Alma Mater Studiorum · Università di Bologna

Scuola di Scienze
Corso di Laurea Magistrale in Matematica

Tensor decompositions for Face Recognition

Relatore:
Chiar.ma Prof.ssa
Valeria Simoncini

Presentata da:
Domitilla Brandoni

Sessione II
Anno Accademico 2017/2018

Act as if what you do makes a difference. It does.
(William James)

A chi ha lottato al mio fianco, senza arrendersi mai.

Abstract

Automatic Face Recognition has become increasingly important in the past few years due to its several applications in daily life, such as in social media platforms and security services. Numerical linear algebra tools such as the SVD (*Singular Value Decomposition*) have been extensively used to allow machines to automatically process images in the recognition and classification contexts. On the other hand, several factors such as expression, view angle and illumination can significantly affect the image, making the processing more complex. To cope with these additional features, multilinear algebra tools, such as high-order tensors are being explored.

In this thesis we first analyze tensor calculus and tensor approximation via several different decompositions that have been recently proposed, which include HOSVD (*Higher-Order Singular Value Decomposition*) and Tensor-Train formats. A new algorithm is proposed to perform data recognition for the latter format.

Sommario

Lo sviluppo di procedure automatiche per il riconoscimento facciale è diventato sempre più importante negli ultimi anni, anche grazie alle sue numerose ricadute nella vita quotidiana, come ad esempio nei social network o nei sistemi di sicurezza. Alcune tecniche di algebra lineare, come la Decomposizione in Valori Singolari (SVD), sono state utilizzate per implementare algoritmi di riconoscimento facciale. Tuttavia alcuni fattori come l'espressione, l'angolo di visuale e l'illuminazione del viso possono condizionare negativamente il risultato dell'algoritmo. Per far fronte a tale problema, è possible utilizzare alcune tecniche di algebra multilineare. In questo lavoro di tesi sono state esaminate due tecniche di decomposizione tensoriale: HOSVD e Tensor-Train. Per quest'ultima decomposizione è stato infine proposto un nuovo algoritmo di riconoscimento facciale.

Contents

Introduction

Natural images are composed by several factors such as illumination and view angle. Human perception remains robust despite these variations. To develop automatic procedures for Face Recognition is a challenging research problem, which has become increasingly attractive in the past few years due to its several applications in daily life, such as social media platforms and security services. The aim of Face Recognition methodologies is to make this process automatic by means of a fast and reliable computational algorithm. Suppose that a huge set of images, collecting faces of different people with a variety of their expressions is available. Given a new image, the Face Recognition problem consists of detecting the closest person in the given set. In statistical terms, this procedure is called allocation, however in the image context the term *recognition* admittedly better fits the actual procedure. Multilinear algebra and the algebra of higher-order tensors offer a powerful mathematical framework for analyzing the multifactor structure of image ensembles and for addressing the huge problem of disentangling the constituent features.

In this thesis we first analyze matrix and tensor calculus and their approximations via different strategies such as SVD, HOSVD and Tensor-Train.

In the first chapter we deal with the Singular Value Decomposition describing its properties and its truncated version.

In the second chapter we analyze some basic tensor concepts such as the idea of transforming a tensor into a matrix (*unfolding*) and the concept of tensor matrix multiplication.

In the third chapter we describe two remarkable tensor decompositions: the Higher-Order Singular Value Decomposition and the Tensor-Train Decomposition.

In the last chapter we exploit the decomposition seen in the previous chapters to solve the classification problem. We analyze three different algorithms and we test them on three different datasets.

Chapter 1

Singular Value Decomposition

The Singular Value Decomposition (SVD) is a numerical technique that enables us to extend the idea of matrix diagonalization ($A = X\Lambda X^{-1}$) to any complex or real matrix. The SVD decomposes a matrix A in a product of three matrices $A = U\Sigma V^T$, where U and V are orthogonal. The most significant differences between a proper diagonalization and the SVD are the following. First of all, usually $U \neq V$. Then, U and V are orthogonal, while in general X is not.

The SVD has many useful applications in data mining, signal processing and statistics because of its endless advantages. For example, it allows to order the information contained in the matrix so that, loosely speaking, the "dominating part" becomes visible. This is due to the properties of orthogonal matrices. As a matter of fact $\|A\|_F^2 = \|\Sigma\|_F^2 = \|\operatorname{diag}(\sigma_1, \cdots, \sigma_n)\|_F^2 = \sigma_1^2 + \cdots + \sigma_n^2$.

1.1 The Decomposition

Theorem 1.1.1. *[11, Theorem 6.1] Any matrix $A \in \mathbb{R}^{m\times n}$, with $m \geq n$, can be factorized as*

$$A = U \begin{pmatrix} \Sigma \\ 0 \end{pmatrix} V^T \tag{1.1}$$

where $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are orthogonal, and $\Sigma \in \mathbb{R}^{m\times n}$ is diagonal,

$$\Sigma = \operatorname{diag}(\sigma_1, \sigma_2, \cdots, \sigma_n) = \begin{pmatrix} \sigma_1 & 0 & & \\ 0 & \sigma_2 & \ddots & \\ & \ddots & \ddots & 0 \\ & & 0 & \sigma_n \end{pmatrix}$$

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0.$$

The columns of U and V are called *left singular vectors* and *right singular vectors*, respectively, and the diagonal elements σ_i are called *singular values*. The SVD is illustrated symbolically in Figure 1.1.

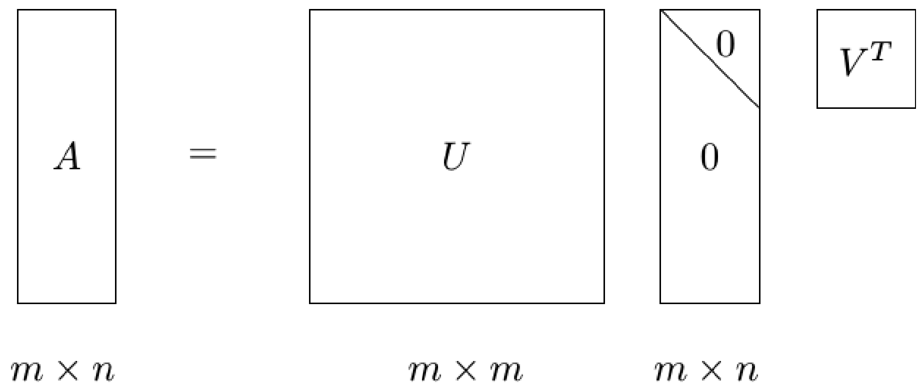


Figure 1.1: SVD [11, p. 59].

Proof. The assumption $m \geq n$ is no restriction, because if $m < n$ Theorem 1.1.1 can be applied to A^T. Consider now the following maximization problem

$$\max_{\|x\|_2=1} \|Ax\|_2 \tag{1.2}$$

Let $\hat{x} \in \mathbb{R}^n$ be the solution of (1.2), then define y such that $A\hat{x} = \sigma_1 y$, where $\|y\|_2 = 1$, $y \in \mathbb{R}^m$ and $\|A\|_2 = \sigma_1$ (by definition).
Using a linear algebra result [12, Theorem 2.5.1] we can construct the orthogonal matrices

$$U = (y\ U_2) \in \mathbb{R}^{m\times m} \qquad\qquad V = (\hat{x}\ V_2) \in \mathbb{R}^{n\times n}\ .$$

Since $y^T A\hat{x} = \sigma_1$ and $U_2^T A\hat{x} = \sigma_1 U_2^T y = 0$, then explicit computation shows that $U^T AV$ has the following structure:

$$U^T AV = \begin{pmatrix} \sigma_1 & y^T AV_2 \\ 0 & U_2^T AV_2 \end{pmatrix}.$$

Now set

$$A_1 = U^T AV = \begin{pmatrix} \sigma_1 & \omega^T \\ 0 & B \end{pmatrix}.$$

Then, since

$$\frac{1}{\sigma_1^2 + \omega^T\omega} \left\| A_1 \begin{pmatrix} \sigma_1 \\ \omega \end{pmatrix} \right\|_2^2 = \frac{1}{\sigma_1^2 + \omega^T\omega} \left\| \begin{pmatrix} \sigma_1^2 + \omega^T\omega \\ B\omega \end{pmatrix} \right\|_2^2 \geq \sigma_1^2 + \omega^T\omega$$

we have that $\|A_1\|_2 \geq \sigma_1^2 + \omega^T\omega$. But $\sigma_1^2 = \|A\|_2^2 = \|A_1\|_2^2$ and so we must have $\omega = 0$. Thus we have taken one step towards the diagonalization of A. The proof is completed by induction. $\square$

If $A = U\Sigma V^T$ is the SVD of an m by n matrix, where $m \geq n$, then

$$A = U_1\Sigma_1 V^T \tag{1.3}$$

where

$$U_1 = U(:, 1:n) = [u_1, u_2, \cdots, u_n] \in \mathbb{R}^{m\times n}$$

and

$$\Sigma_1 = \Sigma(1:n, 1:n) = diag(\sigma_1, \sigma_2, \cdots, \sigma_n) \in \mathbb{R}^{n\times n}\ .$$

This is usually called the *thin SVD*, illustrated symbolically in Figure 1.2.

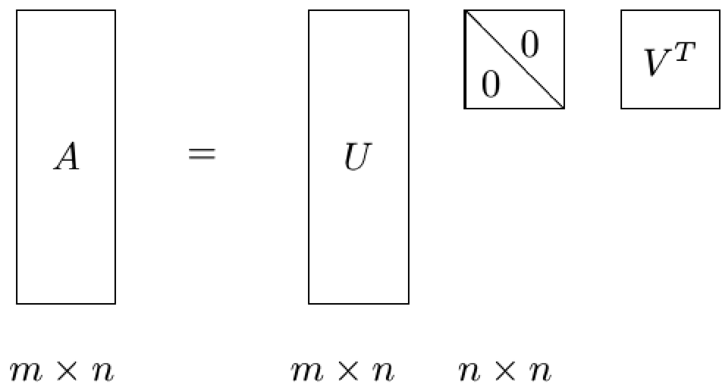


Figure 1.2: Thin SVD [11, p. 59].

The SVD of a matrix $A \in \mathbb{R}^{m\times n}$ can be thought of as a weighted, ordered sum of the matrices $u_i v_i^T$, where $u_i = U(:, i)$ and $v_i = V(:, i)$ are the i-th orthonormal columns of U and V. In particular, the matrix A can be decomposed in the following way:

$$A = \sum_{i=1}^{n} \sigma_i u_i v_i^T\ .$$

This is usually called the *outer product form* and it is derived from the thin SVD of A

$$A = U_1\Sigma_1 V^T = (u_1, u_2, \cdots, u_n)\begin{pmatrix} \sigma_1 v_1^T \\ \sigma_2 v_2^T \\ \vdots \\ \sigma_n v_n^T \end{pmatrix} = \sum_{i=1}^{n} \sigma_i u_i v_i^T.$$

The outer product form of the SVD is illustrated symbolically in Figure 1.3.

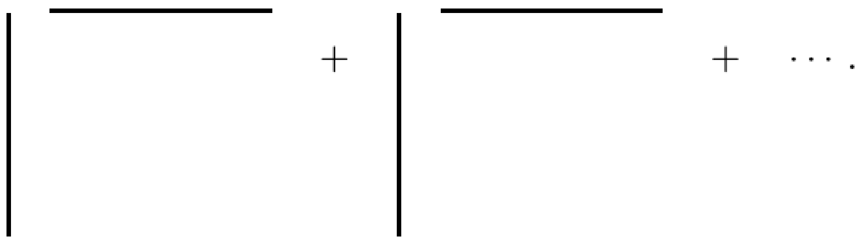


Figure 1.3: Outer product form of the SVD [11, p. 60].

1.2 Properties of SVD

The SVD of a matrix $A \in \mathbb{R}^{m \times n}$ $(m \geq n)$ $A = U\Sigma V^T$ has the following properties:

- the singular values are unique and for distinct positive singular values $\sigma_i > 0$, the $i-th$ columns of U and V are also unique up to a sign change of both columns.
- $\|A\|_F^2 := \sum_{i,j} a_{ij}^2 = \sigma_1^2 + \sigma_2^2 + \cdots + \sigma_p^2 \qquad p = \min\{m,n\};$
- $\|A\|_2^2 := \max_{0 \neq x \in \mathbb{R}^n} \frac{\|Ax\|_2^2}{\|x\|_2^2} = \max_{\|x\|=1} \|Ax\|_2^2 = \sigma_1^2;$
- $\min_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_2} = \sigma_p$, where $p = \min\{m,n\}$.
- Suppose that $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > \sigma_{r+1} = 0 = \cdots = 0 = \sigma_p$, then
    1. $\mathrm{rank}(A) = r$;
    2. $\mathrm{range}(A) := \{y \mid y = Ax \text{ for arbitrary } x\} = \mathrm{span}\{u_1, \cdots, u_r\}$;
    3. $\mathrm{null}(A) := \{x \mid Ax = 0\} = \mathrm{span}\{v_{r+1}, \cdots, v_n\}$.
- The singular values of the matrix A are equal to the square root of the eigenvalues $\lambda_1, \cdots, \lambda_m$ of the matrix A^TA.
- If $A \in \mathbb{R}^{n \times n}$ is invertible, then $A^{-1} = V\Sigma^{-1}U^T$ so that

$$A^{-1} = \sum_{i=1}^{n} \frac{1}{\sigma_i} v_i u_i^T$$

Using the SVD, it is possible to define the condition number of a matrix $A \in \mathbb{R}^{m\times n}$. Let $p = \min\{m, n\}$ and $r = \text{rank}(A)$, if $p = r$, the condition number $\kappa(A)$ is

$$\kappa(A) = \frac{\sigma_r}{\sigma_1} \ ;$$

whereas, if $p < r$

$$\kappa(A) = \frac{\sigma_p}{\sigma_1} \ .$$

1.3 The truncated SVD

One of the most interesting aspects of the SVD is that enables us to deal with the concept of matrix rank. In several applications (e.g. compression of data) one may need to determine a low-rank approximation of a matrix A. It turns out that the truncated SVD is the solution of approximation problems where one wants to approximate a given matrix by one of lower rank.

Theorem 1.3.1. *[12, Theorem 2.5.3] Let the SVD of a matrix $A \in \mathbb{R}^{m\times n}$ be given by Theorem 1.1.1. If $k < r = rank(A)$ and*

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T, \tag{1.4}$$

then

$$\min_{rank(B)=k} \|A - B\|_2 = \|A - A_k\|_2 = \sigma_{k+1} \ . \tag{1.5}$$

Proof. Since $U^T A_k V = \text{diag}(\sigma_1, \cdots, \sigma_k, 0, \cdots, 0)$, $\text{rank}(A) = k$ and $U^T(A - A_k)V = \text{diag}(0, \cdots, 0, \sigma_{k+1}, \cdots, \sigma_p)$, where $p = \min\{m, n\}$. So, since orthogonal matrices preserve the norm, $\|A - A_k\|_2 = \|U^T(A - A_k)V\|_2 = \sigma_{k+1}$.
Now suppose $\text{rank}(B) = k$ for some $B \in \mathbb{R}^{m\times n}$. It follows that we can find orthonormal vectors $x_1, \cdots, x_{n-k}$ such that $\text{null}(B) = \text{span}\{x_1, \cdots, x_{n-k}\}$. Now take $v_1, \cdots, v_{k+1}$ the first $k + 1$ columns of V. Since $\{x_1, \cdots, x_{n-k}, v_1, \cdots, v_{k+1}\}$ are $n + 1$ vectors in $\mathbb{R}^n$, $\text{span}\{x_1, \cdots, x_{n-k}\} \cap \text{span}\{v_1, \cdots, v_{k+1}\} \neq \{0\}$.

