbf{x}}_i)^{\mathrm{T}}(A^{\mathrm{T}}\tilde{\mathbf{x}}_i), \tag{2.29}$$
$$(or \ \sum_{i=1}^k (A\mathbf{y}_i)^{\mathrm{T}}(A\mathbf{y}_i) \le \sum_{i=1}^k (A\tilde{\mathbf{y}}_i)^{\mathrm{T}}(A\tilde{\mathbf{y}}_i)).$$

Furthermore

$$\lambda_i = (A^{\mathrm{T}}\tilde{\mathbf{x}}_i)^{\mathrm{T}}(A^{\mathrm{T}}\tilde{\mathbf{x}}_i), i = 1, \ldots, k, \quad (A^{\mathrm{T}}\tilde{\mathbf{x}}_i)^{\mathrm{T}}(A^{\mathrm{T}}\tilde{\mathbf{x}}_j) = 0 \ for \ i \ne j, \tag{2.30}$$
$$(or \ \lambda_i = (A\tilde{\mathbf{y}}_i)^{\mathrm{T}}(A\tilde{\mathbf{y}}_i), i = 1, \ldots, k, \quad (A\tilde{\mathbf{y}}_i)^{\mathrm{T}}(A\tilde{\mathbf{y}}_j) = 0 \ for \ i \ne j.)$$

We now explain the essence of Theorem 2.7. View $\mathbf{x}_1, \ldots, \mathbf{x}_k$ as an approximation to the first k-left singular vectors of A, and $\sum_{i=1}^k \mathbf{x}_i(A^{\mathrm{T}}\mathbf{x}_i)^{\mathrm{T}}$ as the k-rank approximation to A. Hence $(A^{\mathrm{T}}\mathbf{x}_i)^{\mathrm{T}}(A^{\mathrm{T}}\mathbf{x}_i)$ is an approximation to σ_i^2 of A for $i = 1, \ldots, k$. Read additional vectors $\mathbf{w}_1, \ldots, \mathbf{w}_l \in \mathbb{R}^m$ such that at least one of

these vectors is not in the subspace spanned by $\mathbf{x}_1, \ldots, \mathbf{x}_k$. Let $\mathbf{X}$ be the subspace spanned by $\mathbf{x}_1, \ldots, \mathbf{x}_k$ and $\mathbf{w}_1, \ldots, \mathbf{w}_l$. Hence $\mathbf{x}_1, \ldots, \mathbf{x}_k, \ldots, \mathbf{x}_p$ is the orthonormal basis of $\mathbf{X}$ obtained from the vectors $\mathbf{x}_1, \ldots, \mathbf{x}_k, \mathbf{w}_1, \ldots, \mathbf{w}_l$, using the Gram-Schmidt process. Note that $k < p \leq k + l$, and in general one has that $p = k + l$. Consider the $p \times p$ nonnegative definite matrix $S = ((A^{\mathrm{T}}\mathbf{x}_i)^{\mathrm{T}}(A^{\mathrm{T}}\mathbf{x}_j))_{i,j=1}^{p}$. Find its first k eigenvectors to obtain $\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_k \in \mathbb{R}^m$. Then $C := \sum_{i=1}^{k} \tilde{\mathbf{x}}_i (A^{\mathrm{T}}\tilde{\mathbf{x}}_i)^{\mathrm{T}}$ is the best approximation of A by matrix B of rank at most k, whose columns are in the subspace $\mathbf{X}$. In particular, the approximation C is better than the previous approximation $\sum_{i=1}^{k} \mathbf{x}_i (A^{\mathrm{T}}\mathbf{x}_i)^{\mathrm{T}}$, which is equivalent to the (2.29). Similar situation holds for in the second case $\mathbf{y}_1, \ldots, \mathbf{y}_k, \mathbf{z}_1, \ldots, \mathbf{z}_l \in \mathbb{R}^n$.

Outline of the Proof of Theorem 2.7. Let $S = (s_{ij})_{i,j=1}^{p}$. Let $\mathbf{e}_i = (\delta_{i1}, \ldots, \delta_{ip})^{\mathrm{T}}, i = 1, \ldots, p$ be the standard orthonormal basis in $\mathbb{R}^p$. Assume that we are dealing with the orthonormal set $\mathbf{x}_1, \ldots, \mathbf{x}_p \in \mathbb{R}^m$. Use the definition of A and the Ky Fan characterization of the sum of the maximal k eigenvalues of symmetric S to deduce

$$\sum_{i=1}^{k} (A^{\mathrm{T}}\mathbf{x}_i)^{\mathrm{T}}(A^{\mathrm{T}}\mathbf{x}_i) = \sum_{i=1}^{k} \mathbf{e}_i^{\mathrm{T}} S \mathbf{e}_i \leq \sum_{i=1}^{k} \lambda_i = \sum_{i=1}^{k} \mathbf{o}_i^{\mathrm{T}} S \mathbf{o}_i.$$

Let $C := A^{\mathrm{T}}(\mathbf{x}_1, \ldots, \mathbf{x}_p)$. Then $S = C^{\mathrm{T}}C$. Hence the $\sqrt{\lambda_1} \geq \ldots \geq \sqrt{\lambda_p}$ are the singular values of C and $\mathbf{o}_1, \ldots, \mathbf{o}_p$ are the right singular vectors of C. Thus $C\mathbf{o}_i = \sqrt{\lambda_i}\mathbf{t}_i \in \mathbb{R}^n$, where $\mathbf{t}_i$ is the left singular vector of C corresponding to the singular value $\sqrt{\lambda_i}$ for $i = 1, \ldots, p$. A straightforward calculation shows that $\lambda_i = \mathbf{o}_i^{\mathrm{T}} S \mathbf{o}_i = (A^{\mathrm{T}}\tilde{\mathbf{x}}_i)^{\mathrm{T}}(A^{\mathrm{T}}\tilde{\mathbf{x}}_i)$ for $i = 1, \ldots, k$. Hence the first inequality in (2.29) holds. Furthermore we deduce the first set of equalities in (2.30) for $i = 1, \ldots, k$.

The second set of equalities in (2.30) follows from the orthonormality of $\mathbf{t}_1, \ldots, \mathbf{t}_k$.

Similar arguments apply when dealing with the orthonomal system $\mathbf{y}_1, \ldots, \mathbf{y}_p$.

□

Algorithm [17]

One starts the algorithm by choosing the first k-rank approximation to A as follows. Let $\mathbf{c}_1, \ldots, \mathbf{c}_n \in \mathbb{R}^m$ and $\mathbf{r}_1^{\mathrm{T}}, \ldots, \mathbf{r}_m^{\mathrm{T}} \in \mathbb{R}^n$ be the n columns and the m rows of A. (We view the rows of A as row vectors.) Choose $1 \leq n_1 < \ldots < n_k \leq n$, (or $1 \leq m_1 < \ldots < m_k \leq m$), to be k integers. Let $\mathbf{x}_1, \ldots, \mathbf{x}_q \in \mathbb{R}^m$, (or $\mathbf{y}_1, \ldots, \mathbf{y}_q \in \mathbb{R}^n$), be the orthonormal set obtained from $\mathbf{c}_{n_1}, \ldots, \mathbf{c}_{n_k}$, (or $\mathbf{r}_{m_1}^{\mathrm{T}}, \ldots, \mathbf{r}_{m_k}^{\mathrm{T}}$), using the Modified Gram-Schmidt process. Then let

$$B_0 := \sum_{i=1}^{q} \mathbf{x}_i (A^{\mathrm{T}} \mathbf{x}_i)^{\mathrm{T}}, \quad (\texttt{or} \;\; B_0 := \sum_{i=1}^{q} (A\mathbf{y}_i)\mathbf{y}_i^{\mathrm{T}}). \tag{2.31}$$

That is, the first k-rank B_0 approximation to A is obtained as follows. Choose at random k columns, (or rows), of the data matrix A. Apply the Gram-Schmidt process to them to obtain the orthonormal column vectors $\mathbf{x}_1, \ldots, \mathbf{x}_q$ with m coordinates, (or the orthonormal column vectors $\mathbf{y}_1, \ldots, \mathbf{y}_q$ with n coordinates). In general $q = k$, but in some cases if $\mathbf{x}_1, \ldots, \mathbf{x}_k$ are linearly dependent, (or if $\mathbf{y}_1, \ldots, \mathbf{y}_q$ are linearly dependent), $q < k$. Assume for simplicity of the exposition that $q = k$. Then B_0 is of the form (2.23) where $\mathbf{y}_i = A^{\mathrm{T}}\mathbf{x}_i, i = 1, \ldots, k$, (or $\mathbf{x}_i = A\mathbf{y}_i, i = 1, \ldots, k$.)

Now update iteratively the k-rank approximation of B_{t-1} of A to B_t, using Theorem 2.7, by letting $\mathbf{w}_1 := \mathbf{c}_{j_1}, \ldots, \mathbf{w}_l = \mathbf{c}_{j_l} \in \mathbb{R}^m$, (or $\mathbf{z}_1 := \mathbf{r}_{j_1}^{\mathrm{T}}, \ldots, \mathbf{z}_l = \mathbf{r}_{j_l}^{\mathrm{T}} \in \mathbb{R}^n$), for some l integers $1 \leq j_1 < \ldots < j_l \leq n$, (or $1 \leq j_1 < \ldots < j_l \leq m$).

That is, we choose another l sets of columns of A, (or rows of A), preferably that were not chosen before, and update the k-rank approximation using the algorithm suggested by Theorem 2.7 to obtain an improved k-rank approximation B_t of A.

Furthermore one can use the k-rank approximation B_t from the above algorithm to approximate the first k-singular values $\sigma_1, \ldots, \sigma_k$, and the left and the right singular vectors $\mathbf{u}_1, \ldots, \mathbf{u}_k$ and $\mathbf{v}_1, \ldots, \mathbf{v}_k$ as follows. First the square roots $\sqrt{\lambda_1(S)}, \ldots, \sqrt{\lambda_k(S)}$ of the matrix S are approximations for $\sigma_1, \ldots, \sigma_k$. If S was obtained using $\mathbf{x}_1, \ldots, \mathbf{x}_p \in \mathbb{R}^m$ then the vectors $\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_k$ approximate $\mathbf{u}_1, \ldots, \mathbf{u}_k$. Let $\tilde{\mathbf{v}}_i := A^{\mathrm{T}}\mathbf{u}_i, i = 1, \ldots, k$. Then $||\tilde{\mathbf{v}}_i|| = \sqrt{\lambda_i(S)}$ is an approximation to σ_i of A for $i = 1, \ldots, k$. The renormalized $\tilde{\mathbf{v}}_i$ which are given as $\frac{1}{||\tilde{\mathbf{v}}_i||}\tilde{\mathbf{v}}_i$ approximate the right singular eigenvectors $\mathbf{v}_i$ for $i = 1, \ldots, k$.

If S was obtained using $\mathbf{y}_1, \ldots, \mathbf{y}_p \in \mathbb{R}^n$ then the vectors $\tilde{\mathbf{y}}_1, \ldots, \tilde{\mathbf{y}}_k$ approximate $\mathbf{v}_1, \ldots, \mathbf{v}_k$ and the vectors $\frac{A\tilde{\mathbf{y}}_1}{||A\tilde{\mathbf{y}}_1||}, \ldots, \frac{A\tilde{\mathbf{y}}_k}{||A\tilde{\mathbf{y}}_k||}$ approximate $\mathbf{u}_1, \ldots, \mathbf{u}_k$.

Fast k-rank approximation and SVD algorithm

Input: positive integers m, n, k, l, N, $m \times n$ matrix A, $\epsilon > 0$.

Output: an $m \times n$ k-rank approximation B_f of A, with the ratios $\frac{||B_0||}{||B_t||}$ and $\frac{||B_{t-1}||}{||B_t||}$, approximations to k-singular values and k left and right singular vectors of A.

1. Choose k-rank approximation B_0 using k columns (rows) of A.

2. **for** $t = 1$ **to** N

- Choose l columns (rows) from A at random and update B_{t-1} to B_t.

- Compute the approximations to k-singular values, and k left and right singular vectors of A.

- If $\frac{||B_{t-1}||}{||B_t||} > 1 - \epsilon$ let $f = t$ and finish.

We now explain briefly the main steps of our algorithm. We read the dimensions m, n of the data matrix A. We set N as the maximal number of iterations we are going to execute to find the k-rank approximation of A. We read the entries of the data matrix A, and finally the small parameter $\epsilon > 0$. We choose the k-rank approximation B_0 using (2.31) as explained above. Assume that B_{t-1} is the current k-rank approximation to A. Then we pick up additional l columns, (or rows), of A and update B_{t-1} to B_t using Theorem 2.7 as explained. Recall that $||B_{t-1}|| \leq ||B_t||$. If the relative improvement in $||B_t||$ is less than ϵ, i.e. $\frac{||B_{t-1}||}{||B_t||} > 1 - \epsilon$, we are satisfied the approximation B_t and finish our algorithm. If this does not happen

then our algorithm stops after the N iteration.

3 Cluster Analysis

3.1 Clusters and Clustering

Clustering is the process of grouping data objects into a set of disjoint classes called clusters, so that the objects within the class have high similarity to each other, while objects in separate clusters are more dissimilar [25]. Clustering is an example of unsupervised classification. Classification refers to a procedure that assigns data objects to a set of classes. Unsupervised means that clustering does not rely on predefined classes and training examples while classifying the data objects. Thus, clustering is distinguished from pattern recognition or the areas of statistics known as discriminant analysis and decision analysis, which seek to find rules for classifying objects from a given set of pre-classified objects.

3.2 Various Gene Expression Clustering Procedures

Usually a microarray experiment contains 10^6 genes. One of the characteristics of gene expression data is that it is meaningful to cluster both genes and samples. We could group the co-expressed genes in a cluster according to their expression patterns. In such gene based clustering, the genes are treated as objects, while the samples are the features. On the other hand the samples can be partitioned into homogeneous groups. Each group may correspond to some particular cancer types.

Such sample based clustering regards the samples as the objects and the genes as the features.

3.2.1 Proximity (Similarity) Measurement

Proximity measurement for gene expression data computes the distance between the two data points. As mentioned before, gene profile in a gene expression matrix can be formalized as a vector $\mathbf{x}_i = \{x_{ij} : 1 \leq j \leq m\}$, where x_{ij} is the value of the j-th

feature the i-th data object, and m is the number of features or experiments. This similarity between two expression profiles $\mathbf{x}_i$ and $\mathbf{x}_j$ is measured by the distance function of corresponding vectors $\mathbf{x}_i$ and $\mathbf{x}_j$.

Euclidean distance is one of the most commonly used methods to measure the distance between two gene expression profiles. The distance between two gene expression profiles $\mathbf{x}_i$ and $\mathbf{x}_j$ in m-dimensional space is defined as

$$d_{\text{Euclidean}}(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{k=1}^{m}(x_{ik} - x_{jk})^2}$$

However, for gene expression data, the overall Euclidean distance does not perform well for shifting or scaled patterns (profiles). To circumvent this, each gene profile is standardized with zero mean and variance one, before calculating the distance.

An alternate measure of similarity is the Pearson correlation coefficient, which measures similarity between the shapes of two expression patterns between two

profiles.

$$\text{Pearson}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sum_{k=1}^{m}(x_{ik} - \mu_{x_i})(x_{jk} - \mu_{x_j})}{\sqrt{\sum_{k=1}^{m}(x_{ik} - \mu_{x_i})^2}\sqrt{\sum_{k=1}^{m}(x_{jk} - \mu_{x_j})^2}}$$

where μ_{x_i} and μ_{x_j} are the means for gene profiles of $\mathbf{x}_i$ and $\mathbf{x}_j$ respectively. Pearson's correlation coefficient views each gene profile as a random variable, and measures the similarity between two gene profiles by calculating the linear relationship between their distribution. The shortcoming of Pearson correlation is that it is not robust for outliers.

3.2.2 Hierarchical Cluster Analysis

One possibility for clustering objects is their hierarchical aggregation. Here the objects are combined according to their distances from or similarities to each other. Cluster formation is based on splitting the whole set of objects into individual clusters. With the more frequently used agglomerative clustering, one starts with single objects and measures them to larger object groups.

To decide the number of clusters, different criteria can be used. Very often, the number of clusters is unknown. For example, in given clinical data, the number of clusters might be predefined by a given number of diseases. In some cases, the number of clusters can be obtained from a predetermined distance measure or difference between clusters. In general, the distance to a new object or cluster K is computed by calculating the distance from the object A and B to objects i. In this case, the weighted average linkage $d_{k_i} = (d_{A_i} + d_{B_i})/2$. The sizes of the clusters and their weights are assumed to be equal. Several other formulas exist for the

contruction of the distance matrix. The most important ones are considered now for the case of aggregating two clusters.

Single Linkage

Here the shortest distance between the opposite clusters is calculated, i.e.

$$d_{ki} = \frac{d_{A_i} + d_{B_i}}{2} - \frac{|d_{A_i} - d_{B_i}|}{2} = \min(d_{A_i}, d_{B_i})$$

As a result, clusters are formed that are loosely bound. The clusters are often linearly elongated in contrast to the usual spherical clusters. This chaining is caused by the fusion of single objects to a cluster.

3.2.3 Clustering Using K-means Algorithm

K-means algorithm. Of all clustering procedures for microarray data, K-means is among the simplest and most widely used, and has probably the cleanest probabilistic interpretation as a form of *expectation maximization* (EM) on the underlying mixture model. In a typical implementation of the K-means algorithm, the number of clusters is fixed at some value K, based, for instance on the expected number of regularity patterns. K representative points or centers are initially chosen for each cluster more or less at random. In microarray data, this could reflect, for instance, the expected number of regularity patterns. These points are also called *centroids* or *prototypes*. Then at each step, we have the following:

- Each point in the data is assigned to the cluster associated with the closest representative.

- After the assignment, new representative points are computed, for instance by computing the center of gravity of each computed cluster.

- The two procedures above are repeated until the system converges or fluctuation remains small.

We notice that using K-means clustering method requires choosing the number of clusters and also being able to compute a distance or similarity between points and compute a representative for each cluster given its members. The general idea behind K-means clustering can lead to different software implementations depending on how the initial centroids are chosen, how symmetries are broken, and so forth. A good implementation has to run the algorithm multiple times with different initial conditions and possibly also try different values of K automatically. When the cost function corresponds to an underlying probabilistic mixture model, K-means is an online approximation to the classical EM algorithm, and as such, in general is bound to converge towards a solution that is at least a local maximum likelihood or maximum posterior solution. A classical case is when Euclidean distances are used in conjunction with a mixture of Gaussian model.

3.2.4 Mixture Models and EM Algorithm

Let us consider a data set $D = (d_1, \ldots, d_N)$ and an underlying mixture model with K components of the form

$$P(d) = \sum_{k=1}^{K} P(M_k)P(d|M_k) = \sum_{k=1}^{K} \lambda_k P(d|M_k)$$

where $\lambda_k \geq 0$ and $\sum_k \lambda_k = 1$ and M_k is the model for cluster k [7]. Mixture distributions provide a flexible way of modelling complex distributions, combining together simple building blocks, such as Gaussian distributions. The Lagrangian associated with $\log$-likelihood and normalization constraints on the mixing coefficients is given by

$$\mathcal{L} = \sum_{i=1}^{N} \log \left(\sum_{k=1}^{K} \lambda_k P(d_i|M_k) \right) - \mu \left(\sum_{k=1}^{K} \lambda_k - 1 \right)$$

with the corresponding critical equation

$$\frac{\partial \mathcal{L}}{\partial \lambda_k} = \sum_{i=1}^{N} \frac{P(d_i|M_k)}{P(d_i)} - \mu = 0$$

multiplying each critical equation by λ_k and summing over k gives us the value of the Lagrange multiplier $\mu = N$.

Multiplying again the critical equation across by $P(M_k) = \lambda_k$ and using Bayes' theorem in the form

$$P(M_k|d_i) = P(d_i|M_k)\frac{P(M_k)}{P(d_i)}$$

gives

$$\lambda_k^* = \frac{1}{N} \sum_{i=1}^{N} P(M_k|d_i)$$

Therefore the maximum likelihood estimate of the mixing coefficients for class K is the sample mean of the conditional probabilities that d_i comes from the model K. Consider now that each model M_k has its own vector of parameters (w_{kj}). Differentiating the Lagrangian with respect to w_{kj} gives

$$\frac{\partial \mathcal{L}}{\partial w_{kj}} = \sum_{i=1}^{N} \frac{\lambda_k}{P(d_i)} \cdot \frac{\partial P(d_i|M_k)}{\partial w_{kj}}$$

Combining the above equations we get

$$\sum_{i=1}^{N} P(M_k|d_i) \frac{\partial \log P(d_i|M_k)}{\partial w_{kj}} = 0$$

for each k and j. The maximum likelihood equations for estimating the parameters are weighted averages of the maximum likelihood equations $\partial \log P(d_i|M_k)/\partial w_{kj} = 0$. Notice that the weights are the probabilities of membership of the d_i in each class.

4 Various Methods of Imputation of Missing Values in DNA Microarrays

4.1 The Gene Expression Matrix

In this section we will view $E \in \mathbb{R}^{n\times m}$, with $n \geq m$ as the gene expression matrix:

$$E = \begin{pmatrix} g_{11} & g_{12} & \cdots & g_{1m} \\ g_{21} & g_{22} & \cdots & g_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ g_{j1} & g_{j2} & \cdots & g_{jm} \\ \vdots & \vdots & \vdots & \vdots \\ g_{n1} & g_{n2} & \cdots & g_{nm} \end{pmatrix} = \begin{pmatrix} \mathbf{g}_1^{\mathrm{T}} \\ \mathbf{g}_2^{\mathrm{T}} \\ \vdots \\ \mathbf{g}_j^{\mathrm{T}} \\ \vdots \\ \mathbf{g}_n^{\mathrm{T}} \end{pmatrix} = [\mathbf{c}_1 \quad \mathbf{c}_2 \quad \ldots \mathbf{c}_m], \tag{4.1}$$

$$\mathbf{g}_j^{\mathrm{T}} := (g_{j1}, g_{j2}, ..., g_{jm}),\ j = 1, ..., n, \quad \mathbf{c}_i = \begin{pmatrix} g_{1i} \\ g_{2i} \\ \vdots \\ g_{ji} \\ \vdots \\ g_{ni} \end{pmatrix},\ i = 1, ..., m.$$

The row vector $\mathbf{g}_j^{\mathrm{T}}$ corresponds to the (relative) expression levels of the j^{th} gene in m experiments. The column vector $\mathbf{c}_i$ corresponds to the (relative) expression levels of the n genes in the i^{th} experiment.

Consider the SVD of the gene expression matrix $E = U\Sigma V^T$. In the terminology of [4], the columns of U are eigenarrays, the columns of V are eigengenes, and

the singular values of E are eigenexpression levels.

In many microarray data sets, researchers have found that only a few eigengenes are needed to capture the overall gene expression pattern. (Here, by a "few" we mean less than half of the number of experiments m.) The number of these *significant* eigengenes is a fundamental problem in principal component analysis [24]. Let us mention explicitly three methods to estimate the number of significant eigengenes. The *fraction* criterion can be stated simply as follows. Let

$$p_q := \frac{\sigma_q^2}{\sum_{t=1}^{m} \sigma_t^2},\ q = 1, ..., m, \quad \mathbf{p} := (p_1, ..., p_m)^{\mathrm{T}}. \tag{4.2}$$

Thus p_q represents the fraction of the expression level contributed by the q^{th} eigengene. Then we choose the l eigengenes that contribute about $70\% - 90\%$ of the total expression level. Another method is to use scree plots for the σ_q^2. (In principal component analysis, the p_q are proportional to the variances of the principal components, so we choose the principal components of maximum variability [26].) According to [24], the most consistent estimates of the number of significant eigengenes is achieved by the broken-stick model.