Let z be a unit 2-norm vector in this intersection. Since $Bz = 0$ and $Az = \sum_{i=1}^{k+1} \sigma_i(v_i^T z_i)u_i$, we have

$$\|A - B\|_2^2 = \|A - B\|_2^2 \|z\|_2^2 \geq \|(A - B)z\|_2^2 = \|Az\|_2^2 = \sum_{i=1}^{k+1} \sigma_i^2 (v_i^T z_i)^2 \geq \sigma_{k+1}^2$$

□

It is possible to give a similar approximation result with the Frobenius norm.

Theorem 1.3.2. *[11, Theorem 6.7] Let the SVD of a matrix $A \in \mathbb{R}^{m\times n}$ be given by the Theorem 1.1.1. If $k < r = rank(A)$ and*

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$$

then

$$\min_{rank(B)=k} \|A - B\|_F = \|A - A_k\|_F = \left(\sum_{i=k+1}^{r} \sigma_i^2 \right)^{\frac{1}{2}}$$

Proof. Consider the vector space $\mathbb{R}^{m\times n}$ with the inner product

$$\langle A, B\rangle = tr(A^T B) = \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} b_{ij},$$

and the norm $\|A\|_F = \sqrt{\langle A, A\rangle}$.
Now consider the SVD of $A = U\Sigma V^T$, where $\Sigma = (\sigma_{i,j})$. For $i \neq j$ $\sigma_{i,j} = 0$, while for $i = j$ $\sigma_{ii} = \sigma_i$. Then the matrices

$$u_i v_j^T \qquad \forall\, i = 1, \cdots, m, \quad \forall\, j = 1, \cdots, n \tag{1.6}$$

are an orthonormal basis in $\mathbb{R}^{m\times n}$; so that any $B \in \mathbb{R}^{m\times n}$ of rank k can be written in terms of (1.6) as

$$B = \sum_{i,j} \xi_{ij} u_i v_j^T, \tag{1.7}$$

where the coefficients ξ_{ij} are to be chosen. Due to the orthogonality of the basis, we have:

$$\|A - B\|_F^2 = \sum_{i,j} (\sigma_{ij} - \xi_{ij})^2 = \sum_{i} (\sigma_{ii} - \xi_{ii})^2 + \sum_{i\neq j} \xi_{ij}^2.$$

Since rank$(B) = k$ we can choose $\xi_{ij} = 0$ for $i \neq j$ in (1.7). We then have:

$$B = \sum_{i=k} \xi_{ii} u_i v_i^T.$$

To minimize the objective function, we then choose $\xi_{ii} = \sigma_{ii}$, which gives:

$$\begin{aligned}\|A - B\|_F^2 &= \sum_{i=1}^{r} (\sigma_{ii} - \xi_{ii})^2 = \sum_{i=1}^{k} (\sigma_{ii} - \xi_{ii})^2 + \sum_{i=k+1}^{r} \sigma_{ii}^2 \\ &= \sum_{i,i} (\sigma_{ii} - \sigma_{ii})^2 + \sum_{i=k+1}^{r} \sigma_{ii}^2 = \sum_{i=k+1}^{r} \sigma_{ii}^2.\end{aligned}$$

□

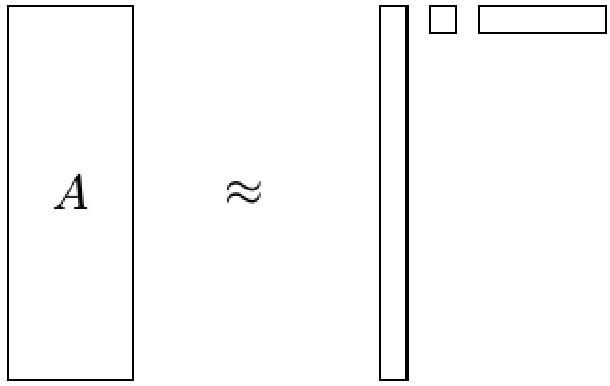


Figure 1.4: Truncated SVD [11, p. 65].

The low-rank approximation of a matrix is illustrated symbolically in Figure 1.4.

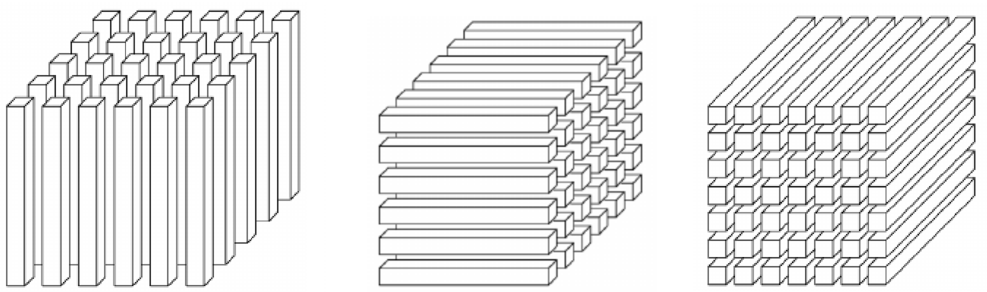
Chapter 2

Basic Tensor Concepts

In many applications data are organized in more than two categories. The corresponding mathematical objects are usually referred to as *tensors*. A tensor is a multidimensional array. More formally, an Nth-order tensor is an element of the tensor product of N vector spaces, each of which has its own coordinate system. For example, a first-order tensor is a vector and a second-order vector is a matrix.
In this chapter we present some basic tensor concepts.

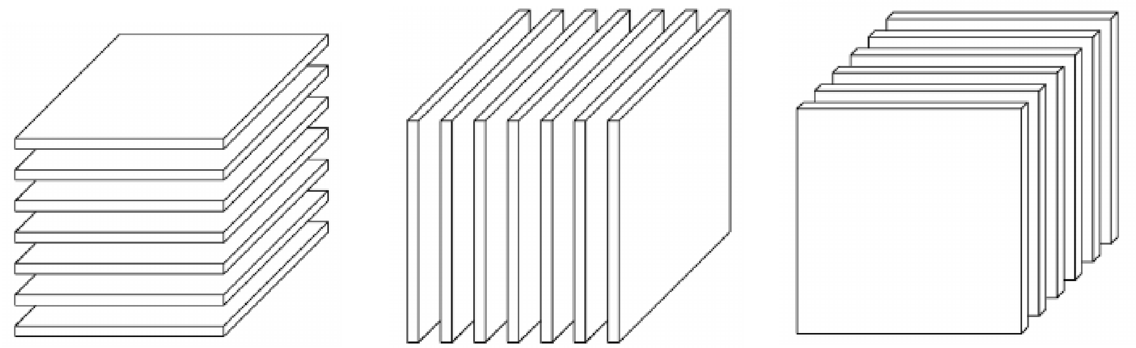
First we define the *order* of a tensor, which is the number of dimensions, also known as *ways* or *modes*. For example, a tensor $\mathcal{A} \in \mathbb{R}^{4\times2\times3}$ is a third-order tensor.

Next we define the *fibers*, which are the higher-order analogue of matrix rows and columns. A fiber is defined by fixing every index but one. A third-order tensor has column, row and tube fibers (see Figure 2.1). When extracted from the tensor, fibers are assumed to be oriented as column vectors.
Finally we define the *slices*, which are two-dimensional sections of a tensor. A slice is defined by fixing all but two indices. Figure 2.2 shows the slices of a third-order tensor, which are usually denoted by $\mathcal{A}_{i,:,:}$, $\mathcal{A}_{:,j,:}$, $\mathcal{A}_{:,:,k}$.



(a) *Mode-1 (column) fibers.* (b) *Mode-2 (row) fibers.* (c) *Mode-3 (tube) fibers.*

Figure 2.1: Fibers of a third-order tensor [18, Figure 2.1].



(a) *Horizontal slices*($\mathcal{A}_{i,:,:}$). (b) *Lateral slices* ($\mathcal{A}_{:,j,:}$). (c) *Frontal slices* ($\mathcal{A}_{:,:,k}$).

Figure 2.2: Slices of a third-order tensor [18, Figure 2.2].

Definition 2.0.1. *[18, p. 458] The inner product of two same-sized tensors $\mathcal{A}$, $\mathcal{B} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is the sum of the products of their entries, i.e.*

$$\langle \mathcal{A}, \mathcal{B} \rangle = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_n=1}^{I_N} a_{i_1,i_2,\cdots,i_n} b_{i_1,i_2,\cdots,i_n} \tag{2.1}$$

The corresponding norm is $\|A\|^2 = \langle \mathcal{A}, \mathcal{A} \rangle$.

2.1 Unfolding: transforming a tensor into a matrix

It is sometimes convenient to *unfold* a tensor into a matrix. The unfolding, also known as *matricization* or *flattening*, is the process of reordering the elements of an N-way array into a matrix. In this work we consider two different types of unfolding of a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$: the *mode-n unfolding* and the *reshape*.

2.1.1 The mode-n unfolding

The mode-n unfolding of a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is denoted by $A_{(n)}$ and arranges the mode-n fibers to be the columns of the resulting matrix. Tensor element $(i_1, i_2, \cdots, i_n)$ maps to matrix element (i_n, j), where:

$$j = 1 + \sum_{\substack{k=1,\\ k \neq n}}^{N} (i_k - 1) J_k \qquad J_k = \prod_{\substack{m=1,\\ m \neq n}}^{k-1} I_m$$

An example follows.

Example 2.1.1. *Let $\mathcal{A} \in \mathbb{R}^{4\times 2\times 3}$ be a tensor with the following frontal slices*

$$A_1 = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \\ 7 & 8 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 11 & 12 \\ 13 & 14 \\ 15 & 16 \\ 17 & 18 \end{pmatrix}, \quad A_3 = \begin{pmatrix} 21 & 22 \\ 23 & 24 \\ 25 & 26 \\ 27 & 28 \end{pmatrix}.$$

Then the three mode-n unfoldings are

$$A_{(1)} = \begin{pmatrix} 1 & 2 & 11 & 12 & 21 & 22 \\ 3 & 4 & 13 & 14 & 23 & 24 \\ 5 & 6 & 15 & 16 & 25 & 26 \\ 7 & 8 & 17 & 18 & 27 & 28 \end{pmatrix},$$

$$A_{(2)} = \begin{pmatrix} 1 & 3 & 5 & 7 & 11 & 13 & 15 & 17 & 21 & 23 & 25 & 27 \\ 2 & 4 & 6 & 8 & 12 & 14 & 16 & 18 & 22 & 24 & 26 & 28 \end{pmatrix},$$

$$A_{(3)} = \begin{pmatrix} 1 & 3 & 5 & 7 & 2 & 4 & 6 & 8 \\ 11 & 13 & 15 & 17 & 12 & 14 & 16 & 18 \\ 21 & 23 & 25 & 27 & 22 & 24 & 26 & 28 \end{pmatrix}.$$

It is possible to use different orderings of the columns for the mode-n unfolding. In general, the specific permutation of columns is not important as long as it is consistent across related calculations.

2.1.2 The reshape

The reshape of a tensor $\mathcal{A} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$ is denoted by A_k and is an $(I_1 I_2 \cdots I_k)$ by $(I_{k+1} I_{k+2} \cdots I_N)$ matrix, whose elements are taken columnwise from $\mathcal{A}$, that is

$$A_k(i_1 \cdots i_k, i_{k+1} \cdots i_N) = \mathcal{A}(i_1, \cdots, i_k, i_{k+1}, \cdots i_N). \tag{2.2}$$

Example 2.1.2. *Let $\mathcal{A}$ be the tensor defined in Example 2.1.1, then A_1 and A_2 are defined as follows*

$$A_1 = \begin{pmatrix} 1 & 2 & 11 & 12 & 21 & 22 \\ 3 & 4 & 13 & 14 & 23 & 24 \\ 5 & 6 & 15 & 16 & 25 & 26 \\ 7 & 8 & 17 & 18 & 27 & 28 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 1 & 11 & 21 \\ 3 & 13 & 23 \\ 5 & 15 & 25 \\ 7 & 17 & 27 \\ 2 & 12 & 22 \\ 4 & 14 & 24 \\ 6 & 16 & 26 \\ 8 & 18 & 28 \end{pmatrix}.$$

It is worth observing that, given a Nth-order tensor $\mathcal{A}$, it is possible to have N unfoldings and $N-1$ reshape.

2.2 The n-mode product

In this section we introduce an important tensor by matrix operation: the n-mode product.

Definition 2.2.1. *[18, p. 460] The n-mode matrix product of a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, with a matrix $U \in \mathbb{R}^{J \times I_n}$ is denoted by $\mathcal{A} \times_n U \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times J \times \cdots \times I_N}$ and its entries are given by*

$$(\mathcal{A} \times_n U)_{i_1, \cdots, i_{n-1}, j, i_{n+1}, \cdots, I_N} = \sum_{i_n=1}^{I_n} a_{i_1, i_2, \cdots, i_n, \cdots, i_N} u_{j, i_n}$$

If $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2}$ is a matrix and $U \in \mathbb{R}^{J \times I_1}$

$$\mathcal{A} \times_1 U = U\mathcal{A} \qquad (UA)(i,j) = \sum_{i_1=1}^{I_1} u_{j,i_1} a_{i_1,i}.$$

Similarly the 2-mode multiplication of $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2}$ with $V \in \mathbb{R}^{J \times I_2}$ is equivalent to matrix multiplication with V^T from the right:

$$\mathcal{A} \times_2 V = \mathcal{A}V^T \qquad (AV^T)(i,j) = \sum_{i_2=1}^{I_2} a_{i,i_2} v_{j,i_2}.$$

An important property of the n-mode multiplication is the following:

$$\mathcal{A} \times_n U \times_m V = \mathcal{A} \times_m V \times_n U \qquad \forall m \neq n,$$

where $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_n \times \cdots \times I_m \times \cdots \times I_N}$, $U \in \mathbb{R}^{J \times I_n}$ and $V \in \mathbb{R}^{J \times I_m}$. This means that for distinct modes in a series multiplication, the order of the multiplication is irrelevant. If the modes are the same, i.e. $m = n$,

$$\mathcal{A} \times_n U \times_n V = \mathcal{A} \times_n (VU).$$

This tensor matrix multiplication will be of paramount importance in the Higher-Order Singular Value Decomposition (HOSVD). As a matter of fact, the HOSVD of a tensor $A \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ will be defined by looking for orthogonal coordinate transformations of $\mathbb{R}^{I_1}$, $\mathbb{R}^{I_2}$, $\cdots$, $\mathbb{R}^{I_N}$. To do this, the $n - mode$ product will be used.