If E has l significant eigenvalues, we view σ_q to be effectively equal to zero for $q > l$. We define the matrix

$$E_l := \sum_{q=1}^{l} \sigma_q \mathbf{u}_q \mathbf{v}_q^{\mathrm{T}} \tag{4.3}$$

as the *filtered* part of E and consider $E - E_l$ the *noise* part of E.

Let

$$1 \geq h(\mathbf{p}) := -\frac{1}{\log m} \sum_{q=1}^{m} p_q \log p_q \geq 0. \tag{4.4}$$

Then $h(\mathbf{p})$ is the rescaled entropy of the probability vector $\mathbf{p}$. $h(\mathbf{p}) = 1$ only when $p_q = \frac{1}{m},\ q = 1, ..., m$; in other words, all the eigengenes are equally expressed. On the other hand, $h(\mathbf{p}) = 0$ if and only if $p_q(1 - p_q) = 0,\ q = 1, ..., m$ and this corresponds to $r = 1$: in other words, the gene expression is captured by a single eigengene (and eigenarray).

The following example points out a potential weakness of SVD theory in trying to detect groups of genes with similar properties.

4.2 SVD and gene clusters

Suppose the set of genes $\mathbf{g}_j^{\mathrm{T}},\ j \in [n]$ can be grouped into $k + 1$ disjoint subsets $[n] = \cup_{q=1}^{k+1} G_q$ with $G_1, ..., G_k$ nonempty and $m \geq k$ (usually $m > k$). In particular, consider the genes in each group G_q ($q = 1, ..., k$) to have similar characteristics (in other words, G_q is a cluster). Genes that have no similar characteristics are placed in G_{k+1}. Denote by $\#G_q$ the cardinality of the set G_q for $q = 1, ..., k + 1$. Suppose that our m experiments do not distinguish between any two genes belonging to the same group G_q for $q = 1, ..., k + 1$. More precisely we assume:

$$g_{ji} = a_{qi} \text{ for each } j \in G_q \text{ and } q = 1, ..., k,\ i = 1, ..., m, \tag{4.5}$$

$$g_{ji} = 0 \text{ for each } j \in G_{k+1} \text{ and } i = 1, ..., m,$$

Let $A = (a_{qi})_{q,i=1}^{k,m} \in \mathbb{R}^{k \times m}$ be the corresponding $k \times m$ matrix with the rows

$\mathbf{r}_1^{\mathrm{T}}, ..., \mathbf{r}_k^{\mathrm{T}}$:

$$A = \begin{pmatrix} \mathbf{r}_1^{\mathrm{T}} \\ \mathbf{r}_2^{\mathrm{T}} \\ \vdots \\ \mathbf{r}_k^{\mathrm{T}} \end{pmatrix}.$$

Then the row $\mathbf{r}_q$ appears exactly $\#G_q$ times in E for $q = 1, ..., k$. In addition E has $\#G_{k+1}$ zero rows. Clearly the row space of E is the row space of A. So $k \geq \operatorname{rank} E = \operatorname{rank} A$. Hence if $\operatorname{rank} A = k$ then

$$\sigma_1(E) \geq ... \geq \sigma_k(E) > \sigma_{k+1}(E) = ... = \sigma_m(E) = 0.$$

However, there is no simple formula relating the singular values of E and A. It may happen that the rows of A are linearly dependent which indicates that several groups out of $G_1, ..., G_k$ are somehow related, and the number of the significant singular values of E is less than k.

Conclusion: *The number of gene clusters is no less than the number of significant singular values of the gene expression matrix.*

4.3 Missing Data in the Gene Expression Matrix

We now consider the problem of missing data in the gene expression matrix E. (Our analysis can be applied to any matrix E.) Let $\mathcal{N} \subset [n]$ denote the set of rows of E that contain at least one missing entry. Thus for each $j \in \mathcal{N}^c := [n] \backslash \mathcal{N}$, the gene $\mathbf{g}_j^{\mathrm{T}}$ has all of its entries. Let n' denote the size of $\mathcal{N}^c$ so that the size of $\mathcal{N}$ is $n - n'$. We want to complete the missing entries of each $\mathbf{g}_j^{\mathrm{T}}$, $j \in \mathcal{N}$, under some

assumptions.

We first describe the reconstruction of the missing data in E using the SVD as given in [4].

4.3.1 Reconsideration of 4.2

Let us reconsider Example 3.1. Assume that $\operatorname{rank} A = k$. Let $j \in \mathcal{N}$ and assume that the gene j is in the cluster G_q. Then we can reconstruct all missing entries of $\mathbf{g}_j^{\mathrm{T}}$ if $G_q \backslash \mathcal{N} \neq \emptyset$. Indeed, if for some gene $p \in G_q$ we have the results of m experiments, then $\mathbf{g}_j = \mathbf{g}_p$ and we reconstructed the missing entries for $\mathbf{g}_j$. In this example we can reconstruct all the missing entries in E if E' has the same rank as E. Equivalently, we can reconstruct all the missing entries in E if the equality (4.1) holds, where l and l' are the ranks of E and E' respectively.

4.3.2 Bayesian Principal Component (BPCA):

BPCA consists of three processes [28]:

- Principal Component (PC) regression
- Bayesian estimation
- Iterative expectation-maximization (EM) algorithm.

Here we give a summary of each process.

PC regression

In order to explain PC, we need to mention principal component analysis (PCA).

Consider a $D \times N$ gene matrix Y where D represents the number of arrays (samples) and N the number of genes. Let $\boldsymbol{\mu}$ be the mean vector of $\mathbf{y}$:

$$\boldsymbol{\mu} := \left(\frac{1}{N}\right) \sum_{i=1}^{N} \mathbf{y}_i$$

Then construct the correlation matrix

$$S = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{y}_i - \boldsymbol{\mu})(\mathbf{y}_i - \boldsymbol{\mu})^T$$

Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_D$ be the eigenvalues of S and $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_D$ the corresponding eigenvectors. Then we scale the eigenvectors by their singular values. The l-th principal access vector $\mathbf{w}_l$ could be defined by $\mathbf{w}_l = \sqrt{\lambda_l}\mathbf{u}_l$ and the l-th factor score by $x_l = (\mathbf{w}_l/\lambda_l)^T \mathbf{y}$.

Then the variation of the gene expression of $\mathbf{y}$ could be represented as a linear combination of principal access vectors $\mathbf{w}_l$ ($1 \leq l \leq K$), $K < D$ (there are a few $\mathbf{w}_l$):

$$\mathbf{y} = \sum_{l=1}^{K} \mathbf{x}_l \mathbf{w}_l + \epsilon$$

This is the spirit of PC regression. We now consider the missing values in the gene expression matrix. We could estimate the missing part $\mathbf{y}^{\text{miss}}$ of gene expression $\mathbf{y}$ by estimating the of observed segment of it $\mathbf{y}^{\text{obs}}$, using PCA. Denote $\mathbf{w}_l^{\text{obs}}$ and $\mathbf{w}_l^{\text{miss}}$ be the principal axis $\mathbf{w}_l$ corresponding to observe and missing data respectively. Now construct a matrix $\boldsymbol{W} = (\boldsymbol{W}^{\text{obs}}, \boldsymbol{W}^{\text{miss}})$, where

$$\boldsymbol{W}^{\text{obs}} = (\mathbf{w}_1^{\text{obs}}, \ldots, \mathbf{w}_k^{\text{obs}}) \quad \text{and} \quad \boldsymbol{W}^{\text{miss}} = (\mathbf{w}_1^{\text{miss}}, \ldots, \mathbf{w}_k^{\text{miss}})$$

respectively.

Now we could obtain the factor score $\mathbf{x} = (x_1, \ldots, x_k)$ by minimizing the residual error $err = \|\mathbf{y}^{\text{obs}} - \mathbf{w}^{\text{obs}} \cdot \mathbf{x}\|^2$. Using the least square method,

$$\mathbf{x} = [(\boldsymbol{W}^{\text{obs}})^T \boldsymbol{W}^{\text{obs}}]^{-1} (\boldsymbol{W}^{\text{obs}})^T \mathbf{y}^{\text{obs}}$$

Then we could estimate the missing part of the gene expression $\mathbf{y}$ by $\mathbf{y}^{\text{miss}} = \mathbf{w}^{\text{miss}} \cdot \mathbf{x}$.

Bayesian estimation Probabilistic extension of Principal Component Analysis (PPCA) has found useful applications in supervised learning (tipping). In this setting the residual error and the factor scores are normally distributed.

$$\mathbf{x} \sim \mathcal{N}(\mathbf{0}, I_k) \quad \text{and}$$

$$\epsilon \sim \mathcal{N}(\mathbf{0}, \frac{1}{\tau} I_D)$$

the interesting fact is that the maximum likelihood estimation of PPCA and PCA are identical

$$\ln P(\mathbf{y}, \mathbf{x}|\boldsymbol{\theta}) = \ln P(\mathbf{y}, \mathbf{x}|\boldsymbol{W}, \boldsymbol{\mu}, \boldsymbol{\tau}) =$$
$$-\frac{\boldsymbol{\tau}}{2}\|\mathbf{y} - \boldsymbol{W}\mathbf{x} - \boldsymbol{\mu}\|^2 - \frac{1}{2}\|\mathbf{x}\|^2 + \frac{D}{2}\ln\tau - \frac{K+D}{2}\ln 2\pi$$

where $\boldsymbol{\theta} \equiv \{\boldsymbol{W}, \boldsymbol{\mu}, \boldsymbol{\tau}\}$ is the parameter set. Using Bayes theorem, we can obtain the posterior probability distribution of θ and $\boldsymbol{X}$:

$$p(\boldsymbol{\theta}, \boldsymbol{X}|\boldsymbol{Y}) \propto (\boldsymbol{Y}, \boldsymbol{X}|\boldsymbol{\theta}) p(\boldsymbol{\theta})$$

Iterative E-M algorithm

Having the information about the true parameter $\boldsymbol{\theta}_{\text{true}}$, the posterior pobability of

the missing values is given by

$$q(\boldsymbol{Y}^{\text{miss}}) = P(\boldsymbol{Y}^{\text{miss}}|\boldsymbol{Y}^{\text{obs}}, \boldsymbol{\theta}_{\text{true}}).$$

Having the parameter posterior $q(\boldsymbol{\theta})$, then

$$q(\boldsymbol{y}^{\text{miss}}) = \int d\boldsymbol{\theta} q(\boldsymbol{\theta}) p(\boldsymbol{Y}^{\text{miss}}|\boldsymbol{Y}^{\text{obs}}, \boldsymbol{\theta}).$$

Then the variation Bayes (VB) algorithm could be used to estimate the model parameter $\boldsymbol{\theta}$ and missing value $\boldsymbol{Y}^{\text{miss}}$. VB algorithm is similar to EM and it obtains the posterior distribution for $\boldsymbol{\theta}$, $\boldsymbol{Y}^{\text{miss}}$, $q(\boldsymbol{\theta})$, $q(\boldsymbol{Y}^{\text{miss}})$ by using a repetitive algorithm.

Summary of VB algorithm

1. Initialize $q(\boldsymbol{Y}^{\text{miss}})$ by replacing missing genes by the gene-wise average.
2. Estimate $q(\boldsymbol{\theta})$ using $\boldsymbol{Y}^{\text{obs}}$ and current $q(\boldsymbol{Y}^{\text{miss}})$.
3. Estimate $q(\boldsymbol{Y}^{\text{miss}})$ using current $q(\boldsymbol{\theta})$
4. Update the hyperparameter $\boldsymbol{\alpha}$ using both current $q(\boldsymbol{\theta})$ and the curent $q(\boldsymbol{Y}^{\text{miss}})$.
5. Repeat (2)-(4) until convergence.

Then we could impute the missing values in the gene expression matrix by

$$\hat{\boldsymbol{Y}^{\text{miss}}} = \int \boldsymbol{Y}^{\text{miss}} q(\boldsymbol{Y}^{\text{miss}}) d\boldsymbol{Y}^{\text{miss}}$$

4.3.3 The least square imputation method

Let E' be the $n' \times m$ matrix containing the rows $\mathbf{g}_j^{\text{T}},\ j \in \mathcal{N}^c$ of E which do not have any missing entries, and l' be the number of significant singular values of E'. Let

$\mathbf{X} \subset \mathbb{R}^m$ be the invariant subspace of the symmetric matrix $(E')^{\mathrm{T}}E'$ corresponding to the eigenvalues $\sigma_1(E')^2, ..., \sigma_{l'}(E')^2$. Let $\mathbf{x}_1, ..., \mathbf{x}_{l'}$ be the orthonormal eigenvectors of $(E')^{\mathrm{T}}E$ corresponding to the eigenvalues $\sigma_1(E')^2, ..., \sigma_{l'}(E')^2$. Then $\mathbf{x}_1, ..\mathbf{x}_{l'}$ is a basis of $\mathbf{X}$.

Let $\mathcal{M} \subset [m]$ be a subset of cardinality $m - m'$. Consider the projection $\pi_{\mathcal{M}} : \mathbb{R}^m \to \mathbb{R}^{m'}$ by deleting all the coordinates $i \in \mathcal{M}$ for any vector $\mathbf{x} = (x_1, ..., x_m)^{\mathrm{T}} \in \mathbb{R}^m$. Then $\pi_{\mathcal{M}}(\mathbf{X})$ is spanned by $\pi_{\mathcal{M}}(\mathbf{x}_1), ..., \pi_{\mathcal{M}}(\mathbf{x}_{l'})$.

Fix $j \in \mathcal{N}$ and let $\mathcal{M} \subset [m]$ be the set of experiments (columns) where the gene $\mathbf{g}_j^{\mathrm{T}}$ has missing entries. Let $\mathbf{y} \in \pi_{\mathcal{M}}(\mathbf{X})$ be the least square approximation to $\pi_{\mathcal{M}}(\mathbf{g}_j)$. Then any $\overline{\mathbf{g}}_j \in \pi_{\mathcal{M}}^{-1}(\mathbf{y})$ is a completion of $\mathbf{g}_j$. If $\pi_{\mathcal{M}}|\mathbf{X}$ is 1-1 then $\overline{\mathbf{g}}_j$ is unique. Otherwise one can choose $\overline{\mathbf{g}}_j \in \pi_{\mathcal{M}}^{-1}(\mathbf{y})$ with the least norm. Note that to find $\mathbf{y} \in \pi_{\mathcal{M}}(\mathbf{X})$ one needs to solve the least square problem for a subspace $\pi_{\mathcal{M}}(\mathbf{X})$. In principle, for each $j \in \mathcal{N}$ one solves a different least square problem. The crucial assumption of this method is

$$l = l'. \tag{4.1}$$

That is *the completed matrix E and its submatrix E' have the same number of significant singular values*. This follows from the observation that the completion of the row $\mathbf{g}_j, j \in \mathcal{N}$ lies in the subspace $\mathbf{X}$. (Note that the inequalities (4.5) imply that the assumption (4.1) can be a very restrictive assumption.)

The significant singular values of E' and of the reconstructed E are joint functions of all the rows (genes). By trying to reconstruct the missing data in each gene $\mathbf{g}_j^{\mathrm{T}}$, for $j \in \mathcal{N}$, separately, we ignore any correlation between $\mathbf{g}_j^{\mathrm{T}}$ and the genes

$\mathbf{g}_q^{\mathrm{T}},\ q \in \mathcal{N}$; consequently, this will have an impact on the singular values of E.

4.3.4 Iterative method using SVD

In the recent papers [35] and [9], the following iterative method using SVD to impute missing values in a gene expression matrix is suggested. First, replace the missing values with 0 or with values computed from another method. Call the estimated matrix E_p, where $p = 0$. Find the l_p significant singular values of E_p, and let E_{p,l_p} be the filtered part of E_p (4.3). Replace the missing values in E by the corresponding values in E_{p,l_p} to obtain the matrix E_{p+1}. Continue this process until E_p converges to a fixed matrix (within a given precision). This algorithm takes into account implicitly the influence of the estimation of one entry on the other ones. But it is not clear if the algorithm converges, nor what are the features of any fixed point(s) of this algorithm.

4.3.5 Local least squares imputation (LLSimpute)

Kim et. al. in their recent paper introduced an algorithm which is based on a least squares method. This method considers k genes which are similar to the gene with missing data. In the following, we review the essence of their method. In their method, we are concerned with two matrices A and B and a vector $\mathbf{w}$. We construct the matrix A by eliminating the elements of k nearest neighbors at q missing locations of the given genes. Then construct a matrix B by collecting all the eliminated elements. Finally we construct a row vector $\mathbf{w}$ which consists of missing values in the original gene. Then the recovery of the missing genes can be formulated as the

following least square problem:

$$\min_{\mathbf{x}} \|A^T\mathbf{x} - \mathbf{w}\|_2$$

Denote $\mathbf{u} = (\alpha_1\alpha_2 \ldots \alpha_q)^T$ of q missing values. Then $\mathbf{u}$ could be estimated as follows:

$$\mathbf{u} = B^T\mathbf{x} = B^T(A^T)^\dagger\mathbf{w}$$

where $(A^T)^\dagger$ is the pseudoinverse of A^T.

Now we use an example which is borrowed from [27] to explain the model. Suppose the target gene $\mathbf{g}$ has two missing values in the first and tenth positions in total of ten experiments. Now we use k similar genes $\mathbf{g}_{s_1}^T, \ldots, \mathbf{g}_{s_k}^T$. Now we could reconstruct the original gene and the k similar genes with the following matrices

$$\begin{pmatrix} \mathbf{g}_{s_1}^T \\ \mathbf{g}_{s_2}^T \\ \vdots \\ \mathbf{g}_{s_k}^T \end{pmatrix} = \begin{pmatrix} \alpha_1 & \mathbf{w}_1 & \mathbf{w}_2 & \cdots & \mathbf{w}_8 & \alpha_2 \\ B_{1,1} & A_{1,1} & A_{1,2} & \cdots & A_{1,8} & B_{1,2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ B_{k,1} & A_{k,1} & A_{k,2} & \cdots & A_{k,8} & B_{k,2} \end{pmatrix}$$

The values of $\mathbf{w}$ could be written as

$$\mathbf{w} \simeq \mathbf{x}_1\mathbf{a}_1 + \mathbf{x}_2\mathbf{a}_2 + \cdots + \mathbf{x}_k\mathbf{a}_k$$

Then the missing value could be estimated by

$$\alpha_1 = B_{1,1}\mathbf{x}_1 + B_{2,1}\mathbf{x}_2 + \cdots + B_{k,1}\mathbf{x}_k$$

$$\alpha_2 = B_{1,2}\mathbf{x}_1 + B_{2,2}\mathbf{x}_2 + \cdots + B_{k,2}\mathbf{x}_k$$

where α_1 and α_2 are two missing values in the original gene.

4.4 Motivation for FRAA

We suggest a new method in which the estimation of missing entries is done simultaneously, i.e., the estimation of one missing entry influences the estimation of the other missing entries. If the gene expression matrix E has missing data, we want to complete its entries to obtain a matrix $\hat{E}$, such that the rank of $\hat{E}$ is equal to (or does not exceed) d, where d is taken to be the number of significant singular values of E. The estimation of the entries of E to a matrix with a prescribed rank is a variation of the *problem of communality* (*see* [16, p. 637].) We give an optimization algorithm for finding $\hat{E}$ using the techniques for inverse eigenvalue problems discussed in [14].

It is likely that our algorithm can be used to estimate missing entries in data sets other than gene expression data. Such a data set should be represented by an $n \times m$ matrix whose rank is smaller than $\min(m, n)$.

4.4.1 Additional matrix theory background for FRAA

To compute the decomposition (1.21), it is enough to know $\mathbf{v}_q$ and $\sigma_q \mathbf{u}_q$. If σ_q repeats $k > 1$ times in the sequence $\sigma_1 \geq ... \geq \sigma_r > 0$, then the choice of the corresponding k eigenvectors $\mathbf{v}_j$ is not unique: any choice of the orthonormal basis in the eigenspace of $E^{\mathrm{T}}E$ corresponding to the eigenvalue σ_q^2 is a legitimate choice.

In what follows we will use yet another equivalent definition of the singular values of E. Let $\mathbb{R}^{n\times m}$ denote the space of all real $n \times m$ matrices and let $\mathrm{S}_m(\mathbb{R})$

denote the space of all real $m \times m$ symmetric matrices. For $A \in \mathrm{S}_m(\mathbb{R})$, we let

$$\lambda_1(A) = \lambda_1 \geq ... \geq \lambda_m(A) = \lambda_m,\ A\mathbf{z}_q = \lambda_q \mathbf{z}_q,\ \mathbf{z}_q^{\mathrm{T}}\mathbf{z}_t = \delta_{qt},\ q,t = 1,...,m, \tag{4.2}$$

be the eigenvalues and corresponding eigenvectors of A, where the eigenvalues are counted with their multiplicities, and the eigenvectors form an orthonormal basis in $\mathbb{R}^m$.

Consider the following $(n+m) \times (n+m)$ real symmetric matrix:

$$E^s := \begin{pmatrix} 0 & E \\ E^{\mathrm{T}} & 0 \end{pmatrix}. \tag{4.3}$$

It is known [22, §7.3.7]

$$\sigma_q(E) := \sigma_q = \lambda_q(E^s) = -\lambda_{n+m+1-q}(E^s),\ \text{for } q = 1,...,m, \tag{4.4}$$

$$\lambda_q(E^s) = 0 \text{ for } q = m+1,...,n.$$

The Cauchy interlacing property for E^s implies [22, §7.3.9]

Let $[n] := \{1, 2, \ldots, n\}$, and let $\mathcal{N} \subset [n],\ \mathcal{M} \subset [m]$ denote sets of cardinalities $n - n', m - m' \geq 0$ respectively.

Proposition 4.1 *Let $E \in \mathbb{R}^{n \times m}$ and denote by $E' \in \mathbb{R}^{n' \times m'}$ the matrix obtained from E by deleting all rows $i \in \mathcal{N}$ and all columns $j \in \mathcal{M}$. Then*

$$\sigma_q(E) \geq \sigma_q(E') \text{ for } q = 1,...,m, \tag{4.5}$$

$$\sigma_q(E') \geq \sigma_{q+n-n'+m-m'}(E) \text{ for } q = 1,...,m'+n'-n.$$

The significance of this proposition is explained in §4 and §5.

4.4.2 The Optimization Problem

We suggest a new method in which the estimation of missing entries is done simultaneously, i.e., the estimation of one missing entry influences the estimation of the other missing entries. If the gene expression matrix E has missing data, we want to complete its entries to obtain a matrix $\hat{E}$, such that the rank of $\hat{E}$ is equal to (or does not exceed) d, where d is taken to be the number of significant singular values of E. The estimation of the entries of E to a matrix with a prescribed rank is a variation of the *problem of communality* (*see* [16, p. 637].) We give an optimization algorithm for finding $\hat{E}$ using the techniques for inverse eigenvalue problems discussed in [14]. We now show that the estimation problem discussed in the previous section can be cast as the following optimization problem:

Problem 4.2 *Let $\mathcal{S}$ be a given subset of $[n] \times [m]$. ($\mathcal{S}$ is the set of uncorrupted entries of the gene expression matrix E given by (4.1).) Let $e(\mathcal{S}) := \{e_{ji},\ (j,i) \in \mathcal{S}\}$ be a given set of real numbers. ($e(\mathcal{S})$ is the set of uncorrupted (known) values of the entries of E.) Let $\mathrm{M}(e(\mathcal{S})) \subset \mathbb{R}^{n\times m}$ be the affine subset of all matrices $A = (a_{ji}) \in \mathbb{R}^{n\times m}$ such that $a_{ji} = e_{ji}$ for all $(j,i) \in \mathcal{S}$. ($\mathrm{M}(e(\mathcal{S}))$ all possible choices for E.) Let ℓ be a positive integer not exceeding m. Find $\hat{E} \in \mathrm{M}(e(\mathcal{S}))$ with the minimal σ_ℓ.*

Let $E = (g_{ji})$ denote the gene expression matrix with missing values. We choose the $\mathcal{S}$ in Problem 1 to be the set of coordinates (j,i) for which the entry g_{ji}

is not missing. Recall that $\mathcal{N} \subset [n]$ denotes the set of rows of E, such that each row $j \in \mathcal{N}$ contain at least one missing entry. The cardinality of $\mathcal{N}$ is $n - n'$. Thus the set $\mathcal{S}$ contains all elements $(j,1),...(j,m)$ for each $j \in \mathcal{N}^c$. The complement of $\mathcal{S}$ is the set of coordinates $\mathcal{S}^c = \{(j,i) \,|\, g_{ji} \text{ is missing}\} \subset \mathcal{N} \times [m]$. Let o denote the total number of missing entries in E. Then $o \geq n - n'$.