2.3 Rank properties

The definition of tensor rank was first introduced by Hitchcock in 1927 and it is analogue to the definition of matrix rank, even if the properties of tensor and matrix ranks are quite different.

Definition 2.3.1. *[18, p. 458] An N-th order tensor $\mathcal{A}$ has rank 1 when it can be written as the outer product of N vectors $x^{(1)}$, $x^{(2)}, \ldots, x^{(N)}$, i.e.*

$$\mathcal{A} = x^{(1)} \circ x^{(2)} \circ \cdots \circ x^{(N)}.$$

The symbol "$\circ$" represents the vector outer product. This means that each element of the tensor is the product of the corresponding vector elements:

$$\mathcal{A}(i_1, i_2, \cdots, i_N) = x^{(1)}_{i_1} x^{(2)}_{i_2} \cdots x^{(N)}_{i_N}, \qquad \forall\, 1 \leq i_n \leq I_n.$$

Definition 2.3.2. *[10, Definition 2] The n-rank of $\mathcal{A}$ denoted by $R_n = \mathrm{rank}_n(\mathcal{A})$ is the dimension of the vector space spanned by the n-mode vectors.*

An important property of the n-rank is the following:

$$\mathrm{rank}_n(\mathcal{A}) = \mathrm{rank}(A_{(n)}). \tag{2.3}$$

Definition 2.3.3. *[10, Definition 4] The rank of an arbitrary N-th order tensor $\mathcal{A}$, denoted by $R = \mathrm{rank}(\mathcal{A})$ is the minimal number of rank-one tensors that yield $\mathcal{A}$ as a linear combination.*

One of the main differences between tensor rank and matrix rank is that the rank of a real-valued tensor may be different over $\mathbb{R}$ and $\mathbb{C}$ [18, Section 3.1]. A second difference is that, except in special cases, there is no straightforward algorithm to determine the rank of a specific given tensor; in fact the problem is NP-hard.

Chapter 3

Tensor Decompositions

In this chapter we present two different Tensor Decomposition techniques: HOSVD (High-Order Singular Value Decomposition), which is a generalization of the matrix SVD to N-mode tensors and Tensor-Train Decomposition, which decomposes an N dimensional tensor in a product of 3-dimensional tensors.

3.1 The Higher-Order Singular Value Decomposition

The HOSVD, also known as Tucker decomposition, was first introduced by Tucker in 1963. It decomposes a tensor into a core tensor multiplied by an orthogonal matrix along each mode.

Theorem 3.1.1. *[10, Theorem 2] Every $(I_1 \times I_2 \times \cdots \times I_N)$-tensor $\mathcal{A}$ can be factorized as*

$$\mathcal{A} = \mathcal{S} \times_1 U^{(1)} \times_2 U^{(2)} \times_3 \cdots \times_N U^{(N)} \tag{3.1}$$

where:

- $U^{(n)} = (U_1^{(n)}\, U_2^{(n)} \, \cdots \, U_{I_n}^{(n)})$ *are orthogonal matrices $I_n \times I_n$ and the vector $U_i^{(n)}$ is the i-th n-mode singular vector;*
- $\mathcal{S}$*, called* core tensor*, is a complex $(I_1 \times I_2 \times \cdots \times I_N)$-tensor, whose subtensors $\mathcal{S}_{i_n=\alpha}$, obtained by fixing the nth index equal to α, have the properties of*
    1. all-orthogonality*: two subtensors $\mathcal{S}_{i_n=\alpha}$ and $\mathcal{S}_{i_n=\beta}$ are orthogonal for all possible values of n in the sense of the scalar product (2.1)*

    $$\langle \mathcal{S}_{i_n=\alpha}, \mathcal{S}_{i_n=\beta} \rangle = 0 \qquad \forall\, \alpha \neq \beta; \tag{3.2}$$

    2. ordering*:*

    $$\|\mathcal{S}_{i_n=1}\| \geq \|\mathcal{S}_{i_n=2}\| \geq \cdots \geq \|\mathcal{S}_{i_n=I_n}\| \geq 0 \qquad \forall\, n = 1, \cdots, N. \tag{3.3}$$

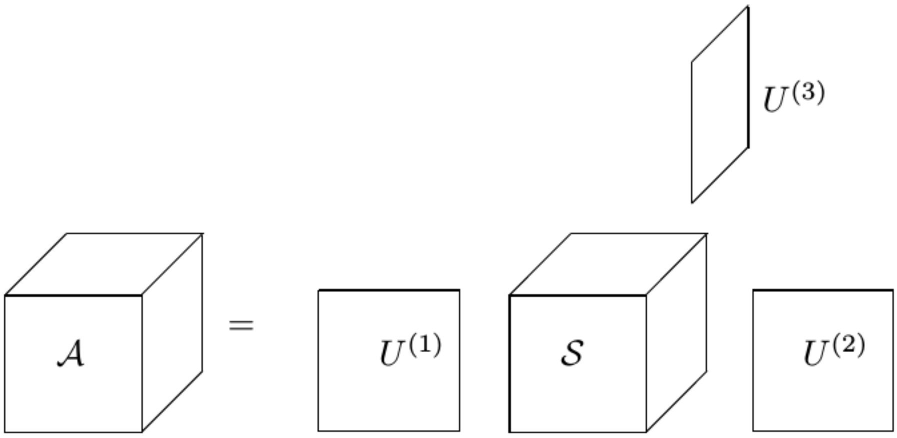


Figure 3.1: Visualization of the HOSVD [11, Figure 8.2].

This means that, if the n-mode singular values are defined as

$$\sigma_i^{(n)} = \|\mathcal{S}_{i_n=i}\|,$$

then

$$\sigma_1^{(n)} \geq \sigma_2^{(n)} \geq \cdots \geq \sigma_{I_n}^{(n)} \geq 0 \qquad \forall \;\; n = 1, \cdots, N.$$

Proof. The proof shows the strong connection between the HOSVD of $\mathcal{A}$ and the SVD of its matrix unfoldings. The derivation is given in terms of real-valued tensors. Consider the 1-mode unfolding of $\mathcal{A}$, $A_{(1)}$ and its SVD

$$A_{(1)} = U^{(1)}\Sigma^{(1)}\left(V^{(1)}\right)^T,$$

in which $\Sigma^{(1)} = \mathrm{diag}(\sigma_1^{(1)}, \sigma_2^{(1)}, \cdots \sigma_{I_1}^{(1)})$, where $\sigma_1^{(1)} \geq \sigma_2^{(1)} \geq \cdots \geq \sigma_{I_1}^{(1)} \geq 0$. This can be done for every $n = 1, \cdots, N$. So we have found the orthogonal matrices $U^{(1)}, U^{(2)}, \cdots, U^{(N)}$. It can be shown that, if we take $\mathcal{S}$ such that

$$\mathcal{S} = \mathcal{A} \times_1 \left(U^{(1)}\right)^T \times_2 \left(U^{(2)}\right)^T \times_3 \cdots \times_N \left(U^{(N)}\right)^T,$$

then (3.2) and (3.3) are satisfied. □

The HOSVD is visualized for a third order tensor in Figure 3.1. The proof of Theorem 3.1.1 enables us to write Algorithm 1.

We now illustrate the HOSVD of a third-order tensor in Example 3.1.1.

Algorithm 1 HOSVD.

Require: Tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$
1: **for** $n = 1, \cdots, N$ **do**
2: Consider the unfolding $A_{(n)}$ of $\mathcal{A}$
3: compute the SVD of $A_{(n)}$, $A_{(n)} = USV^T$;
4: Set the n-th orthogonal matrix of the HOSVD $U^{(n)} = U$.
5: **end for**

Example 3.1.1. *Let $\mathcal{A} \in \mathbb{R}^{4\times 2\times 3}$ be a tensor with the following frontal slices*

$$A_1 = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \\ 7 & 8 \end{pmatrix} \quad A_2 = \begin{pmatrix} 11 & 12 \\ 13 & 14 \\ 15 & 16 \\ 17 & 18 \end{pmatrix} \quad A_3 = \begin{pmatrix} 21 & 22 \\ 23 & 24 \\ 25 & 26 \\ 27 & 28 \end{pmatrix}.$$

Applied to $\mathcal{A}$, Theorem 3.1.1 says that it is possible to find $\mathcal{S} \in \mathbb{R}^{4\times 2\times 3}$, $U^{(1)} \in \mathbb{R}^{4\times 4}$, $U^{(2)} \in \mathbb{R}^{2\times 2}$ and $U^{(3)} \in \mathbb{R}^{3\times 3}$ such that

$$\mathcal{A} = \mathcal{S} \times_1 U^{(1)} \times_2 U^{(2)} \times_3 U^{(3)},$$

where

$$U^{(1)} = \begin{pmatrix} -0.4183 & -0.7246 & 0.5461 & 0.0423 \\ -0.4705 & -0.2804 & -0.7596 & 0.3507 \\ -0.5227 & 0.1637 & -0.1190 & -0.8281 \\ -0.5749 & 0.6079 & 0.3326 & 0.4352 \end{pmatrix},$$

$$U^{(2)} = \begin{pmatrix} -0.6887 & -0.7251 \\ -0.7251 & 0.6887 \end{pmatrix},$$

$$U^{(3)} = \begin{pmatrix} -0.1633 & 0.8981 & 0.4082 \\ -0.5053 & 0.2792 & -0.8165 \\ -0.8473 & -0.3397 & 0.4082 \end{pmatrix},$$

and $\mathcal{S}$ has the following frontal slices

$$S_1 = \begin{pmatrix} -82.1099 & 0.0179 \\ 0.0041 & 0.2503 \\ 0.0000 & 0.0000 \\ -0.0000 & 0.0000 \end{pmatrix} S_2 = \begin{pmatrix} -0.0014 & -1.1770 \\ -5.3330 & -0.2755 \\ -0.0000 & 0.0000 \\ 0.0000 & -0.0000 \end{pmatrix}$$

$$S_3 = 10^{-14} \begin{pmatrix} 0.1563 & 0.1290 \\ -0.0267 & -0.0736 \\ -0.0230 & -0.1032 \\ 0.0007 & -0.0256 \end{pmatrix}.$$

Equation (3.1) can also be written elementwise as

$$\mathcal{A}(j_1, j_2, \cdots, j_N) = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} \mathcal{S}(i_1, i_2, \cdots, i_N) U^{(1)}_{j_1,i_1} U^{(2)}_{j_2,i_2} \cdots U^{(N)}_{j_N,i_N},$$

which has the following interpretation: the element $\mathcal{S}(i_1, i_2, \cdots, i_N)$ reflects the variation by the combination of the singular vectors $U^{(1)}_{i_1}$, $U^{(2)}_{i_2}, \cdots, U^{(N)}_{i_N}$.
It is also possible to express the HOSVD as an expansion of mutually orthogonal rank-one tensors

$$\mathcal{A} = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} \mathcal{S}(i_1, i_2, \cdots, i_N) U^{(1)}_{i_1} U^{(2)}_{i_2} \cdots U^{(N)}_{i_N}. \tag{3.4}$$

Figure 3.2 shows the decomposition (3.4) for a third-order tensor.

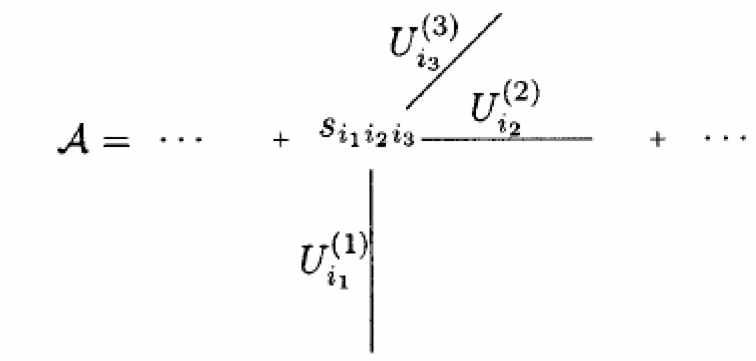


Figure 3.2: Visualization of a triadic decomposition [10, Figure 5].

In several applications it may happen that the dimension of one mode is larger than the product of the dimensions of the other modes. For example, consider a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ with $I_1 > I_2 I_3 \cdots I_N$. It can be shown that the core tensor satisfies the following equation

$$\mathcal{S}_{i_1=\alpha} = 0 \qquad \alpha > I_2 I_3 \cdots I_N,$$

and we can omit the zero part of the core tensor and rewrite (3.1) as a *thin HOSVD*,

$$\mathcal{A} = \tilde{\mathcal{S}} \times_1 \tilde{U}^{(1)} \times_2 U^{(2)} \times_3 \cdots \times_N U^{(N)} \tag{3.5}$$

where $\tilde{\mathcal{S}} \in \mathbb{R}^{I_2 I_3 \cdots I_N \times I_2 \times \cdots \times I_N}$ and $\tilde{U}^{(1)} \in \mathbb{R}^{I_1 \times I_2 I_3 \cdots I_N}$.

3.1.1 Properties of HOSVD

Many properties of the SVD have a higher-order counterpart. Consider $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, its HOSVD has the following properties:

1. the n-mode singular values are unique, and for distinct positive singular values the n-mode singular vectors are also unique up to a sign change.
2. Suppose that $\sigma_1^{(n)} \geq \sigma_2^{(n)} \geq \cdots \geq \sigma_{R_n}^{(n)} > \sigma_{R_{n+1}}^{(n)} = 0 = \cdots = 0 = \sigma_{I_n}^{(n)}$ for all $n = 1, \cdots, N$. Then, by observing that $\|\mathcal{A}\| = \|A_{(n)}\|$ which equals $\|\mathcal{S}\|^2 = \sum_{i_1=1}^{R_1} \|S_{i_1=1}\|^2$, we have that

$$\|\mathcal{A}\|^2 = \sum_{i_1=1}^{R_1} \left(\sigma_{i_1}^{(1)}\right)^2 = \sum_{i_2=1}^{R_2} \left(\sigma_{i_2}^{(2)}\right)^2 = \cdots = \sum_{i_N=1}^{R_N} \left(\sigma_{i_N}^{(N)}\right)^2 = \|\mathcal{S}\|^2.$$

3. Suppose that $\sigma_1^{(n)} \geq \sigma_2^{(n)} \cdots \geq \sigma_{R_n}^{(n)} > \sigma_{R_{n+1}}^{(n)} = 0 = \cdots = 0 = \sigma_{I_n}^{(n)}$ for all $n = 1, \cdots, N$. Then
   (a) the n-mode vector space range$(A_{(n)})$ satisfies

   $$\mathrm{range}(A_{(n)}) = \mathrm{span}(U_1^{(n)}, U_2^{(n)}, \cdots, U_{R_n}^{(n)}),$$

   where $U_j^{(n)}$ for $j = 1, \cdots, R_n$ are column vectors of $U^{(n)}$.
   (b) The left n-mode null space null$(A_{(n)}^T)$ satisfies

   $$\mathrm{null}(A_{(n)}^T) = \mathrm{span}(U_{R_{n+1}}^{(n)}, U_{R_{n+2}}^{(n)}, \cdots, U_{I_n}^{(n)}).$$

4. Let r_n be equal to the highest index for which $\|\mathcal{S}_{i_n=r_n}\| > 0$ in (3.3). Then we have

$$R_n = \mathrm{rank}_n(\mathcal{A}) = r_n.$$

It is also possible to notice that the HOSVD is a true generalization of the matrix SVD, in the sense that, when Theorem 3.1.1 is applied to matrices, it leads to the classical matrix SVD.