Let E' be the matrix as in §4.3.3 with l' significant singular values. Note that (4.5) yields $\sigma_q(E) \geq \sigma_q(E')$ for $q = 1, ..., m$. Thus if we want to complete E such that the resulting matrix still has exactly l' significant singular values, we should consider Problem 4.2 with $\ell = l' + 1$.

A more general possibility is to assume that the number of significant singular values of a possible estimation of E is $l = l' + k$ where k is a small integer, e.g. $k = 1$ or 2. That is, the group of genes $\mathbf{g}_j^{\mathrm{T}}$ for $j \in \mathbb{N}$ contributes to $l' + 1, ..., l' + k$ significant eigengenes of E. Then one considers Problem 4.2 with $\ell = l' + k + 1$.

We now consider a modification of Problem 4.2 which has a nice numerical algorithm.

Problem 4.3 *Let $\mathcal{S} \subset [n] \times [m]$ and denote by $e(\mathcal{S})$ a given set of real numbers e_{ji} for $(j,i) \in \mathcal{S}$. Let $\mathrm{M}(e(\mathcal{S})) \subset \mathbb{R}^{n \times m}$ be the affine subset of all matrices $A = (a_{ji}) \in \mathbb{R}^{n \times m}$ such that $a_{ji} = e_{ji}$ for all $(j,i) \in \mathcal{S}$. Let ℓ be a positive integer not exceeding m. Find $\hat{E} \in \mathrm{M}(e(\mathcal{S}))$ such that $\sum_{q=\ell}^{m} \sigma_q^2$ is minimal.*

Clearly, we can find $E \in \mathrm{M}(e(\mathcal{S}))$ with a "small" $\sigma_\ell^2(E)$ if and only if we can find $E \in \mathrm{M}(e(\mathcal{S}))$ with a "small" $\sum_{q=\ell}^{m} \sigma_q^2(E)$.

4.5 Fixed Rank Approximation Algorithm

4.5.1 Description of FRAA

We now describe one of the standard algorithms to solve Problem 4.3. Mathematically it is stated as follows:

Algorithm 4.4 Fixed Rank Approximation Algorithm (FRAA) *Let $E_p \in \mathrm{M}(e(\mathcal{S}))$ be the p^{th} approximation to a solution of Problem 4.3. Let $A_p := E_p^{\mathrm{T}} E_p$ and find an orthonormal set of eigenvectors for A_p, $\mathbf{v}_{p,1}, ..., \mathbf{v}_{p,m}$ as in (4.2). Then E_{p+1} is a solution to the following minimum of a convex nonnegative quadratic function*

$$\min_{E \in \mathrm{M}(e(\mathcal{S}))} \sum_{q=\ell}^{m} (E\mathbf{v}_{p,q})^{\mathrm{T}} (E\mathbf{v}_{p,q}). \tag{4.1}$$

The flow chart of this algorithm can be given as:

Fixed Rank Approximation Algorithm (FRAA)

Input: integers $m, n, L, iter$, the locations of non-missing entries $\mathcal{S}$, initial approximation E_0 of $n \times m$ matrix E.

Output: an approximation E_{iter} of E.

for $p = 0$ **to** $iter - 1$

- Compute $A_p := E_p^{\mathrm{T}} E_p$ and find an orthonormal set of eigenvectors for A_p, $\mathbf{v}_{p,1}, ..., \mathbf{v}_{p,m}$.

- E_{p+1} is a solution to the minimum problem (4.1) with $\ell = L$.

4.5.2 Explanation and justification of FRAA

We now explain the algorithm and show that in each step, we decrease the value of the function we minimize:

$$\sum_{q=\ell}^{m} \sigma_q^2(E_p) \geq \sum_{q=\ell}^{m} \sigma_q^2(E_{p+1}). \tag{4.2}$$

For any integer $k \in [m]$, let Ω_k denote the set of all k orthonormal vectors $\{\mathbf{y}_1, ..., \mathbf{y}_k\}$ in $\mathbb{R}^m$. Let A be an $m \times m$ real symmetric matrix and assume (4.2). Then the minimal principle (the Ky-Fan characterization for $-A$) is:

$$\sum_{q=\ell}^{m} \lambda_q(A) = \sum_{q=\ell}^{m} \mathbf{z}_q^{\mathrm{T}} A \mathbf{z}_q = \min_{\{\mathbf{y}_\ell, ..., \mathbf{y}_m\} \in \Omega_{m-\ell+1}} \sum_{q=\ell}^{m} \mathbf{y}_q^{\mathrm{T}} A \mathbf{y}_q. \tag{4.3}$$

See for example [14].

Let $E = E_p + X \in \mathrm{M}(e(\mathcal{S}))$. Then $X = (x_{ji})_{j,i=1}^{n,m}$ where $x_{ji} = 0$ if $(j,i) \in \mathcal{S}$ and x_{ji} is a free variable if $(j,i) \notin \mathcal{S}$.

Let $\mathbf{x} = (x_{j_1 i_1}, x_{j_2 i_2}, \ldots, x_{j_o i_o})^{\mathrm{T}}$ denote the $o \times 1$ vector whose entries are indexed by $\mathcal{S}^c$, the coordinates of the missing values in E. Then there exists a unique $o \times o$ real valued symmetric nonnegative definite matrix $o \times o$ matrix B_p which satisfies the equality

$$\mathbf{x}^{\mathrm{T}} B_p \mathbf{x} = \sum_{q=\ell}^{m} \mathbf{v}_{p,q}^{\mathrm{T}} X^{\mathrm{T}} X \mathbf{v}_{p,q}. \tag{4.4}$$

Let $F(j,i)$ be the $n \times m$ matrix with 1 in the (j,i) entry and 0 elsewhere. Then the (s,t) entry of B_p is given by

$$b_p(s,t) = \frac{1}{2} \sum_{q=\ell}^{m} \mathbf{v}_{p,q}^{\mathrm{T}} (F(j_s, i_s)^{\mathrm{T}} F(j_t, i_t) + F(j_t, i_t)^{\mathrm{T}} F(j_s, i_s)) \mathbf{v}_{p,q}, \tag{4.5}$$

$$s, t = 1, \ldots o.$$

Let $\mathcal{N} \subset [n]$. Let $\mathcal{S}(j)$ denote the set of coordinates in row j with known values in E so that $\mathcal{S}(j)^{\mathrm{c}}$ denotes the set of coordinates of the missing values in row j.

$$\mathcal{S}^c = \cup_{j\in\mathcal{N}}\mathcal{S}(j)^c,\ \mathcal{S}(j)^c = \{(j, i(j,1)), ..., (j, i(j,o_j))\}, \tag{4.6}$$

$$m \geq i(j,o_j) > ... > i(j,1) \geq 1 \quad \text{for } j \in \mathcal{N},$$

$$o := \sum_{j\in\mathcal{N}} o_j. \tag{4.7}$$

Note that the set O_j described just after (4.13) is given by $O_j := \{i(j,1), ..., i(j,o_j)\}$.

Theorem 4.5 *The $o \times o$ symmetric nonnegative definite matrix B_p given by (4.4) decomposes into a direct sum of $\#\mathcal{N} = n - n'$ symmetric nonnegative definite matrices indexed by the set $\mathcal{N}$:*

$$B_p = \oplus_{j\in\mathcal{N}} B_{p,j}, \quad B_{p,j} = (b_{p,j}(q,r))_{q,r=1}^{o_j}) \text{ is } o_j \times o_j \text{ for } j \in \mathcal{N}, \tag{4.8}$$

and

$$\mathbf{x}^{\mathrm{T}} B_p \mathbf{x} = \sum_{i\in\mathcal{N}} \mathbf{x}_j^{\mathrm{T}} B_{p,j} \mathbf{x}_j. \tag{4.9}$$

More precisely, let $\mathbf{v}_{p,k} = (v_{p,k,1}, ..., v_{p,k,m})^{\mathrm{T}},\ k = 1, ..., m$ be given as in Algorithm 4.4. Then

$$b_{p,j}(q,r) = \sum_{k=\ell}^{m} v_{p,k,i(j,q)} v_{p,k,i(j,r)}, \quad q,r = 1, ..., o_j. \tag{4.10}$$

Equivalently, let W_p be the following $m \times m$ idempotent symmetric matrix ($W_p^2 = W_p$) of rank $m - l + 1$:

$$W_p = \sum_{k=\ell}^{m} \mathbf{v}_{p,k}\mathbf{v}_{p,k}^{\mathrm{T}} = T_p T_p^{\mathrm{T}}, \quad T_p = [\mathbf{v}_{p,\ell}, ..., \mathbf{v}_{p,M}] \in \mathbb{R}^{m\times(m-\ell+1)}. \tag{4.11}$$

Then $B_{p,j}$ is the submatrix of W_p of order o_j with respect to the rows and columns in the set O_j for $j \in \mathcal{N}$. In particular, if in each row of E there is at most one missing entry then B_p is a diagonal matrix.

Proof. View the rows and the columns of B_p as indexed by $(s, i(s,q))$ and $(t, i(t,r))$ respectively, where $s,t \in \mathcal{N}$ and $q = 1, ..., o_s,\ r = 1, ..., o_t$. (For the purposes of this proof, the notation here is slightly different from that in the body of the paper.) So $B_p = (b_p((s, i(s,q)), (t, i(t,r))))$. Let $F(j,i)$ be the $n \times m$ matrix which has 1 on the (j,i) place and all other entries are equal to zero. Then

$$b_p((s, i(s,q)), (t, i(t,r))) =$$
$$\frac{1}{2} \sum_{k=\ell}^{m} \mathbf{v}_{p,k}^{\mathrm{T}} (F(s, i(s,q))^{\mathrm{T}} F(t, i(t,r)) + F(t, i(t,r))^{\mathrm{T}} F(s, i(s,q))) \mathbf{v}_{p,k}, \quad (4.12)$$
$$s,t \in \mathcal{N},\ q = 1, ..., o_s, r = 1, ..., o_t.$$

It is straightforward to show that $F(s, i(s,q))^{\mathrm{T}} F(t, i(t,r)) = 0$ if $s \neq t$. Furthermore, for $s = t$ the matrix $F(s, i(s,q))^{\mathrm{T}} F(t, i(t,r)) + F(t, i(t,r))^{\mathrm{T}} F(s, i(s,q))$ has 1 in the places $(i(s,q), i(t,r))$ and $(i(t,r), i(s,q))$ for $r \neq q$, and has 2 in the place $(i(s,q), i(s,q))$ if $r = q$ and zero in all other positions. Hence $b_p((s, i(s,q)), (t, i(t,q))) = 0$ unless $s = t$. If $s = t$ then a straightforward calculation yields (4.10). Other claims of the theorem follow straightforward from the equality (4.10). □

Observe that B_p can be decomposed into the direct sum of o symmetric non-negative definite matrices indexed by $\mathcal{N}$. Hence the function minimized in (4.1) is

given by

$$\sum_{q=\ell}^{m} \mathbf{v}_{p,q}^{\mathrm{T}} E^{\mathrm{T}} E \mathbf{v}_{p,q} = \sum_{q=\ell}^{m} \mathbf{v}_{p,q}^{\mathrm{T}} (A_p + E_p^{\mathrm{T}} X + X^{\mathrm{T}} E_p + X^{\mathrm{T}} X) \mathbf{v}_{p,q} =$$

$$\mathbf{x}^{\mathrm{T}} B_p \mathbf{x} + 2\mathbf{w}_p^{\mathrm{T}} \mathbf{x} + \sum_{q=\ell}^{m} \lambda_q(A_p) =$$

$$\sum_{i \in \mathcal{N}} (\mathbf{x}_j^{\mathrm{T}} B_{p,j} \mathbf{x}_j + 2\mathbf{w}_{p,j}^{\mathrm{T}} \mathbf{x}_j) + \sum_{q=\ell}^{m} \lambda_q(A_p), \tag{4.13}$$

where $\mathbf{w}_p := (w_{p,1}, \ldots, w_{p,o})^{\mathrm{T}}$, and

$$w_{p,t} = \sum_{q=\ell}^{m} \mathbf{v}_{p,q}^{\mathrm{T}} E_p^{\mathrm{T}} F(j_t, i_t) \mathbf{v}_{p,q}, \ t = 1, ..., o.$$

For $j \in \mathcal{N}$ the vector $\mathbf{x}_j \in \mathbb{R}^{o_j}$ contains all o_j missing entries of E in the row j of the form $x_{ji_t}, i_t \in O_j$ for the corresponding set $O_j \subset [m]$ of cardinality o_j. (See Appendix.) Since the expression in (4.1), and hence in (4.13), is always nonnegative, it follows that $\mathbf{w}_p$ is in the column space of B_p. Hence the minimum of the function given in (4.13) is achieved at the critical point

$$B_p \mathbf{x}_{p+1} = -\mathbf{w}_p, \tag{4.14}$$

and this system of equations is always solvable. (If B_p is not invertible, we find the least-squares solution).