3.1.2 The truncated HOSVD

The truncated SVD of a matrix is the best approximation, in a least square sense, of the matrix itself by one of lower rank. Unfortunately, this property has not a high-order equivalent. Consider a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, by discarding the smallest n-mode singular values, we obtain a tensor $\hat{\mathcal{A}}$ with a column rank equal to I_1', row rank equal to I_2', etc. However, this tensor in general is not the best approximation under the given n-mode rank constraints. Nevertheless, due to the ordering assumptions for the n-mode singular values, $\hat{\mathcal{A}}$ is still a good approximation of $\mathcal{A}$.

Theorem 3.1.2. *[10, Property 10] Consider $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and its HOSVD (as in Theorem 3.1.1) and let the n-mode rank of $\mathcal{A}$ be equal to R_n $(1 \leq n \leq N)$. Define a tensor $\hat{\mathcal{A}}$ by discarding the n smallest n-mode singular values $\sigma^{(n)}_{I_n'+1}, \sigma^{(n)}_{I_n'+2}, \cdots, \sigma^{(n)}_{R_n}$ for given values of I_n', i.e., set the corresponding parts of $\mathcal{S}$ equal to zero. Then we have*

$$\|\mathcal{A} - \hat{\mathcal{A}}\|^2 \leq \sum_{i_1=I_1'+1}^{R_1} \left(\sigma^{(1)}_{i_1}\right)^2 + \sum_{i_2=I_2'+1}^{R_2} \left(\sigma^{(2)}_{i_2}\right)^2 + \cdots + \sum_{i_N=I_N'+1}^{R_N} \left(\sigma^{(N)}_{i_N}\right)^2.$$

Proof. It holds

$$\begin{aligned}
\|\mathcal{A} - \hat{\mathcal{A}}\|^2 &= \|\mathcal{S} - \hat{\mathcal{S}}\|^2 \\
&= \sum_{i_1=1}^{R_1}\sum_{i_2=1}^{R_2}\cdots\sum_{i_N=1}^{R_N} s^2_{i_1,i_2,\cdots,i_N} - \sum_{i_1=1}^{I_1'}\sum_{i_2=1}^{I_2'}\cdots\sum_{i_N=1}^{I_N'} s^2_{i_1,i_2,\cdots,i_N} \\
&= \sum_{i_1=I_1'+1}^{R_1}\sum_{i_2=I_2'+1}^{R_2}\cdots\sum_{i_N=I_N'+1}^{R_N} s^2_{i_1,i_2,\cdots,i_N} \\
&\leq \sum_{i_1=I_1'+1}^{R_1}\sum_{i_2=1}^{R_2}\cdots\sum_{i_N=1}^{R_N} s^2_{i_1,i_2,\cdots,i_N} + \sum_{i_1=1}^{R_1}\sum_{i_2=I_2'+1}^{R_2}\cdots\sum_{i_N=1}^{R_N} s^2_{i_1,i_2,\cdots,i_N} + \cdots \\
&\quad + \sum_{i_1=1}^{R_1}\sum_{i_2=1}^{R_2}\cdots\sum_{i_N=I_N'+1}^{R_N} s^2_{i_1,i_2,\cdots,i_N} \\
&= \sum_{i_1=I_1'+1}^{R_1} \left(\sigma^{(1)}_{i_1}\right)^2 + \sum_{i_2=I_2'+1}^{R_2} \left(\sigma^{(2)}_{i_2}\right)^2 + \cdots + \sum_{i_N=I_N'+1}^{R_N} \left(\sigma^{(N)}_{i_N}\right)^2.
\end{aligned}$$

□

Example 3.1.2. *[10, Example 4] Consider the tensor $\mathcal{A}$, whose unfolding along the first mode is given by*

$$A_1 = \left(\begin{array}{ccc|ccc|ccc} 0.9073 & 0.7158 & -0.3698 & 1.7842 & 1.6970 & 0.0151 & 2.136 & -0.0740 & 1.4429 \\ 0.8924 & -0.4898 & 2.4288 & 1.7753 & -1.5077 & 4.0337 & -0.6631 & 1.9103 & -1.7495 \\ 2.1488 & 0.3054 & 2.3753 & 4.2495 & 0.3207 & 4.7146 & 1.8260 & 2.1335 & -0.2716 \end{array}\right).$$

Its HOSVD is given by

$$\mathcal{A} = \mathcal{S} \times_1 U^{(1)} \times_2 U^{(2)} \times_3 U^{(3)},$$

where

$$U^{(1)} = \begin{pmatrix} -0.1121 & 0.7739 & -0.6233 \\ -0.5771 & -0.5613 & -0.5932 \\ -0.8090 & 0.2932 & 0.5095 \end{pmatrix},$$

$$U^{(2)} = \begin{pmatrix} -0.6208 & -0.4986 & -0.6050 \\ 0.0575 & -0.7986 & 0.5992 \\ -0.7818 & 0.3372 & 0.5244 \end{pmatrix},$$

$$U^{(3)} = \begin{pmatrix} -0.4624 & 0.0102 & 0.8866 \\ -0.8866 & -0.0135 & -0.4623 \\ 0.0072 & -0.9999 & 0.0152 \end{pmatrix}$$

and $\mathcal{S}$, whose unfolding along the first mode is

$$S_1 = \left(\begin{array}{ccc|ccc|ccc} -8.7088 & 0.0489 & 0.2797 & 0.1066 & -3.2737 & 0.3223 & -0.0033 & 0.1797 & -0.2223 \\ 0.0256 & 3.2546 & 0.2854 & 3.1965 & 0.2130 & 0.7829 & 0.2948 & 0.0378 & -0.3704 \\ -0.0000 & -0.0000 & -0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \end{array}\right).$$

Now, discarding $\sigma_3^{(2)}$ and $\sigma_3^{(3)}$, i.e. replacing $\mathcal{S}$ with $\hat{\mathcal{S}}$ having the following unfolding

$$\hat{S}_1 = \left(\begin{array}{ccc|ccc|ccc} -8.7088 & 0.0489 & 0.0000 & 0.1066 & -3.2737 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0256 & 3.2546 & 0.0000 & 3.1965 & 0.2130 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ -0.0000 & -0.0000 & -0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \end{array}\right),$$

we obtain an approximation $\hat{\mathcal{A}}$, for which

$$1.1838 = \|\mathcal{A} - \hat{\mathcal{A}}\|^2 \leq \left(\sigma_3^{(2)}\right)^2 + \left(\sigma_3^{(3)}\right)^2 = 1.3704.$$

3.2 Tensor-Train Decomposition

Consider the tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and its HOSVD. The need for storing the core tensor $\mathcal{S}$ renders the HOSVD increasingly unattractive as N gets larger. This has motivated the search for decompositions that do not suffer the curse of dimensionality, while preserving the good properties of the HOSVD: closeness and SVD-based compression. One good candidate for such a decomposition is the *Tensor-Train Decomposition*, which decomposes an N- dimensional tensor in a product of 3-dimensional tensors.

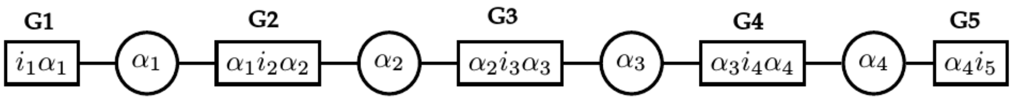


Figure 3.3: Visualization of the Tensor-Train Decomposition [20, Figure 1.1].

3.2.1 The decomposition

Let $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ be an N-dimensional tensor. Its Tensor-Train Decomposition has the following form

$$\mathcal{A}(i_1, \cdots, i_N) = G_1(:, i_1, :)G_2(:, i_2, :) \cdots G_N(:, i_N, :) = G_1(i_1)G_2(i_2) \cdots G_N(i_N) \quad (3.6)$$

where

- $G_k \in \mathbb{R}^{R_{k-1} \times I_k \times R_k}$, for $k = 1, \cdots, N$ are called *TT-cores*,
- $R_k = \text{rank}(A_k)$, for $k = 1, \cdots, N$ are called *TT-ranks*,
- $R = \max_{1 \leq k \leq N} R_k$ is called *maximal TT-rank*.

It is important to notice that, since $\mathcal{A}(i_1, \cdots, i_N)$ is a scalar, we have the boundary conditions $R_0 = R_N = 1$, so G_1 and G_{N-1} can be considered both as tensors and as matrices.
In index form, the expression in (3.6) can be equivalently written as

$$\mathcal{A}(i_1, \cdots, i_N) = \sum_{\alpha_0, \alpha_1, \cdots, \alpha_N} G_1(\alpha_0, i_1, \alpha_1) \cdots G_N(\alpha_{N-1}, i_N, \alpha_N)$$

Definition 3.2.1. *[20, p. 2297] A tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ is said to be in the TT-format if its elements are given by (3.6).*

The Tensor-Train Decomposition is illustrated symbolically in Figure 3.3. This graphical representation means the following. There are two types of nodes: rectanglular nodes, which contain the indices i_k of the original tensor and at least one auxiliary index, and circular nodes which contain only one auxiliary index. Two rectangular nodes are connected if and only if they have a common auxiliary index α_k. To evaluate the entry of a tensor, we have to multiply the elements of the tensors corresponding to the rectanglular nodes and then perform the summation over all auxiliary indices. The picture in Figure 3.3 looks like a train with carriages and this is why the decomposition is called Tensor-Train Decomposition.

Now it is worth introducing one of the main theorems for the Tensor-Train Decomposition, which also gives a constructive way to compute it.

Theorem 3.2.1. *[20, theorem 2.1] If for each unfolding matrix A_k of form (2.2) of a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$*

$$\textit{rank}(A_k) = R_k, \tag{3.7}$$

then there exists a decomposition (3.6) with TT-ranks not higher than r_k.

Proof. Consider the first unfolding of $\mathcal{A}$ and its dyadic decomposition $A_1 = UV^T$, which can be written also in the index form

$$A_1(i_1, i_2 i_3 \cdots i_N) = \sum_{\alpha_1=1}^{R_1} U(i_1, \alpha_1) V(\alpha_1, i_2 \cdots i_N),$$

and put $U(i_1, \alpha_1) = G_1(i_1, \alpha_1)$. The matrix V can be expressed as

$$V = A_1^T U (U^T U)^{-1} = A_1^T W,$$

or in the index form

$$V(\alpha_1, i_2 \cdots i_N) = \sum_{i_1=1}^{n_1} \mathcal{A}(i_1, \cdots, i_N) W(i_1, \alpha_1).$$

It is possible to notice that V can be treated as an $(N-1)$-dimensional tensor

$$\mathcal{V}(\alpha_1 i_2, i_3, \cdots, i_N) = V(\alpha_1, i_2 \cdots i_N)$$

so that V can be identified with the unfolding V_1 of $\mathcal{V}$. Now, considering the unfoldings V_k, it can be shown that $\text{rank}(V_k) \leq R_k$. Indeed, since (3.7) holds,

$$A_k(i_1 \cdots i_k, i_{k+1} \cdots i_N) = \sum_{\beta=1}^{R_k} F(i_1 \cdots i_k, \beta) G(\beta, i_{k+1} \cdots i_N).$$

Using this last expression we obtain

$$\begin{aligned} V_k(\alpha_1 i_1 \cdots i_{k+1}, i_{k+2} \cdots i_N) &= \sum_{i_1=1}^{I_1} A_k(i_1 \cdots i_k, i_{k+1} \cdots i_N) W(i_1, \alpha_1) \\ &= \sum_{i_1=1}^{I_1} \sum_{\beta=1}^{R_k} F(i_1 \cdots i_k, \beta) G(\beta, i_{k+1} \cdots i_N) W(i_1, \alpha_1) \qquad (3.8) \\ &= \sum_{\beta=1}^{R_k} H(\alpha_1 i_2 \cdots i_k, \beta) G(\beta, i_{k+1} \cdots i_N), \end{aligned}$$

Algorithm 2 TT-SVD [23, p.31-32].

Require: Tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$

Compute the size of the first unfolding of $\mathcal{A}$, A_1:

$$N_l = I_1, \qquad N_r = \prod_{k=2}^{N} I_k;$$

create a copy of the original tensor $\mathcal{M} = \mathcal{A}$;

unfold $\mathcal{M}$ into the computed dimensions: $M = reshape(\mathcal{M}, [N_l, N_r])$;

compute the SVD of M, $M = U\Sigma V^T$;

set the first core tensor $G_1 := U$;

recompute $M = \Sigma V^T = U^T M$ and let $[\sim, R_1] = size(U)$;

for $k = 2, \cdots, N-1$ **do**

calculate the dimensions

$$N_l = I_k, \qquad N_r = \frac{N_r}{I_k};$$

unfold M into the computed dimensions: $M = reshape(M, [R_{k-1} * Nl, Nr])$;

compute the SVD of M, $M = U\Sigma V^T$;

recompute $M = U^T M$ and let $[\sim, R_k] = size(U)$;

Set the k-th core tensor $G_k = reshape(U, [R_{k-1}, I_k, R_k])$.

end for

$G_d := M$

where

$$H(\alpha_1 i_2 \cdots i_k, \beta) = \sum_{i_1=1}^{I_1} F(i_1 \cdots i_k, \beta) W(i_1, \alpha_1).$$

From (3.8) we have that $\text{rank}(V_k) \leq R_k, \forall\, k = 1, \cdots, N$. Now if we consider the unfolding V_1, we have

$$V_1(\alpha_1 i_2, i_3 \cdots i_N) = \sum_{\alpha_2=1}^{R_2} G_2(\alpha_1 i_2, \alpha_2) V'(\alpha_2, i_3 \cdots i_N).$$

If we iterate this process, we can find the other tensors G_k, for $k = 3, \cdots, N$. $\square$

The proof of Theorem 3.2.1 enables us to write Algorithm 2. First of all, the first unfolding of $\mathcal{A}$, A_1 should be computed. According to (2.2),

$$A_1(i_1, i_2 \cdots i_N) = \mathcal{A}(i_1, \cdots, i_N),$$

so

$$A_1 = reshape(\mathcal{A}, [I_1, \prod_{k=2}^{N} I_k]).$$

Then, in order to compute the dyadic decomposition of A_1 the SVD of A_1 ($A_1 = U\Sigma V^T$) can be considered. Now, according to the proof of Theorem 3.2.1, we put $G_1 := U$. To compute the second term of the Tensor-Train Decomposition G_2, we have to consider the dyadic decomposition of ΣV^T, i.e. the matrix V_1 of the proof. Thus, in Algorithm 2 the SVD of ΣV^T is computed. We iterate this process $n-2$ times. Then we put $G_d := \Sigma V$.