We now show (4.2). The vector $\mathbf{x}_{p+1}$ contains the entries for the matrix X_{p+1}. Then $E_{p+1} := E_p + X_{p+1}$. From the definition of $A_{p+1} := E_{p+1}^{\mathrm{T}} E_{p+1}$ and the

minimality of $\mathbf{x}_{p+1}$ we obtain

$$\sum_{q=\ell}^{m} \sigma_q(E_p)^2 = \sum_{q=\ell}^{m} \mathbf{v}_{p,q}^{\mathrm{T}}(E_p+0)^{\mathrm{T}}(E_p+0)\mathbf{v}_{p,q} \geq$$
$$\sum_{q=\ell}^{m} \mathbf{v}_{p,q}^{\mathrm{T}}(E_p+X_{p+1})^{\mathrm{T}}(E_p+X_{p+1})\mathbf{v}_{p,q} = \sum_{q=\ell}^{m} \mathbf{v}_{p,q}^{\mathrm{T}} A_{p+1}\mathbf{v}_{p,q} \geq$$
$$\sum_{q=\ell}^{m} \lambda_q(A_{p+1}) = \sum_{q=\ell}^{m} \sigma_q(E_{p+1})^2.$$

□

We conclude this section by remarking that to solve Problem 4.2, one may use the methods of [16].

4.5.3 Algorithm for (4.14)

From Theorem 4.5, the system of equations $B_p\mathbf{x} = -\mathbf{w}_p$ in o unknowns is equivalent to $n - n'$ smaller systems

$$B_{p,j}\mathbf{x}_{p+1,j} = -\mathbf{w}_{p,j} \quad j \in \mathcal{N}. \tag{4.1}$$

Thus the big system of equations in o unknowns, the coordinates of $\mathbf{x}_{p+1}$, given (4.14) splits to $n - n'$ independent systems given in (4.1). That is, in the iterative update of the unknown entries of E given by the matrix E_{p+1}, the values in the row $j \in \mathcal{N}$ in the places $\mathcal{S}(j)^c$ are determined by the values of the entries of E_p in the places $\mathcal{S}(j)^c$ and the eigenvectors $\mathbf{v}_{p,\ell}, ..., \mathbf{v}_{p,m}$ of $E_p^{\mathrm{T}}E_p$.

We now show how to efficiently solve the system (4.14).

Algorithm 4.6 *For $j \in \mathcal{N}$ let $T_{p,j}$ is the $o_j \times (m-\ell+1)$ matrix obtained from T_p, given by (4.11), by deleting all rows except the rows $i(j,1), ..., i(j,o_j)$. Then*

(4.1) is equivalent to

$$T_{p,j}T_{p,j}^{\mathrm{T}}\mathbf{x}_{p+1,j} = -\mathbf{w}_{p,j}, \quad i \in \mathcal{N}, \tag{4.2}$$

which can be solved efficiently by the QR algorithm as follows. Write $T_{p,j}$ as $Q_{p,j}R_{p,j}P_{p,j}$, where $Q_{p,j}$ is an $o_j \times d_{p,j}$ matrix with $d_{p,j}$ orthonormal columns, $R_{p,j}$ is an upper triangular $d_{p,j} \times o_j$ matrix of rank $d_{p,j}$ nonzero rows, where the rank $V_{p,j} = d_{p,j}$, and $P_{p,j}$ is a permutation matrix. (The columns of $Q_{p,j}$ are obtained from the columns of $V_{p,j}$ using Modified Gram-Schmidt process.) Then

$$Q_{p,j}^{\mathrm{T}}\mathbf{x}_{p+1,j} = -(R_{p,j}R_{p,j}^{\mathrm{T}})^{-1}Q_{p,j}^{\mathrm{T}}\mathbf{w}_{p,j}$$

and

$$\mathbf{x}_{p+1,j} = -Q_{p,j}(R_{p,j}R_{p,j}^{\mathrm{T}})^{-1}Q_{p,j}^{\mathrm{T}}\mathbf{w}_{p,j}, \quad j \in \mathcal{N} \tag{4.3}$$

is the least square solution for $\mathbf{x}_{p+1,j}$.

4.6 Simulation

We implemented the Fixed Rank Approximation Algorithm (FRAA) in Matlab and tested it on the microarray data *Saccharomyces cerevisiae* [33] as provided at http://genome-www.stanford.edu (the elutriation data set). The dimension of the complete gene expression matrix is 5986×14. We randomly deleted a set of entries and ran FRAA on this "corrupted" matrix to obtain estimates for the deleted entries. The FRAA requires four inputs: the matrix E with N rows and M columns with missing entries, an initial guess for the missing entries, a parameter L–the number of significant singular values, and the number of iterations. We set the initial guess

to the missing data matrix with 0's replacing the missing values, the number of significant values to $L = 2$, and ran the algorithm through 5 iterations. (There was no significant change in the estimates when we replaced $L = 2$ with $L = 3$.)

We compared our estimates to estimates obtained by three other methods: replacing missing values with 0's (zeros method), row means (row means method), or the values obtained by the KNNimpute program [35]. We used a normalized root mean square as the metric for comparison: if C represents the complete matrix and E_p represents an estimate to the corrupted matrix E, then the root mean square (RMS) of the difference $D = C - E_p$ is $\frac{||D||_{\mathcal{F}}}{\sqrt{N}}$. We normalized the root mean square by dividing RMS by the average value of the entries in C.

In simulations where $1\% - 20\%$ of the entries were randomly deleted from the complete matrix C, the FRAA performed slightly better than the row means method, and significantly better than the zeros method. However, the KNNimpute algorithm (with parameters k= 15, d= 0) produced the most accurate estimates, with normalized RMS errors that were smaller than the normalized RMS errors from the other three methods. Figure 7.1 displays the results of one set of experiments estimating the elutriation matrix when each of $1, 5, 10, 15, 20\%$ of entries was removed: the normalized RMS errors are plotted against percent missing. When 25 simulations of deleting and then estimating 5% of the the entries was conducted, we found the average normalized RMS to be approximately 0.19 for KNNimpute and 0.24 for FRAA, with standard deviation to be approximately 0.02 for both methods. Not surprisingly, normalized RMS's increase with increasing percentage of missing values.

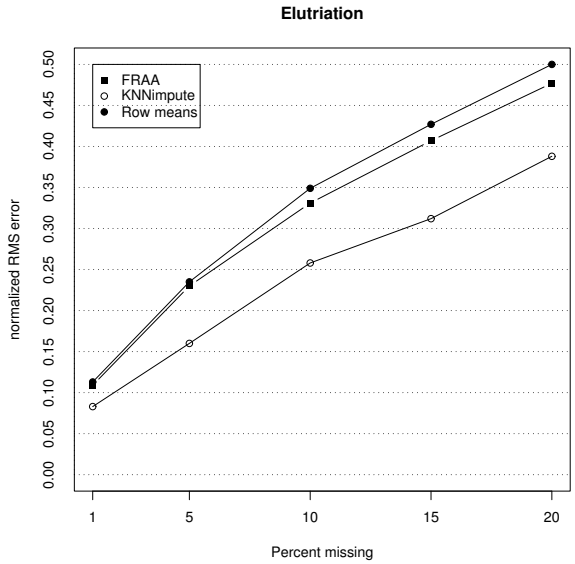

Fig. 7.1 Comparison of normalized RMS against percent missing for threee methods: FRAA, KNNimpute, and row means methods. The normalized RMS for the zeros method is not displayed, but the values are $0.397, 0.870, 1.24, 1.52, 1.76,$ for $1, 5, 10, 15, 20\%$ percent missing, respectively.

In [35], the authors caution against using KNNimpute for matrices with fewer than 6 columns. We randomly selected four columns from the elutriation data set to form a truncated data set, then randomly deleted from $1\% - 20\%$ of the entries from this newly formed matrix. Figure 7.2 gives a comparison of the normalized RMS errors against percent missing in one run of the simulation at each of the percentages. When 25 simulations at 10% missing was run, we found the average normalized RMS to be approximately 0.143 for FRAA and 0.166 for KNNimpute, with standard deviations of approximately, 0.001 and 0.003, respectively.

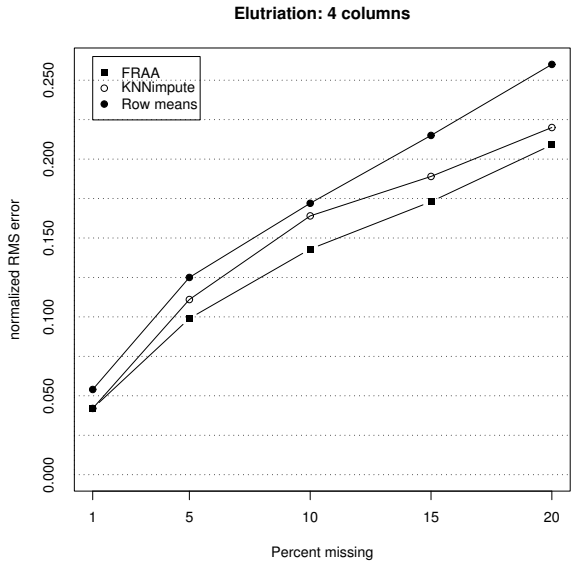

Fig. 7.2 Four columns of the full elutriation matrix were randomly selected. Entries were then randomly deleted from this truncated matrix. Plot of normalized RMS against percent missing.

For one simulation in which we randomly deleted and then estimated 10% (4200) of the entries from the full elutriation matrix,we compared the raw errors (true value - estimated value) for each of the 4200 imputed entries obtained using either KNNimpute or FRAA. Figure 7.3 shows a scatter plot of the raw errors from the estimate using KNNimpute against the raw errors from the estimate using FRAA. This plot seems to suggest that the algorithms KNNimpute and FRAA are rather consistent in how they are estimate the missing values.

Fig. 7.3 Scatter plot of the raw errors (true - estimate) of each of the 4200 imputed entries in one simulation using KNNimpute and FRAA. The correlation between the two sets of raw errors is .84.

We ran similar simulations on the Cdc15 data set available on the web, (http://genome-www.stanford.edu/SVD/htmls/spie.html), and on subsets of this data set (using 4 columns). We also ran a couple of simulations on one of the data sets included by [28]. The outcomes were similar to that using the Elutriation data set, with the FRAA algorithm outperforming KNN on the matrices with a small number of columns.

4.7 Discussion of FRAA

The Fixed Rank Approximation Algorithm uses the singular value decomposition to obtain estimates of missing values in a gene expression matrix. It uses all the known information in the matrix to simultaneously estimate all missing entries. Preliminary tests indicate that, under a normalized root mean square metric, FRAA is more accurate than replacing missing values with 0's or with row means. The

KNNimpute algorithm was more accurate when estimating missing entries deleted from the full elutriation matrix, but FRAA might be a feasible alternative in cases when the number of columns is small.