Now we illustrate the Tensor-Train Decomposition of a third-order tensor for the data in the Example 3.2.1.

Example 3.2.1. *Consider the tensor $\mathcal{A}$ given in Example 3.1.1. Using the Algorithm 2 we can find*

$$G_1 = \begin{pmatrix} -0.4183 & -0.7246 & 0.4743 & -0.2739 \\ -0.4705 & -0.2804 & -0.4283 & 0.7187 \\ -0.5227 & 0.1637 & -0.5664 & -0.6158 \\ -0.5749 & 0.6079 & 0.5204 & 0.1710 \end{pmatrix},$$

$$G_2 = \left(\begin{array}{cc|cc|cc} -0.6885 & -0.7252 & 0.1562 & 0.1481 & -0.4823 & 0.4558 \\ 0.0022 & -0.0021 & 0.7082 & 0.6724 & -0.2516 & 0.4774 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & -0.2339 & 0.2209 \\ 0.0000 & 0.0000 & -0.0000 & 0.0000 & -0.2599 & 0.3119 \end{array}\right),$$

$$G_3 = \begin{pmatrix} 13.4119 & 4.9112 & -0.0000 \\ 41.4930 & 1.5269 & 0.0000 \\ 69.5741 & -1.8574 & 0.0000 \end{pmatrix},$$

where G_2 is displayed in an unfolded manner.
It is important to notice that matrix $G_2(:,:,3)$ and the last two columns of G_1 are not uniquely determined since the corresponding singular values are equal to 0.

3.2.2 The truncated TT-SVD

As for the decompositions described earlier, also the TT-SVD can be truncated. To do this, Algorithm 2 can be modified in the following way. Instead of the exact SVD, the best rank R_k approximation via SVD is computed. Then the introduced error can be estimated.

Theorem 3.2.2. *[20, theorem 2.2] Suppose that the unfoldings A_k of the tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ can be approximated by matrices of low rank $\hat{A}_k$*

$$A_k = \hat{A}_k + E_k, \qquad rank(\hat{A}_k) = R_k, \qquad \|E_k\| = \epsilon_k, \qquad k = 1, \cdots, N-1. \tag{3.9}$$

Then TT-SVD computes a tensor $\mathcal{B}$ in the TT-format with TT-ranks R_k and

$$\|\mathcal{A} - \mathcal{B}\| \leq \sqrt{\sum_{k=1}^{N-1} \epsilon_k^2}.$$

Proof. The proof is by induction on the order N of the tensor.
If $N = 2$, $\mathcal{A}$ and $\mathcal{B}$ are matrices and $\mathcal{B}$ is the truncated SVD of $\mathcal{A}$. Thus, for the property of the SVD (see Theorem 1.3.2) we have

$$\|\mathcal{A} - \mathcal{B}\| = \min_{\operatorname{rank}(\mathcal{C})=k} \|\mathcal{A} - \mathcal{C}\| \leq \|A_1 - \hat{A}_1\| = \epsilon.$$

Now consider $N > 2$. The unfolding of $\mathcal{A}$, A_1 can be decomposed, using the truncated SVD, as

$$A_1 = U_1 \Sigma V_1^T + E_1 = U_1 B_1 + E_1, \qquad \|E_1\| = \epsilon_1.$$

According to Algorithm 2, we set $U_1 = G_1$. Thus, we have found the first term of the TT-Decomposition of $\mathcal{A}$. Since $U_1^T E_1 = 0$, in order to find the second term of the TT-Decomposition, we should consider the truncated SVD of B_1, that is

$$B_1 = U_2 B_2 + E_2 \qquad \|E_2\| = \epsilon_2 \qquad G_2 = reshape(U_2, [R_{k-1}, I_k, R_k]).$$

By iterating this process we can find $G_3, \cdots, G_N$. Let

$$\mathcal{B}(i_1, \cdots, i_N) = G_1(i_1) \underbrace{G_2(i_2) \cdots G_N(i_N)}_{=\hat{\mathcal{B}}} = G_1(i_1) \hat{\mathcal{B}}(i_2, \cdots, i_N),$$

Then it follows that

$$\begin{aligned} \|\mathcal{A} - \mathcal{B}\|^2 &= \|A_1 - G_1 \hat{B}_1\|^2 = \|A_1 - U_1 \hat{B}_1\|^2 \\ &= \|A_1 - U_1(\hat{B}_1 + B_1 - B_1)\|^2 \\ &= \|A_1 - U_1 B_1\|^2 + \|U_1(B_1 - \hat{B}_1)\|^2. \end{aligned}$$

and since U_1 has orthonormal columns,

$$\|\mathcal{A} - \mathcal{B}\|^2 \leq \epsilon_1^2 + \|B_1 - \hat{B}_1\|^2.$$

It can be shown that the distance of the k-th unfolding ($k = 2, \cdots, N-1$) of the $(N-1)$-dimensional tensor $\hat{\mathcal{B}}$ to the R_k-th rank matrix cannot be larger than ϵ_k. Proceeding by induction we have

$$\|B_1 - \hat{B}_1\|^2 \leq \sum_{k=2}^{N-1} \epsilon_k^2.$$

This completes the proof. □

Corollary 3.2.1. *Consider a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and its TT approximation $\mathcal{B}$, computed via the TT-SVD algorithm by keeping the first m singular values of the unfolding matrices A_k. Then*

$$\|\mathcal{A} - \mathcal{B}\| \leq \sqrt{\sum_{k=1}^{N-1} \sum_{s=m+1}^{R_k} (\sigma_s^k)^2},$$

where σ_s^k, $k = 1, \cdots, R_k$ are the singular values of A_k.

Proof. Since $\mathcal{B}$ is computed via the TT-SVD algorithm, (3.9) holds and, from Theorem 1.3.2,

$$\epsilon_k = \sqrt{\sum_{s=m+1}^{R_k} (\sigma_s^k)^2}.$$

$\square$

Corollary 3.2.2. *[20, corollary 2.4] Given a tensor $\mathcal{A}$ and rank bounds R_k, the best approximation to $\mathcal{A}$ in the Frobenius norm with TT-ranks bounded by R_k always exists (denote it by $\mathcal{A}^{best}$), and the TT-approximation computed by the TT-SVD algorithm is quasi optimal, that is*

$$\|\mathcal{A} - \mathcal{B}\| \leq \sqrt{(d-1)}\|\mathcal{A} - \mathcal{A}^{(best)}\|.$$

Proof. Consider $\epsilon = \inf_{\mathcal{C}} \|\mathcal{A} - \mathcal{C}\|$, where the infimum is taken over all tensor trains with TT-ranks bounded by R_k. By the definition of infimum, it is possible to find a sequence of tensor trains $\mathcal{B}^{(s)}$ such that

$$\lim_{s \to +\infty} \|\mathcal{A} - \mathcal{B}^{(s)}\| = \epsilon.$$

Since the sequence $\mathcal{B}^{(s)}$ is bounded, there exists a subsequence $\mathcal{B}^{(s_t)}$ that converges elementwise to $\mathcal{B}^{(\min)}$ and unfolding matrices $B_k^{s_t}$ also converge to $B_k^{(\min)}$, for all $1 \leq k \leq N-1$. Since the set of matrices of rank not higher than R_k is closed,

$$\operatorname{rank}(B_k^{(\min)}) \leq R_k,$$

and $\|\mathcal{A} - \mathcal{B}^{(\min)}\| = \epsilon$, $\mathcal{B}^{(\min)}$ is the minimizer. If we notice that $\epsilon_k \leq \epsilon$,because each unfolding can be approximated with at least accuracy ϵ, then we have

$$\begin{aligned}\|\mathcal{A} - \mathcal{B}\| &\leq \sqrt{\sum_{k=1}^{N-1} \epsilon_k^2} \leq \sqrt{\sum_{k=1}^{N-1} \epsilon^2} = \sqrt{(d-1)}\epsilon = \sqrt{(d-1)}\|\mathcal{A} - \mathcal{B}^{(\min)}\| = \\ &= \sqrt{(d-1)}\|\mathcal{A} - \mathcal{A}^{(best)}\|.\end{aligned}$$

$\square$

Example 3.2.2. *Let $\mathcal{A}$ be the tensor given in Example 3.1.2. If we consider its Tensor-Train Decomposition computed via the TT-SVD algorithm by keeping the first 2 singular values of the unfolding matrices A_k, $k = 1, 2$, we have the following TT-cores:*

$$G_1 = \begin{pmatrix} -0.1121 & 0.7739 \\ -0.5771 & -0.5613 \\ -0.8090 & 0.2932 \end{pmatrix},$$

$$G_2 = \left(\begin{array}{ccc|ccc} -0.5602 & 0.0400 & -0.7492 & -0.2943 & -0.6039 & 0.2185 \\ 0.1946 & 0.2608 & -0.1318 & 0.5504 & -0.1037 & 0.4329 \end{array}\right),$$

$$G_3 = \begin{pmatrix} 4.3031 & 8.2509 & -0.0674 \\ -0.0473 & 0.0627 & 4.6585 \end{pmatrix}.$$

If we denote by $\hat{\mathcal{A}}$ the tensor whose TT-decomposition is given by

$$\hat{\mathcal{A}}(i_1, i_2, i_3) = \sum_{\alpha_1, \alpha_2} G_1(i_1, \alpha_1) G_2(\alpha_1, i_2, \alpha_2) G_3(\alpha_2, i_3),$$

Then the error is given by $\|\mathcal{A} - \hat{\mathcal{A}}\|^2 = 0.3073$ and it is bounded by $(\sigma_3^1)^2 + (\sigma_3^2)^2 = 1.0632$.

3.2.3 Properties of the TT-format

Many linear algebra operations with TT-tensors yield results also in the TT-format but with increased ranks. To reduce ranks while mantaining accuracy one can use the truncated version of the TT-SVD algorithm. It is important to notice that, if the tensor is already in the TT-format, the complexity of the algorithm is significantly reduced.
Consider a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ in the TT-format with suboptimal ranks R_k, $k = 1, \cdots, N$

$$\mathcal{A}(i_1, \cdots, i_N) = G_1(i_1) G_2(i_2) \cdots G_N(i_N).$$

We want to estimate the true values of ranks $R'_k \leq R_k$, while mantaining the prescribed accuracy ϵ. Consider the unfolding A_1 of $\mathcal{A}$. Using the TT-decomposition of $\mathcal{A}$, A_1 can be written as

$$A_1 = UV^T, \tag{3.10}$$

where $U(i_1, \alpha_1) = G_1(i_1, \alpha_1)$ and $V(i_2 i_3 \cdots i_N, \alpha_1) = G_2(\alpha_1, i_2) G_3(i_3) \cdots G_{N-1}(i_{N-1}) G_N(i_N)$. In order to compute the SVD of A_1, one can compute the QR-decompositions of U and V, that is

$$U = Q_U R_U, \qquad V = Q_V R_V,$$

and consider the $r \times r$ matrix $P = R_U R_V^T$. Then the truncated SVD [1] of P can be computed

$$P = XDY^T.$$

[1] The truncated SVD is computed with the prescribed accuracy ϵ.

Finally,

$$\hat{U} = Q_U X, \qquad \text{and} \qquad \hat{V} = Q_V Y$$

are matrices of dominant singular vectors of the full matrix A_1. Since the matrix U is small one can compute its QR-decomposition directly, while, for matrix V it is better to use the following result.

Lemma 3.2.1. *[20, Lemma 3.1] Assume that tensor $\mathcal{Z}$ is expressed as*

$$\mathcal{Z}(i_2, \cdots, i_N) = Q_2(i_2) \cdots Q_N(i_N),$$

where $Q_k(i_k)$ is an $R_{k-1} \times R_k$ matrix, $k = 2, \cdots, N$ (for fixed i_k the product reduces to a vector of length R_1 which is indexed by α_1), and that the matrices $Q_k(i_k)$ satisfy the following orthogonality conditions

$$\sum_{i_k} Q_k(i_k) Q_k^T(i_k) = Id_{R_{k-1}}. \tag{3.11}$$

Then $\mathcal{Z}$, considered as an $R_1 \times \prod_{k=2}^{d} I_k$ matrix, has orthonormal rows, i.e.

$$ZZ^T = Id_{R_1}.$$

Proof.

$$\begin{aligned}
ZZ^T &= \sum_{i_1, \cdots, i_N} \left(Q_1(i_1) \cdots Q_N(i_N)\right) \left(Q_1(i_1) \cdots Q_N(i_N)\right)^T \\
&= \sum_{i_1, \cdots, i_{N-1}} \left(Q_1(i_1) \cdots Q_{N-1}(i_{N-1})\right) \left(\sum_{i_N} Q_N(i_N) Q_N(i_N)^T \right) \\
&\quad \left(Q_{N-1}^T(i_{N-1}) Q_{N-2}^T(i_{N-2}) \cdots Q_1^T(i_1)\right) \\
&= \sum_{i_1, \cdots, i_{N-1}} \left(Q_1(i_1) Q_2(i_2) \cdots Q_N(i_N)\right) \left(Q_{N-1}^T(i_{N-1}) Q_{N-2}^T(i_{N-2}) \cdots Q_1^T(i_1)\right) \\
&= \sum_{i_1} Q_1(i_1) Q_1^T(i_1) = Id_{R_1}.
\end{aligned}$$

□

Lemma 3.2.1 enables us to compute the QR-Decomposition of V in a structured way. The matrix V has the following expression

$$V(i_2 \cdots i_N) = G_2(i_2) \cdots G_N(i_N).$$

Now, if we consider the QR-Decomposition of $G_N(i_N) = R_N Q_N(i_N)$ (where $Q_N(i_N)$ has orthonormal rows), V can be written as

$$V(i_2 \cdots i_N) = G_2(i_2)G_3(i_3)\cdots \underbrace{G_{N-1}(i_{N-1})R_N}_{=G'_{N-1}} Q_N(i_N).$$

By iterating this process we have

$$V(i_2 \cdots i_N) = G_2(i_2)\cdots G'_k(i_k)Q_{k+1}(i_{k+1})\cdots Q_N(i_N),$$

where matrices $Q_s(i_s)$ satisfy (3.11) for $s = k+1, \cdots, N$. Thus

$$V(i_2 \cdots i_N) = R_2 Q_2(i_2)\cdots Q_N(i_N),$$

where $Z = Q_2(i_2)\cdots Q_N(i_N)$ is a matrix with orthonormal rows (see Lemma 3.2.1).

Chapter 4

Face Recognition

Automatic Face Recognition has become increasingly important in the past few years due to its several applications in daily life such as social media platforms and security services. In this chapter we first describe a Face Recognition algorithm based on the SVD which does not work well with huge databases. To cope with this problem we introduce two algorithms based on the tensor decompositions seen in the previous chapters: HOSVD and Tensor-Train.

4.1 Description of the Databases

Each database contains images of n_p persons in n_e different expressions. In this chapter we refer to different illuminations, view angle, etc. as expressions. Images can be both considered as $n_1 \times n_2$ matrices and, by reshaping the columns, as vectors in $\mathbb{R}^{n_i}$ where $n_i = n_1 n_2$ is the number of pixels. In Table 4.1 pixel sizes of the images contained in the various databases are summarized.

Now we introduce three different databases that have been used to test the algorithms described in the following sections.

The Yale Database ([9]) contains 165 grayscale images in GIF format of 15 persons. Each subject is photographed in 11 different expressions: center-light, glasses, happy, left-light, no glasses, normal, right-light, sad, sleepy, surprised, and wink. In Figure 4.1 all subjects in the expression *surprised* are illustrated.

The Orl Database ([19]) contains 400 grayscale images in PGM format of 40 persons.

	Yale	Orl	Extended Yale	Extended Yale shrunk
Pixel size	320×243	92×112	640×480	20×15
Pixel size (vector format)	77760	10304	307200	300

Table 4.1: Pixels size of the databases used for Face Recognition.

Figure 4.1: Faces of the Yale Database.

Figure 4.2: Subjects of the Orl Database in expression 1.

Figure 4.3: Subject 1 of the Orl Database in 10 different expressions.

Figure 4.4: Subject 17 of the Extended Yale B Database.

Each subject is photographed in 10 different expressions, that are shown in Figure 4.3. In Figure 4.2 all subjects in the *expression 1* are illustrated.

The Extended Yale Database contains 16380 grayscale images in PGM format of 28 persons. Each person is photographed in 9 poses and 65 illumination conditions. In Figure 4.4 some of the expressions of subject 17 are illustrated.

For the following tests a shrunk version of the Extended Yale Database has been used.

Consider now the following classification problem: given an image of an unknown person, represented by a vector in $\mathbb{R}^{n_i}$, determine which of the n_p persons the new image is closest to. For this classification problem several decomposition techniques, such as SVD, HOSVD and Tensor-Train Decomposition, can be used.

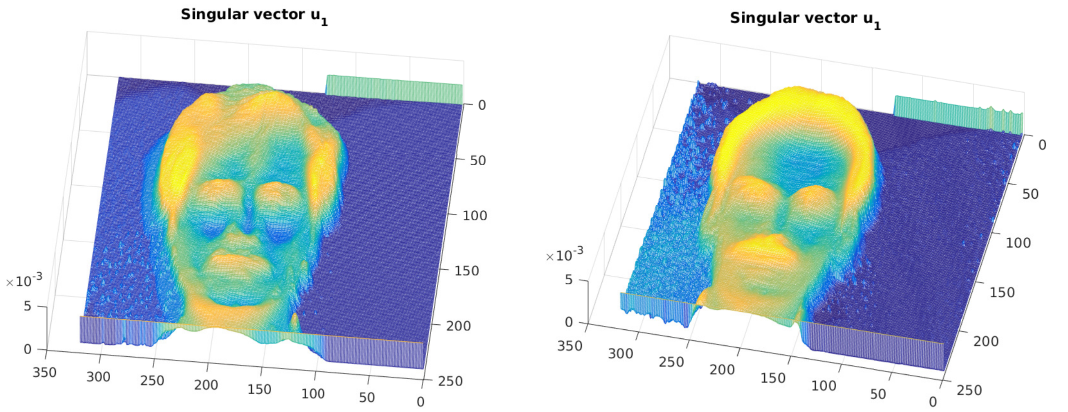
4.2 Face Recognition using SVD

Since images can be considered as vectors in $\mathbb{R}^{n_i}$, a database of images of n_p persons in n_e different expressions can be represented by n_p different matrices $A_p \in \mathbb{R}^{n_i \times n_e}$ $(p = 1, \cdots, n_p)$.

To perform Face Recognition we have to split the database in a *training set* and a *test set*. For example, if we refer to the Yale Database, we can choose the images of 15 persons in the first 9 expressions as the training set and the remaining images (15 persons in the last 2 expressions) as the test set.

We start by describing a simple algorithm which is based on the Singular Value Decomposition. The idea is to "model" the variation of faces of each person in the training set using an orthogonal basis of the subspace of $\mathbb{R}^{n_i}$ spanned by the columns of A_p. This basis can be computed using the SVD, which enables us to write A_p as a sum of rank-one matrices:

$$A_p = \sum_{i=1}^{n_e} \sigma_i^{(p)} u_i^{(p)} \left(v_i^{(p)}\right)^T, \qquad p = 1, \cdots, n_p.$$



(a) *First left singular vector $u_1^{(1)}$ of subject 1.* (b) *First left singular vector $u_1^{(2)}$ of subject 2.*

Figure 4.5: First left singular vectors from the Yale Database.

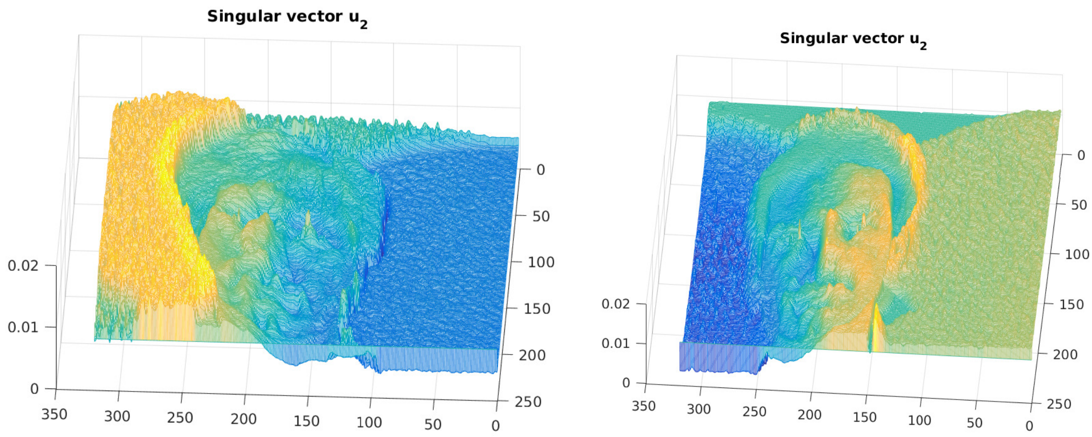
Each column j of A_p can be written as

$$(A_p)_j = \sum_{i=1}^{n_e} \sigma_i^{(p)} u_i^{(p)} v_{ij}^{(p)},$$

where $v_{ij}^{(p)}$ is a scalar and can be thought of as a weight for the expression j of person p. Thus each column in A_p represents an image of the person p in a certain expression and therefore the left singular vectors $u_i^{(p)}$ are an orthogonal basis in the "image space of person p". From Theorems 1.3.1 and 1.3.2, we know that the first left singular vector represents the *dominating* direction of the data. Thus, we expect the first left singular vector to look like person p in a sort of "mean expression" (see Figure 4.5). According to this interpretation of the first left singular vector, the elements of the first right singular vector are almost the same, so different expressions of person p have the same weight.

On the other hand, the subsequent left singular vectors should represent the dominating variations of the training set around the first left singular vector. Thus, the second right singular vector should have the largest entries in correspondence with expressions farthest from the "mean expression". For example, if we refer to the first and the second subjects of the Yale Database, the largest entries are in correspondence with the expressions *leftlight* and *rightlight* (see Figure 4.6).

In the classification of an unknown face (i.e. a face in the test set) we need to compute its *distance* to known faces (i.e. the faces in the training set). To do this, by means of the SVD, we should compute how well an unknown face can be represented in the n_p different bases. This can be done by determining the minimum distance from all "face



(a) *Second left singular vector* $u_2^{(1)}$ *of subject 1.* (b) *Second left singular vector* $u_2^{(2)}$ *of subject 2.*

Figure 4.6: Second left singular vectors from the Yale Database.

subspaces", that is

$$\min_{\alpha_i} \|z - \sum_{i=1}^{n_e} \alpha_i u_i^{(p)}\| \qquad \forall \ p = 1, \cdots, n_p, \tag{4.1}$$

where z is the image of the unknown face. From the previous discussion, we could argue that $U^{(p)}$ contains the principal *latent* expressions of person p.

The minimization problem (4.1) can also be written in the following form

$$\min_{\alpha} \|z - U^{(p)}\alpha\|, \qquad U^{(p)} = \left(u_1^{(p)}, \cdots, u_{n_e}^{(p)}\right).$$

Since the columns of $U^{(p)}$ are orthonormal, the solution of the problem is given by $\alpha = \left(U^{(p)}\right)^T z$, thus

$$\min_{\alpha} \|z - U^{(p)}\alpha\| = \left\|\left(I - U^{(p)}\left(U^{p}\right)^T\right) z\right\|.$$

Now it is possible to give a Face Recognition algorithm based on the SVD (see Algorithm 3). A typical computation of the resulting distances is given in Figure 4.7.

By testing this algorithm on three different datasets, we obtained the results in Table 4.2. For these tests, the expressions used for the test set were chosen randomly, by taking $\frac{s}{100} n_e$ expressions for each person. The remaining images were used as the training set. Looking at Table 4.2, it is possible to notice that the SVD based Face Recognition algorithm works well only when $n_i > n_e$, i.e. with all the choices of s for the Yale and the Orl Database and only with $s = 50$ for the Extended Yale Database. This happens because when $n_e > n_i$ the distance $d(e, p)$ computed in Algorithm 3 is equal to zero. However, in many applications, it happens that $n_i < n_e$ because of the huge database

Algorithm 3 Face Recognition with SVD [11].

Require: $z \in \mathbb{R}^{n_i}$ input image and $U^{(p)}$, $\forall p = 1, \cdots, n_p$.
1: **for** $p = 1, \cdots, n_p$ **do**
2: $\quad d(p) = \| \left(I - U^{(p)} \left(U^p\right)^T \right) z \|$
3: **end for**
4: $[d_{min}, p_{hat}] = \min d$.
5: Classify z as person p_{hat}

s	Yale	Orl	Extended Yale shrunk
50	86.44%	94.84%	99.87%
60	86.19%	96.88%	57.98%
70	87.07%	97.07%	59.03%
80	88.89%	97.80%	59.59%

Table 4.2: Face Recognition performance of Algorithm 3 with three different databases and four different splits (s).

dimension. This suggests that alternative strategies should be considered when whenever the number of expressions n_e is significantly larger than the number of pixels n_i. Tensor methods provide viable strategies.

Face Recognition with truncated SVD

Before moving further, we deal with a variant of Algorithm 3, where the truncated SVD instead of the exact SVD is considered. In particular, we truncated according with a parameter p related to the singular values, whose characteristic pattern is given in Figure 4.8.

We considered the vector w, whose entries are given by

$$w(l) = \frac{\sum_{i=1}^{l} \sigma_i}{\sum_{i=1}^{n_e} \sigma_i}.$$

Then, we considered only the first k singular values, where k is such that

$$w(l) \leq p \quad \forall\, 1 \leq l \leq k \quad \text{and} \quad w(l) > p \quad \forall\, k > l.$$

As we can see from Table 4.3, the recognition performance is quite similar to that in Table 4.2. As a matter of fact, according to Theorem 1.3.2, the error due to the truncation is low if the decay of the singular values is fast enough. Thus, it is worth considering a truncated version of the algorithm, since it gives good recognition performance with higher efficiency in terms of speed and memory requirements. Hence, for $p = 0.9$ and

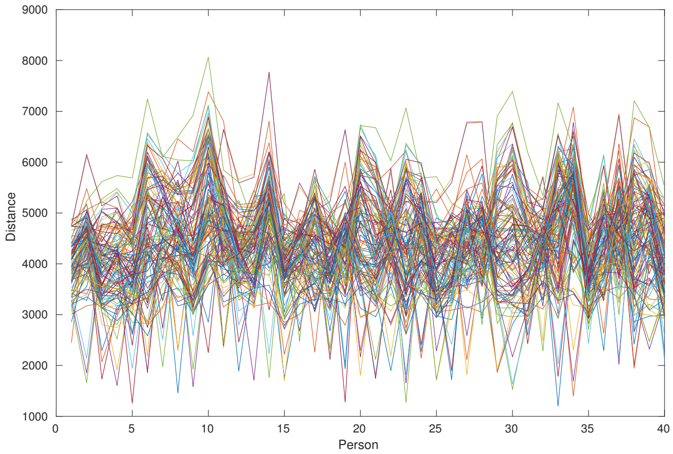
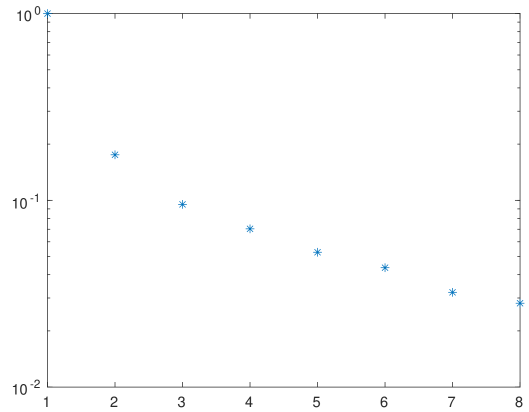


Figure 4.7: Example of distances from the Orl Database.

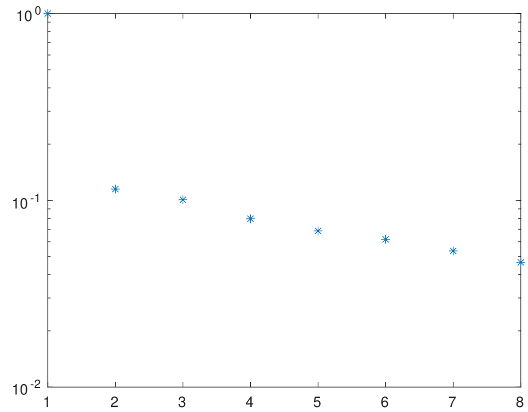
s	p	Yale	Orl	Extended Yale shrunk
80	0.70	58.31%	79.40%	95.51%
	0.75	76.71%	93.70%	95.67%
	0.80	79.29%	96.50%	95.85%
	0.85	81.60%	97.95%	96.52%
	0.90	88.36%	98.15%	96.79%
	0.95	88.71%	98.00%	96.95%
70	0.70	39.20%	59.60%	95.51%
	0.75	68.00%	90.00%	95.67%
	0.80	80.07%	94.77%	95.85%
	0.85	86.27%	96.73%	96.52%
	0.90	87.93%	97.23%	96.79%
	0.95	87.33%	97.40%	96.95%

Table 4.3: Face Recognition performance, using the truncated SVD with different values of parameter p, tested on three different databases and using two different splits: $s = 70$ and $s = 80$.