FRAA is another option, in addition to KNN, Bayesian estimations or local least squares imputations, for estimating missing values in gene expression data. FRAA by itself is a very useful tool for gene data analysis without using clustering methods. Experimental results on various data sets show that FRAA is robust. FRAA has been used by several computational biologists, who confirmed the accessibility of the algorithm.

To improve the results given by FRAA one needs to combine it with an algorithm for gene clustering. A possible implementation is as follows: First, apply FRAA to the corrupted data set; next, using this estimated data set, subdivide the genes into clusters of genes with similar traits; now apply FRAA again to the missing entries of genes in each cluster. We intend to apply these steps in a future paper.

Our final remark is that the biology of the data should guide the researcher in determining the best method to use for imputing missing values in these data sets.

4.8 IFRAA

4.8.1 Introduction

The aim of this section to introduce IFRAA, an improved version of FRAA, which improves significantly the performance of FRAA. IFRAA is a successful combination of FRAA and a good clustering algorithm. IFRAA works as follows. First we

use FRAA to find a completion G. Then we use a clustering algorithm (we used K-means here) to find a reasonable number of clusters of similar genes. For each cluster of genes we apply FRAA separately to recover the missing entries in this cluster. It turns out that this modification results in a very efficient algorithm for reconstructing the missing values of the gene expression matrix. We also note that FRAA and IFRAA are very effective in reconstructing missing values of $n \times m$ matrices, which are expected to have low effective ranks.

4.8.2 Computational comparisons of BPCA, FRAA, IFRAA and LLSimpute

For comparison of different imputation algorithms, five different types of data set were used, including two microarray gene expression data and three randomly generated synthetic data [15]. Microarray data sets were obtained from studies for the identification of cell-cycle regulated genes in yeast (*Saccharomyces cerevisiae*) [33]. The first gene expression data set is a set of complete matrix of 5986 genes and 14 experiments based on the Elutriartion data set in [33]. This data set does not have any missing value since all the genes that originally had missing values were deleted to assess the performance of the imputation algorithms. The second microarray data set is based on the Cdc15 data set in [33], which contains 5611 genes and 24 experiments. We obtained the complete data set in the same way as for first data set. Three synthetic data sets were randomly generated matrices of size 2000×20 and ranks $2, 4, 8$.

To assess the performance of missing value estimation methods, we performed

simulations where 1%, 5%, 10% and 20% of the entries were randomly deleted from the complete matrix C. Then we estimated the various completions of the missing values by BPCA, FRAA, IFRAA and LLSimpute. We used the L_2-norm version of LLSimpute. We set the K-value parameter (number of similar genes) for LLS such that there was no increase in accuracy of LLS by increasing the K-value.

We used a normalized root mean square error (NRMSE) as a metric for comparison. If C represents the complete matrix and $\hat{C}$ represents the completed matrix using an estimate to the corrupted entries in C, then the root mean square error (RMSE) is $\frac{\|D\|_{\mathcal{F}}}{\sqrt{N}}$, where $D = C - \hat{C}$. We normalized the root mean square error by dividing RMSE by the average value of the entries in C.

The random matrices of order 2000×20 and of rank $k = 2, 4, 8$ appearing in Figures 1,2 and 3 were generated as follows. One generates $2k$ random column vectors $\mathbf{x}_1, \ldots, \mathbf{x}_k \in \mathbb{R}^{2000}, \mathbf{y}_1, \ldots, \mathbf{y}_k \in \mathbb{R}^{20}$, where the entries of these vectors are chosen according to a uniform distribution in the interval $[0, 1]$. Then $C = \sum_{i=1}^{k} \mathbf{x}_i \mathbf{y}_i^{\mathrm{T}}$.

Figure 1 represents the comparisons of BPCA, FRAA and LLSimpute for 2000×20 random matrix of rank 2. In this case FRAA completely reconstructed C. The performance of IFRAA was identical to FRAA, and we did not plot the performance of IFRAA. BPCA performed excellent for 1% and its performance somewhat deteriorated with the increase of the percentage of missing data. The performance of LSSimpute was excellent for 1% of missing data, and its performance significantly deteriorated with the increase of the percentage of missing data.

Figure 2 represents the comparisons of BPCA, FRAA, IFRAA and LLSimpute

for 2000×20 random matrix of rank 4. Here BPCA and IFRAA performed extremely well. IFRAA slightly outperformed BPCA in particular in the case with 20% of missing data. FRAA performed reasonably well, but not as good as IFRAA or BPCA. The behavior of LSSimpute was similar to its performance on Figure 1.

Figure 3 represents the comparisons of BPCA, FRAA, IFRAA and LLSimpute for 2000×20 random matrix of rank 8. The behavior of BPCA, IFRAA and LLSimpute are similar to their behavior on Figure 2. In this case, however the performance of the FRAA is good for percentage of missing entries less than 20% but FRAA does not perform well when 20% of entries are missing.

Figure 4 and 5 compare BCPA, IFRAA and LLSimpute for gene expression data matrices,the Elutriation data set and the Cdc15 data set, respectively. Since some of the methods are local and others global, to be fair in comparison we first clustered the data set using the clustering algorithm, K-means, and then we performed the missing value estimation on the clustered data. In our simulation we randomly deleted $1\%, 5\%, 10\%, 15\%$, and 20% entries of data matrix. In IFRAA we chose the parameter L, which means the number of significant singular values $+1$, to be equal to 2 for Elutriation data set and 3 for Cdc15 data set. The initial guess for the missing entries in each gene was chosen to be the row average of its corresponding row. FRAA was not performing as well as the other three methods, so we did not include FRAA graphs in Figures 4 and 5.

Figure 4 depicts the comparison of BPCA, IFRAA and LLSimpute Elutriation data set in [33]. The corresponding gene expression data set is a complete matrix of $n = 5986$ genes and $m = 14$ experiments. In this case BPCA performed the best,

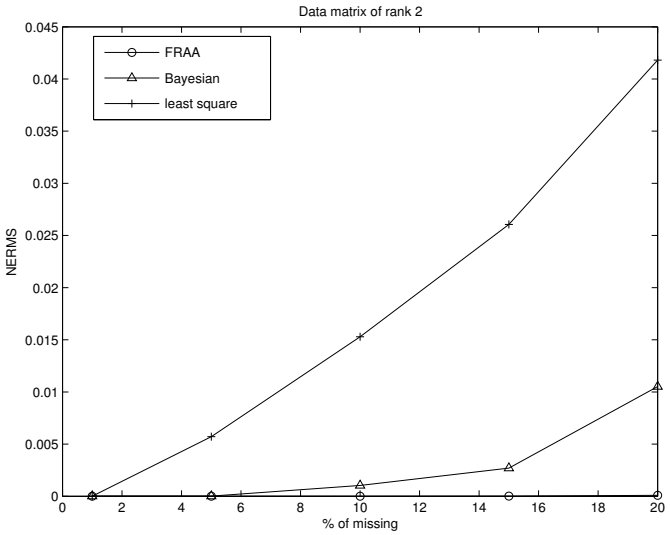

Figure 1: Comparison of NRMSE against percent of missing entries for three methods: FRAA, BPCA and LLS. Data set was a 2000×20 randomly generated matrix of rank 2.

IFRAA was slightly inferior to BPCA, and LLSimpute weaker than IFRAA.

Figure 5 depicts the comparison of BPCA, IFRAA and LLSimpute Cdc15 data set in [33] which contains 5611 genes and 24 experiments. In this case BPCA performed the best and again IFRAA performed slightly better than LLSimpute.

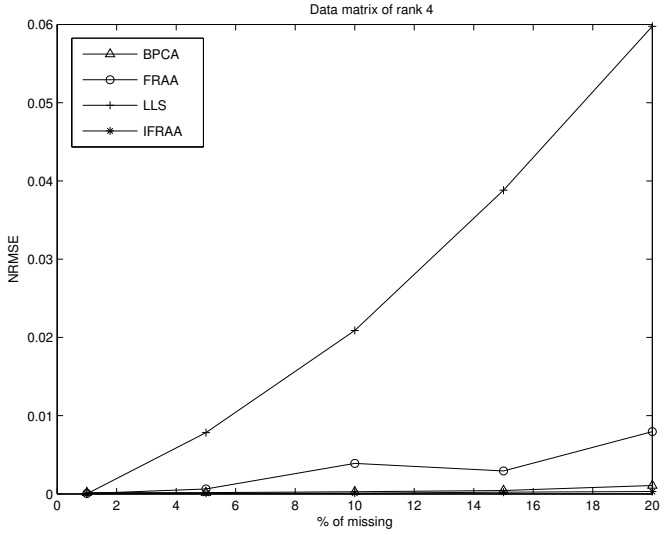

Figure 2: Comparison of NRMSE against percent of missing entries for four methods: FRAA,IFRAA, BPCA and LLS. Data set was a 2000×20 randomly generated matrix of rank 4.

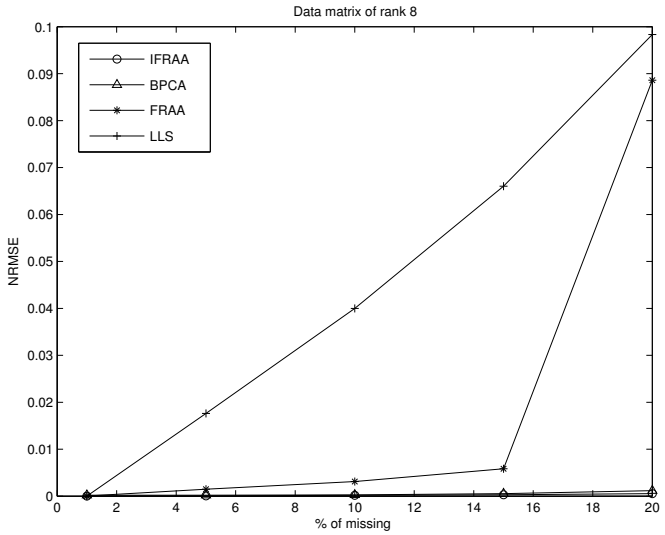

Figure 3: Comparison of NRMSE against percent of missing entries for four methods: FRAA, IFRAA, BPCA and LLS. Data set was a 2000×20 randomly generated matrix of rank 8.

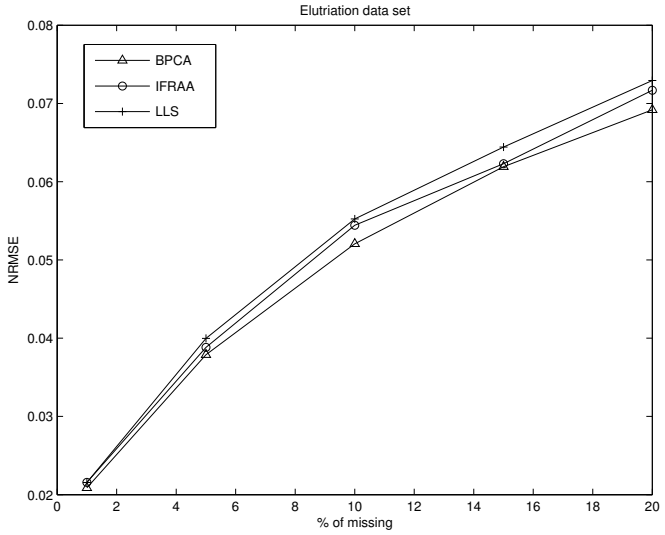

Figure 4: Comparison of NRMSE against percent of missing entries for three methods: IFRAA, BPCA and LLS. Data set was a clustered data from Elutriation data set in [33] with 14 samples.

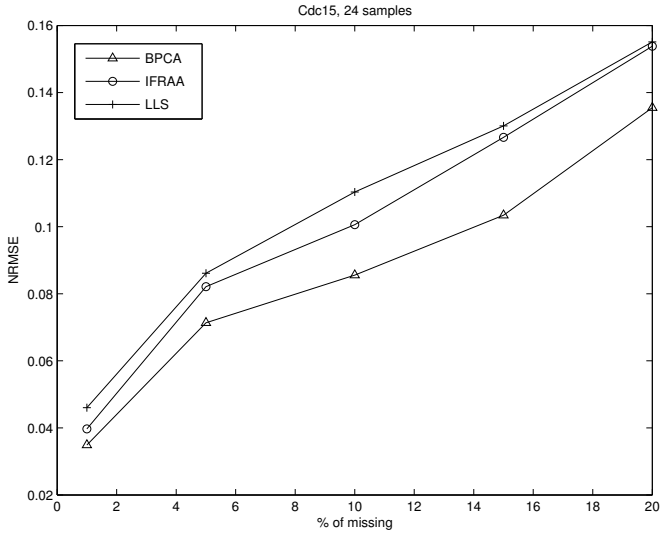

Figure 5: Comparison of NRMSE against percent of missing entries for three methods: IFRAA, BPCA and LLS. Data set was a clustered data from Cdc15 data set in [33] with 24 samples.

4.9 Conclusions

We applied the four algorithms on several data matrices including two microarray data sets. We corrupted, at random, certain percentages of these data sets and let the four algorithms recover them. We found BPCA and IFRAA to be the most reliable approaches. The LLSimpute also performed quite well. FRAA by itself was inferior to all four methods, unless the gene expression matrix has a small rank. We also applied the four algorithms to synthetic data sets, which were random 2000×20 matrices of small ranks $k = 2, 4, 8$, where we again corrupted at random certain percentages of these data sets. Not surprisingly IFRAA and BPCA were able to recover the data quite well. Where the percentage of the corrupted data was not too large, or the matrix had rank 2, FRAA was able to reconstruct the data almost perfectly. LLS-impute performed well when the data set has 1% percentage of missing entries. The performance of LSSimpute deteriorated gradually with increasing percentage of missing entries. In conclusion BPCA and IFRAA, combined with a good clustering algorithm such as K-means used here, appear to be reliable methods for recovering DNA microarray gene expression data, or any other noisy data matrix which is effectively low-rank.

These results and the results in [27] and [28] show that the performances of BPCA, IFRAA and LSSimpute depend on the structure and also the size of the data matrices and none of the above mentioned three algorithms outperforms the others in all cases.

Our studies also show that the performance of FRAA and hence IFRAA is de-

pendent on missing entries distribution. Since the performance of the FRAA depends on effective rank of the submatrix formed by rows which have no missing entries, it is important to have enough number of rows with this property. We can argue that the reason BPCA in some cases works slightly better than IFRAA is because of dependency of IFRAA on missing entries distribution. However, in a real application, when the distribution of missing entries is not uniform, contrary to what we had in our simulations, where not many rows have missing entries we expect IFRAA to perform better than BPCA. For local methods like KNN and LLS where similar genes should not have missing entries in the same column, uniform distribution of missing entries does not have any degradation effect because it is unlikely to have missing entries in the same column for similar genes.

4.10 Matlab code

```
function Ep1 = fraa(E,Ep,L,iter)
%Fixed rank algorithm -- estimate missing values
%Usage: fraa(E,Ep,L,iter)
%E: matrix with missing values
%Ep: initial solution
%L: parameter (number of significant singular values + 1)
%iter: number of iterations to perform
%Note: Any rows with all missing values must be removed
%%%%%%%%%% THIS IS  THE SET-UP
```

```
%Get size of E
    [N,M]=size(E);
       if (L > M)
        error('need L<=#columns of E ')
      end;
  %get index of missing values
   missing=find(isnan(E));
   %Number of missing values
    m=length(missing);
    m2=m*m;
%%%%%%%%%% NOW WE WORK WITH THE ALGORITHM
  Xp1=zeros(N,M);
   track=iter;
 while(iter > 0)
   A=Ep'*Ep;
   %Find singular value decomposition of A
   [U,S,V]=svd(A);
   %Singular values of Ep
   sigma2=S(S~=0);
   singular=sqrt(sigma2);
   partial_sig2=sum(sigma2(L:M));
   total_sig2=sum(sigma2(1:M));
   fprintf('\n iteration %3.0f \n', track-iter+1)
```

```
   fraction=partial_sig2/total_sig2;
   fprintf(' partial sum/total sum of sq. singular values
      \n %1.8f', fraction)
   fprintf('\n')
%Construct B=Bp
     B=sparse(m,m);  %pre-allocate space
      [is,js]=ind2sub([N,M],missing(1:m));
       for s=1:m
         for t=s:m
            if (i(s)==i(t))
            B(s,t)=sum(U(js(s),L:M)*U(js(t),L:M)');
            B(t,s)=B(s,t);    %B is symmetric
           end  %end if
        end %end For t
      end %end for s
%%%NOW CONSTRUCT THE VECTOR Wp
W=sparse(m,1); %pre-allocate space
    for t=1:m
     K=sparse(N,M);
     K(missing(t))=1;
     W(t)=sum(diag(U(:,L:M)'*Ep'*K*U(:,L:M)));
   end %end for
  %Solve Bx_(p+1)= -W
```

```
 xp1=-B\W;
%Create matrix B_{p+1}
 Xp1(missing)=xp1;
%Update solution
 Ep=Ep+Xp1;
%set counter
 iter=iter-1;
end %End while
 fprintf('\n')
 fprintf(' singular values (final iteration):\n')
 fprintf('%16.6f',singular)
 Ep1=Ep;
```

For the Matlab m file or a version of this algorithm for R, see http://people.carleton.edu/~lchihara/LMCProf.html

References

[1] C.C. Aggrawal, C.M. Procopiuc, J.L. Wolf, P.S. Yu and J.S. Park, Fast algorithms for projected clustering, *Proc. of ACM SIGMOD Intl. Conf. Management of Data* 1999, 61-72.

[2] R. Agrawal, J.Gerhrke, D.Gunopulos, and P. Raghavan, Automatic subspace clustering of high dimensional data for data mining applications, *Proc. ACM SIGMOD Conf. on Management of Data*, 1998, 94-105.

[3] U. Alon et al., Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays, *Proc. Natl. Acad. Sci. USA* 96 (1999), 6745-6750.

[4] O. Alter, P.O. Brown and D. Botstein, Processing and modelling gene expression expression data using the singular value decomposition, Proceedings SPIE, vol. 4266 (2001), 171-186.

[5] O. Alter, P.O. Brown and D. Botstein, Generalized singular decomposition for comparative analysis of genome-scale expression data sets of two different organisms, *Proc. Nat. Acad. Sci. USA* 100 (2003), 3351-3356.

[6] O. Alter, G.H. Golub, P.O. Brown and D. Botstein, Novel genome-scale correlation between DNA replication and RNA transcription during the cell cycle in yeast is predicted by data-driven models, 2004 *Miami Nature Winter Symposium*, Jan. 31 - Feb. 4, 2004.

[7] P. Baldi and G. Wesley Hatfield, *DNA Microarrays and Gene Expression*, Cambridge University Press, 2002

[8] T.H. Bo, B. Dysvik and I. Jonassen, LSimpute, Accurate estimation of missing values in microarray data with least squares methods, *Nucleic Acids Research*, 32 (2004), e34.

[9] H. Chipman, T.J. Hastie and R. Tibshirani, Clustering micrarray data In: T. Speed, (Ed.), Statistical Analysis of Gene Expression Microarray Data, , Chapman & Hall/CRC, 2003 pp. 159-200.

[10] E. Domany, Cluster Analysis of Gene Expression Data, *Journal of Statistical Physics*, 110 (2003), 1117-1139

[11] P. Drineas, A. Frieze, R. Kannan, S. Vempala and V. Vinay, Clustering large graphs via the singular value decomposition, *Journal of Machine Learning*, 56 (2004), 9-33.

[12] Petros Drineas, Eleni Drinea, Patrick S. Huggins,*An Experimental Evaluation of a Monte-Carlo Algorithm for Singular Value Decomposition*, Panhellenic Conference on Informatics 2001: 279-296

[13] M. Ester, H.-P. Krieger, J. Sander and X.Xu, A density-based algortihm for discovering clusters in large spatial databases with nose, *Proc. 2nd Intl. Conf. Knowledge Discovery and Data Mining*, 1996, 226-231.

[14] S. Friedland, Inverse eigenvalue problems, *Linear Algebra* Appl., 17 (1977), 15-51.

[15] S. Friedland, M. Kaveh, A. Niknejad, H. Zare, *An Improved Fixed Rank Approximation Algorithm for Missing Value Estimation for DNA Microarray Data*, submitted to the 2005 IEEE Symposium on Computional Intelligence in Bioinformatics and Computational Biology

[16] S. Friedland, J. Nocedal and M. Overton, The formulation and analysis of numerical methods for inverse eigenvalue problems, SIAM J. Numer. Anal. 24 (1987), 634-667.

[17] S. Friedland and A. Niknejad, Fast Monte-Carlo low rank approximations for matrices, preprint, 9 pp..

[18] S. Friedland, A. Niknejad and L. Chihara, A Simultaneous Reconstruction of Missing Data in DNA Microarrays, Linear Algebra Appl., to appear, (Institute for Mathematics and its Applications, Preprint Series, No. 1948).

[19] A. Frieze, R. Kannan and S. Vempala, Fast Monte-Carlo alogrithms for finding low rank approximations, *Proceedings of the 39th Annual Symposium on Foundation of Computer Science*, 1998.

[20] X. Gan, A.W.-C. Liew and H. Yan, Missing Microaaray Data Estimation Based on Projection onto Convex Sets Method, *Proc. 17th International Conference on Pattern Recognition*, 2004.

[21] G.H. Golub and C.F. Van Loan, *Matrix Computations*, John Hopkins Univ. Press, 1983.

[22] R.A. Horn and C.R. Johnson, *Matrix analysis*, Cambridge Univ. Press, 1987.

[23] T.R. Golub, D.K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J.P. Mesirov, H. Coller, M.L. Loh, J.R. Downing, M.A. Caligiuri, C.D. Bloomfield, and E.S. Lander, Molecular classification of cancer: class discovery and class prediction by gene expression monitoring, *Science* 286 (1999), 531-537.

[24] D.A. Jackson, Stopping rules in principal component analysis: a comparison of heuristical and statistical approaches, *Ecology* 74 (1993), 2204-2214.

[25] D. Jiang , C. Tang and A.Zhang, Cluster Analysis for Gene Expression Data: A Survey, *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, Volome 16(11), page 1370 - 1386, 2004

[26] R.A. Johnson and D. W. Wichern, Applied Multivariate Statistical Analysis, Prentice Hall, New Jersey, 4th edition (1998).

[27] 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.

[28] 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.

[29] C.C. Paige and M. A. Saunders, Towards a generalized singular value decomposition, *SIAM J. Numer. Anal.* 18 (1981), 398-405.

[30] C.M. Procopiuc, P.K. Agarwal, M. Jones and T.M. Murali, A Monte Carlo algorithm for fast projective clustering, *Proc. of ACM SIGMOD Intl. Conf. Management of Data* 2002.

[31] M.A. Shipp, K.N. Ross, P. Tamayo, A.P. Weng, J.L. Kutok, R.C. Aguiar, M. Gaasenbeek, M. Angelo, M. Reich, G.S. Pinkus *et al.*, Diffuse large B-cell lymphoma outcome prediction by gene-expression profiling nad supervised machine learning, *Nat. Med.* 8 (2002), 68-74.

[32] A. Schulze and J. Downward, *Nature Cell. Biol.* 3:190 (2001)

[33] P.T. Spellman, G. Sherlock, M.Q. Zhang, V.R. Iyer, K. Anders, M.B. Eisen, P.O. Brown, D. Botstein and B. Futcher, Comprehensive identification of cell cycle-regulated genes of the yeast *Saccharomyces cerevisiae* by microarray hybridization, *Mol. Biol. Cell*, **9** (1998), 3273-3297.

[34] G.W. Stewart, A method for computing the generalized singular value decomposition, *Matrix Pencils*, B. Kagström and A. Ruhe, *Lecture Notes in Mathematics*, 973 (1982), 207-220.

[35] 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.

[36] S. Vempala, The Random Projection Method, DIMACS Vol. 65, American Mathematical Society, 2004