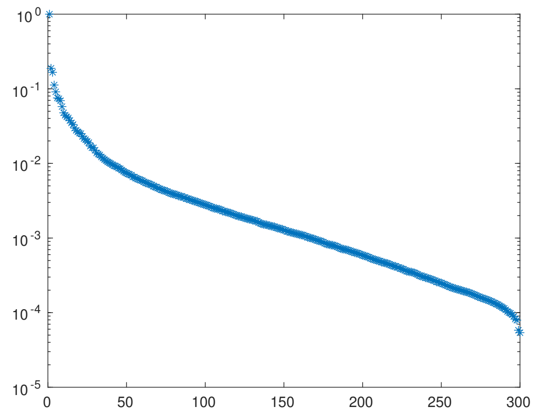
$s = 0.8$ in the Extended Yale Database we are considering only the 22% of singular values.



(a) *Singular values of A_1 from Yale.*



(b) *Singular values of A_1 from Orl.*



(c) *Singular values A_1 from Extended Yale.*

Figure 4.8: Singular values of A_1 from three different databases.

4.3 Face Recognition using Tensor Decompositions

In the previous sections a database of n_p persons photographed in n_e different expressions was represented by p matrices $A_p \in \mathbb{R}^{n_i \times n_e}$, where n_i is the number of pixels of each image. Using multilinear algebra and the algebra of higher-order tensors, the same database can be represented as a tensor $\mathcal{A} \in \mathbb{R}^{n_i \times n_e \times n_p}$ (see Figure 4.9).

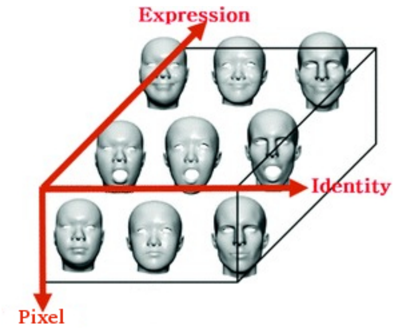


Figure 4.9: Tensor Representation of a database.

4.3.1 Face Recognition using HOSVD

Consider $\mathcal{A} \in \mathbb{R}^{n_i \times n_e \times n_p}$. It is possible to decompose $\mathcal{A}$ using the HOSVD as in (3.1)

$$\mathcal{A} = \mathcal{S} \times_i F \times_e G \times_p H, \tag{4.2}$$

where $\times_i$, $\times_e$, $\times_p$ are the 1-mode, 2-mode, 3-mode multiplication, respectively [1]. To give an interpretation of the HOSVD of $\mathcal{A}$, it is better to write the decomposition in the following form

$$\mathcal{A} = \mathcal{D} \times_e G \times_p H,$$

where $\mathcal{D} = \mathcal{S} \times_i F$. By fixing a particular expression e_0 and a particular person p_0 we obtain the vector

$$\mathcal{A}(:, e_0, p_0) = \mathcal{D} \times_e g_{e_0} \times_p h_{p_0}$$

where $g_{e_0} = G(e_0, :)$ and $h_{p_0} = H(p_0, :)$. In other words, a particular expression e_0 is characterized by the vector g_{e_0} and a particular person p_0 is characterized by the vector h_{p_0}, via the bilinear form

$$\mathcal{D} \times_e g \times_p h.$$

[1] $\times_i$ is the *image-mode* multiplication, $\times_e$ is the *expression-mode* multiplication, $\times_p$ is the *person-mode* multiplication.

The Face Recognition algorithm

To perform Face Recognition, (4.2) can be written in the following form

$$\mathcal{A} = \mathcal{C} \times_p H, \tag{4.3}$$

where

$$\mathcal{C} := \mathcal{S} \times_i F \times_e G. \tag{4.4}$$

If we fix the expression e and we identify the tensors $\mathcal{A}(:, e, :)$ and $\mathcal{C}(:, e, :)$ with matrices A_e and C_e, (4.3) becomes

$$A_e = C_e H^T \qquad\qquad e = 1, 2, \ldots, n_e. \tag{4.5}$$

Hence, column p of A_e can be written as

$$a_p^{(e)} = C_e h_p^T \tag{4.6}$$

Equations (4.5) and (4.6) can be interpreted as follows. Each column of A_e contains the image of the person p in the expression e. This image, using (4.6), can be written as a matrix-vector product, where the columns of the matrix C_e are basis vectors for expression e, and row p of H (h_p) holds the image coordinates of person p in this basis. Notice that the same h_p holds the coordinates of all the images of person p in the different expression bases.

As in the SVD based Face Recognition algorithm, the columns of F are an orthogonal basis in the "image space", since F comes from the SVD of the first unfolding of $\mathcal{A}$, $A_{(1)}$ ($A_{(1)} = FS_1V_1$). Thus, we expect the 1-mode singular vector to look like a "mean person" in a sort of "mean expression" (see Figure 4.10). On the other hand, the subsequent 1-mode singular vector should represent the most significant variations around the first 1-mode singular vector (see Figure 4.11).

Now let $z \in \mathbb{R}^{n_i}$ be the image of an unknown person in an unknown expression that we want to classify. The coordinates of z in the expression basis can be found by solving a least squares problem

$$\min_{\alpha_e} \|C_e \alpha_e - z\|_2, \qquad e = 1, \cdots, n_e. \tag{4.7}$$

Obviously if z is an image of a person p in expression e, then the coordinates in the expression basis are equal to h_p.

A preliminary version of the classification algorithm is given in Algorithm 4. Since the solution of n_e least squares problems is required, the amount of work in Algorithm 4 is high. To cope with this problem, we consider another version of the algorithm, which is based on the following trick. Consider $F \in \mathbb{R}^{n_i \times n_e n_p}$ and assume that $n_i \gg n_e n_p$. For the analysis only, we can enlarge F, so that it becomes square and orthogonal.

$$\hat{F} = (F \;\; F^{\perp}) \tag{4.8}$$

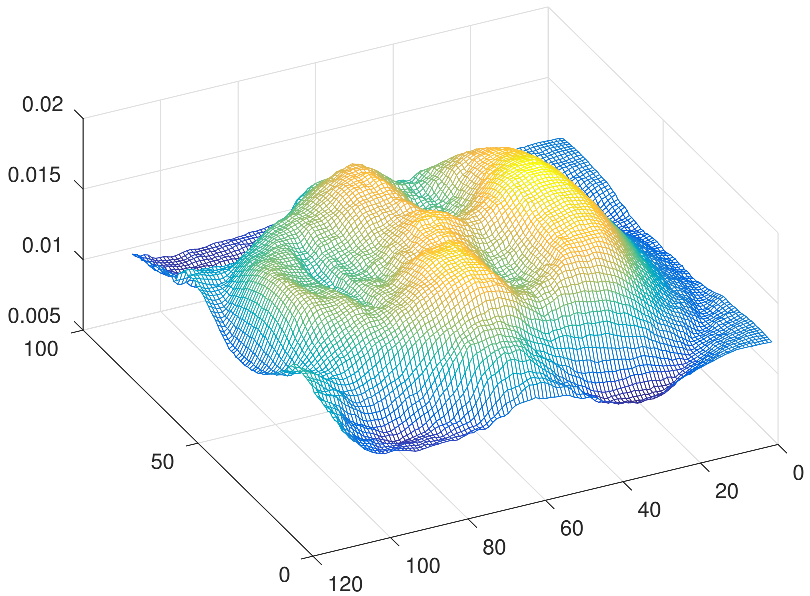


Figure 4.10: First 1-mode singular vector from the Orl Database.

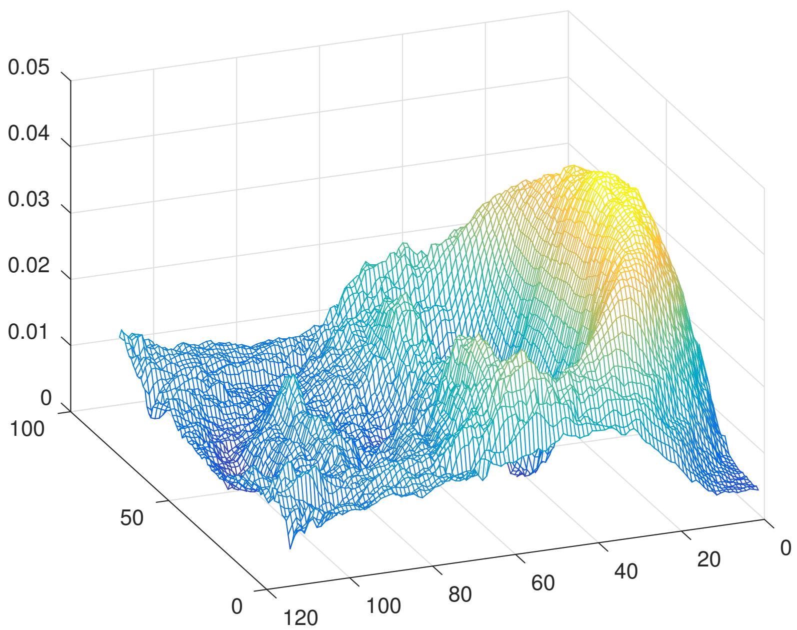


Figure 4.11: Second 1-mode singular vector from the Orl Database.

Algorithm 4 Face Recognition (preliminary version)[11, p.173].

Require: z test image.

for $e = 1, 2, \ldots, n_e$ **do**
  solve $\min_{\alpha_e} \|C_e\alpha_e - z\|_2$.
  for $p = 1, 2, \ldots, n_p$ **do**
    $d(e,p) = \|\alpha_e - h_p^T\|_2$.
  end for
end for
$d1 = \min(d)$;
$[d2, phat] = \min(d1)$.

Algorithm 5 Face Recognition [11, p.174].

Require: z test image.

Compute $\hat{z} = F^T z$:
for $e = 1, \cdots, n_e$ **do**
  Compute the thin QR-Decomposition of B_e
  Solve $R_e\alpha_e = Q_e^T\hat{z}$ for α_e
  for $p = 1, \cdots, n_p$ **do**
    d(e,p)=$\|\alpha_e - h_p\|_2$.
  end for
end for
$d1 = \min(d)$;
$[d2, phat] = min(d1)$.

If we recall equation (4.4) and put $B_e = \mathcal{S} \times_i F \times_e G$, we have

$$\begin{aligned}
\|C_e\alpha_e - z\|_2^2 &= \|\hat{F}(FB_e\alpha_e - z)\|_2^2 \\
&= \left\| \begin{pmatrix} B_e\alpha_e - F^T z \\ -(F^\perp)^T z \end{pmatrix} \right\|_2^2 \\
&= \|B_e\alpha_e - F^T z\|_2^2 + \|(F^\perp)^T z\|_2^2.
\end{aligned}$$

It follows that we can solve n_e least squares problems, by first computing $\hat{z} = F^T z$

$$\min_{\alpha_e} \|B_e\alpha_e - \hat{z}\|_2 \qquad e = 1, \cdots, n_e. \tag{4.9}$$

The matrix B_e is smaller than C_e so it is cheaper to solve (4.9) instead of (4.7). To further reduce work, the QR-Decomposition of B_e can be computed.

Thus we have Algorithm 5. If $n_i < n_e n_p$, like in the Extended Yale Database, $F \in \mathbb{R}^{n_i \times n_i}$ is a square orthogonal matrix, so in equation (4.8) we consider $\hat{F} = F$ and

s	Yale	Orl	Extended Yale shrunk
50	87.08%	91.00%	98.60%
60	87.84%	93.07%	98.94%
70	84.87%	93.86%	98.97%
80	89.65%	95.30%	99.13%

Table 4.4: Face Recognition performance of Algorithm 5 with three different databases and four different splits (s).

both Algorithm 4 and Algorithm 5 can be used. By testing Algorithm 5 on three different datasets, we obtain the results in Table 4.4. The test set was generated randomly, by taking $\frac{s}{100}n_e$ expressions for each person. As in section 4.2, we used a shrunk version of the Extended Yale Database.

Looking at Table 4.4, it is possible to notice that the HOSVD based algorithm works well also when $n_i < n_e$, in the case of the Extended Yale Database for $s = 60, 70, 80$. Thus, in realistic applications, this algorithm performs better than the one based on SVD; compare with Table 4.2.

Face Recognition with HOSVD Compression

Theorem 3.1.2 enables us to deal with a variation of Algorithm 5, where the truncated HOSVD instead of the exact HOSVD is considered. In particular, we truncated according with a parameter p related to the singular values, whose characteristic pattern is given in Figure 4.12. In the following discussion only the truncation along the first mode is considered. Define $F_k = F(:, 1:k)$ for some k smaller than n_i. Then, for the analysis only, we enlarge the matrix so that it becomes square and orthogonal, that is we define

$$\hat{F} = \left(F_k \, \tilde{F}_\perp\right) \in \mathbb{R}^{n_i \times n_i}, \qquad \hat{F}^T \hat{F} = I.$$

Then we truncate the core tensor in the following way

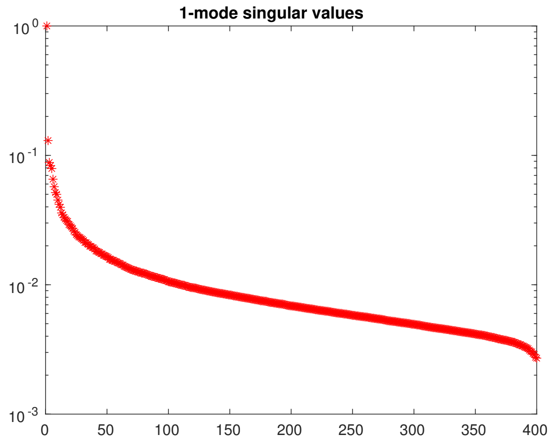
$$\hat{\mathcal{C}} = (\mathcal{S} \times_e G)\,(1:k, :, :) \times_i F_k. \tag{4.10}$$

From Theorem 3.1.2, it follows that

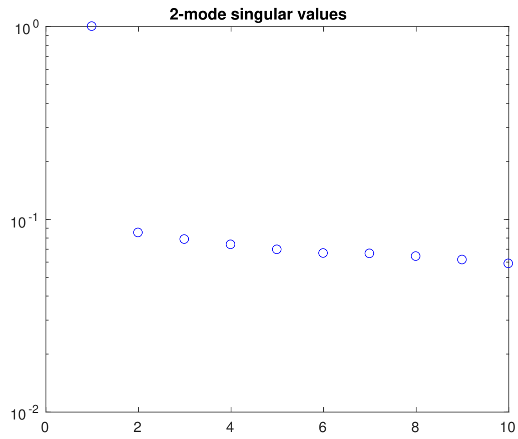
$$\|\hat{\mathcal{C}} - \mathcal{C}\|^2 \le \sum_{j=k+1}^{n_i} \left(\sigma_j^{(i)}\right)^2.$$

Thus, if the decay of the singular values is fast enough, a good recognition accuracy can be obtained, despite the compression. From (4.10), $\hat{\mathcal{C}}_e = F_k \hat{B}_e$, where $\hat{B}_e \in \mathbb{R}^{k \times n_p}$. Multiplying $\left(\hat{\mathcal{C}}_e \alpha_e - z\right)$ by $\hat{F}$ we obtain

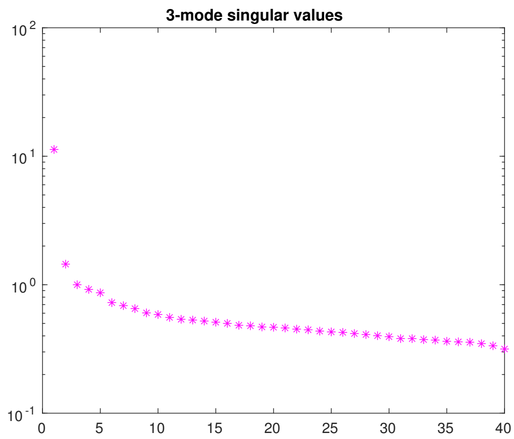
$$\|\hat{\mathcal{C}}_e - z\|_2^2 = \|\hat{B}_e \alpha_e - F_k^T z\|_2^2 + \|\tilde{F}_\perp^T z\|_2^2.$$



(a) *1-mode singular values.*



(b) *2-mode singular values.*



(c) *3-mode singular values.*

Figure 4.12: Singular values from the Orl Database.

s	p	Yale	Orl	Extended Yale shrunk
80	0.30	6.67%	25.10%	14.32%
	0.40	7.38%	57.55%	48.99%
	0.50	37.69%	67.10%	79.15%
	0.60	72.18%	93.40%	83.91%
	0.70	84.80%	95.40%	89.74%
	0.80	92.53%	95.80%	90.42%
	0.90	91.11%	95.45%	91.76%
70	0.30	6.67%	20.60%	15.44%
	0.40	7.27%	53.50%	52.34%
	0.50	37.47%	68.57%	86.66%
	0.60	70.00%	91.20%	93.54%
	0.70	83.40%	93.77%	97.92%
	0.80	87.27%	93.77%	98.60%
	0.90	90.27%	93.67%	98.95%

Table 4.5: Face Recognition performance, using the truncated HOSVD with different values of parameter p, tested on three different databases and using two different splits: $s = 70$ and $s = 80$.

By testing the truncated version of Algorithm 5, we obtained the results in Table 4.5.

As we can see from Table 4.5, the recognition performance is quite similar to that in Table 4.4. Thus, a truncated version of the algorithm can be considered. This enables us to have higher efficiency in terms of speed and memory requirements, while mantaining good recognition performance. Hence, for $p = 0.9$ and $s = 0.8$ in the Extended Yale Database we are considering only the 45% of singular values.

4.3.2 Face Recognition using Tensor Train Decomposition

In this section we explore the use of the Tensor Train decomposition for our problem. We propose a new algorithm for performing a recognition procedure. Given the new image z we thus determine the closest person within the testset, by using a representation of the unknown person in the tensor train basis.

Consider $\mathcal{A} \in \mathbb{R}^{n_i \times n_e \times n_p}$. It is possible to decompose $\mathcal{A}$ using the TT-Decomposition as in (3.6)

$$\mathcal{A}(i_1, i_2, i_3) = G_1(i_1)G_2(i_2)G_3(i_3) \tag{4.11}$$

In order to give a Face Recognition algorithm, (4.11) can be written, using the n-mode product, in the following way

$$\mathcal{A} = G_2 \times_i G_1 \times_p G_3^T, \tag{4.12}$$

Algorithm 6 Face Recognition (preliminary version).

Require: z test image.
for $e = 1, 2, \ldots, n_e$ **do**
  solve $\min_{\alpha_e} \|C_e \alpha_e - z\|_2$.
  for $p = 1, 2, \ldots, n_p$ **do**
    d(e,p)=$\|\alpha_e - g_p\|_2$.
  end for
end for
$d1 = \min(d)$;
$[d2, phat] = min(d1)$.

since

$$\begin{aligned}\mathcal{A} &= \sum_{\alpha_1,\alpha_2} G_1(i_1,\alpha_1)G_2(\alpha_1,i_2,\alpha_2)G_3(\alpha_2,i_3)\\ &= \sum_{\alpha_2}(G_2 \times_i G_1)(i_1,i_2,\alpha_2)G_3(\alpha_2,i_3)\\ &= (G_1 \times_i G_2 \times_p G_3^T)(i_1,i_2,i_3).\end{aligned}$$

If we call $\mathcal{C} = G_2 \times_i G_1$, (4.12) can be written as

$$\mathcal{A} = \mathcal{C} \times_p G_3^T.$$

So, if we identify $\mathcal{A}(:,e,:)$ and $\mathcal{C}(:,e,:)$ with matrices A_e and C_e, we obtain that

$$A_e = C_e G_3$$

and, by fixing the column p of A_e,

$$a_e^{(p)} = C_e g_p,$$

where $g_p = G_3(:,p)$.

Now assume that $z \in \mathbb{R}^{n_i}$ is the image of an unknown person in an unknown expresison that we want to classify. As in subsection 4.3.1, the coordinates of z in the expression basis can be found by solving n_e least squares problems

$$\min_{\alpha_e} \|C_e \alpha_e - z\|_2. \tag{4.13}$$

A first version of the Face Recognition algorithm can thus be given (see Algorithm 6).

As for the HOSVD based Face Recognition algorithm, the final version of the new algorithm can be determined by replacing (4.13) with

$$\min_{\alpha_e} \|G_2^e \alpha_e - \hat{z}\|_2,$$

Algorithm 7 Face Recognition.

Require: z test image.
Compute $\hat{z} = G_1^T z$:
for $e = 1, \cdots, n_e$ **do**
  Compute the thin QR-Decomposition of G_2^e
  Solve $R_e \alpha_e = Q_e^T \hat{z}$ for α_e
  for $p = 1, \cdots, n_p$ **do**
    d(e,p)=$\|\alpha_e - g_p\|_2$.
  end for
end for
$d1 = \min(d)$;
$[d2, phat] = min(d1)$.

s	Yale	Orl	Extended Yale shrunk
50	79.16%	92.96%	98.74%
60	82.83%	94.27%	98.98%
70	84.40%	99.30%	99.09%
80	84.00%	96.35%	99.25%

Table 4.6: Face Recognition performance of TT-Decomposition based algorithm with three different databases and four different splits (s).

where $G_2^e = G_2(:, e, :)$ and $\hat{z} = G_1^T z$. Furthermore, the QR-Decomposition of G_2^e can be considered. The complete procedure is described in Algorithm 7.

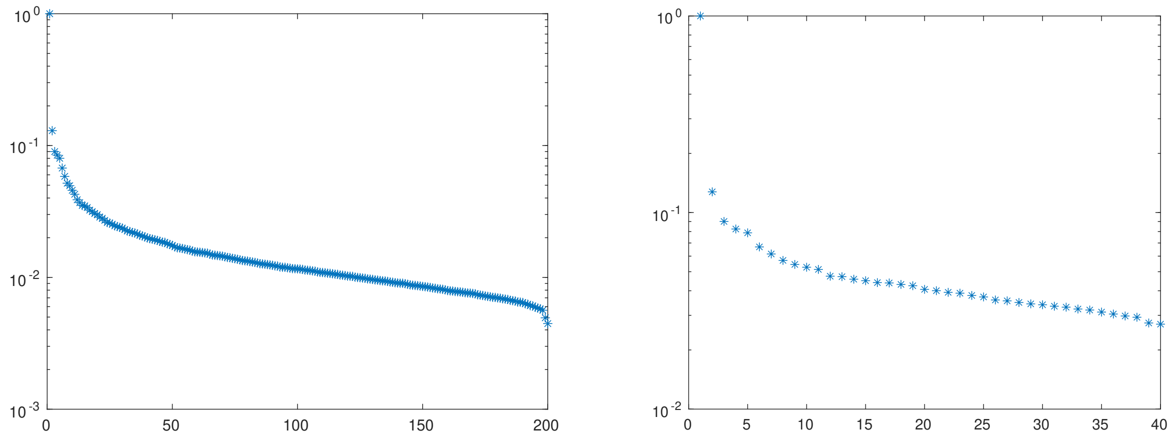
By testing this algorithm on three different datasets, we obtained the results shown in Table 4.6. The test set was generated randomly, by taking $\frac{s}{100} n_e$ expressions for each person. As in section 4.2, we used a shrunk version of the Extended Yale Database.

Looking at Table 4.6, it is possible to notice that the TT-Decomposition based Face Recognition algorithm works well also when $n_i < n_e$, i.e. with the Extended Yale Database choosing $s = 60, 70, 80$. Thus, in realistic applications, this algorithm, together with the HOSVD based algorithm, performs better than the one based on SVD.

Face Recogniton with TT-Decomposition Compression

Corollary 3.2.1 enables us to consider a variation of Algorithm 6, where the truncated TT-SVD, instead of the exact decomposition, is considered.

As in the previous discussion, the truncation is made according with a parameter p related to the singular values, whose characteristic pattern is given in Figure 4.13. Since $\mathcal{C} = G_2 \times_i G_1$, we can consider a truncation along the first mode by truncating $\mathcal{C}$ in the



(a) *Singular values of the first SVD, which gives G_1.* (b) *Singular values of the second SVD, which gives G_2.*

Figure 4.13: Singular values from the Orl Database.

following way

$$\hat{\mathcal{C}} = (G_2 \times_i G_1)(1:k,:,:) = G_2(1:k,:,:) \times_i G_1(1:k,:).$$

From Corollary 3.2.1, it follows that

$$\|\mathcal{C} - \hat{\mathcal{C}}\|^2 \leq \sum_{j=k+1}^{n_i} \left(\sigma_j^{(i)}\right)^2$$

By testing the truncated version of Algorithm 6, we obtained the results shown in Table 4.7. As we can see from Table 4.7, the recognition performance is quite similar to that in Table 4.6. The higher efficiency in terms of speed and memory requirements together with the good recognition performance suggests that the truncated version of the algorithm, instead of the complete one, should be used. The memory requirements for these two versions of Algorithm 6 are significantly different. For $p = 0.9$ and $s = 0.8$ in the Extended Yale Database we are considering only the 45% of singular values.

A comparison between Table 4.5 with 4.7 shows that the Tensor-Train format enables us to achieve a higher recognition performance than with the HOSVD. For example, focusing on the Extended Yale Database and fixing $p = s = 0.8$, we can notice that the TT-SVD algorithm performs significantly better than the HOSVD algorithm.

Furthermore, it is worth observing that the TT-SVD algorithm requires less memory than the one based on HOSVD. This is due to the fact that in the TT-Decomposition of a tensor $\mathcal{A} \in \mathbb{R}^{n_i \times n_e \times n_p}$ we only have three terms (G_1, G_2 and G_3) while in the HOSVD we have four terms ($\mathcal{S}$, F, G, H).

s	p	Yale	Orl	Extended Yale shrunk
80	0.30	6.67%	3.06%	63.84%
	0.40	7.22%	55.12%	78.68%
	0.50	56.00%	82.38%	89.30%
	0.60	76.22%	97.44%	97.15%
	0.70	76.00%	96.63%	98.89%
	0.80	83.78%	96.56%	99.05%
	0.90	86.11%	95.69%	99.23%
70	0.30	6.67%	3.08%	62.23%
	0.40	6.25%	40.83%	80.29%
	0.50	45.08%	75.62%	88.94%
	0.60	70.08%	95.42%	96.83%
	0.70	78.75%	95.63%	98.75%
	0.80	84.92%	96.00%	99.04%
	0.90	86.83%	95.00%	99.15%

Table 4.7: Face Recognition performance, using the truncated TT-SVD with different values of parameter p, tested on three different databases and using two different splits: $s = 70$ and $s = 80$.

Suppose now that we want to consider the truncated version of both algorithms, by taking only the first k singular values in the *pixel-mode*. To do this, with the TT-Decomposition we have to consider $\hat{G}_1 = G_1(:, 1 : k)$, $\hat{G}_2 = G_2(1 : k, :, :)$ and $\hat{G}_3 = G_3$. Thus, the memory requirement m_T for this algorithm is equal to $m_T = n_i k + k n_e n_p + n_p^2$. On the other hand, for the HOSVD we should consider $\hat{\mathcal{S}} = \mathcal{S}(1 : k, :, :)$, $\hat{F} = F(:, 1 : k)$, $\hat{G} = G$ and $\hat{H} = H$. Thus the memory requirement m_H for this algorithm is equal to $m_H = k n_e n_p + n_i k + n_e^2 + n_p^2 > m_T$.

Conclusions and perspectives

In this work we studied tensor calculus by describing some basic tensor concepts, such as the *unfolding* and the *n-mode product*. Then, we anlyzed tensor approximation via two different strategies: Higher-Order Singular Value Decomposition and Tensor-Train Decomposition. Then, we adapted these methodologies to develop several Face Recognition algorithms both in normal and truncated fashion. We tested these algorithms on several databases available online and compared their performance. Our experiments seem to indicate that tensor based methods are able to exploit the better organized information. In particular, the new algorithm we proposed for performing the recognition in the tensor train format showed very good performance: the procedure was able to achieve higher percentages of correct recognitions for comparable memory requirements, with respect to HOSVD.

Further improvements are already foreseen for the near future: firstly, we plan to implement Neural networks using Tensor-Train Decomposition for Face Recognition purposes. Secondly, it is worth considering the Non-negative tensor factorization. This technique enables us to preserve at the tensor level an important property of images, which is the non-negativity of their entries.

Bibliography

[1] Evrim Acar, Daniel M. Dunlavy, and Tamara G. Kolda. "A Scalable Optimization Approach for Fitting Canonical Tensor Decompositions". *Journal of Chemometrics* 25.2 (Feb. 2011), pp. 67–86. DOI: `10.1002/cem.1335`.

[2] Evrim Acar et al. "Scalable Tensor Factorizations for Incomplete Data". *Chemometrics and Intelligent Laboratory Systems* 106.1 (Mar. 2011), pp. 41–56. DOI: `10.1016/j.chemolab.2010.08.004`.

[3] H.C. Andrews and C.L. Patterson. "Singular Value Decompositions and Digital Image Processing". *IEEE Transactions on Acoustics, Speech, and Signal Processing* 24.1 (1976).

[4] Brett W. Bader and Tamara G. Kolda. "Algorithm 862: MATLAB tensor classes for fast algorithm prototyping". *ACM Transactions on Mathematical Software* 32.4 (Dec. 2006), pp. 635–653. DOI: `10.1145/1186785.1186794`.

[5] Brett W. Bader and Tamara G. Kolda. "Efficient MATLAB computations with sparse and factored tensors". *SIAM Journal on Scientific Computing* 30.1 (Dec. 2007), pp. 205–231. DOI: `10.1137/060676489`.

[6] Brett W. Bader, Tamara G. Kolda, et al. *MATLAB Tensor Toolbox Version 3.0-dev.* Available online. Oct. 2017. URL: `https://www.tensortoolbox.org`.

[7] Casey Battaglino, Grey Ballard, and Tamara G. Kolda. *A Practical Randomized CP Tensor Decomposition.* arXiv:1701.06600. Jan. 2017.

[8] Eric C. Chi and Tamara G. Kolda. "On Tensors, Sparsity, and Nonnegative Factorizations". *SIAM Journal on Matrix Analysis and Applications* 33.4 (Dec. 2012), pp. 1272–1299. DOI: `10.1137/110859063`.

[9] Department of Computer Science. *The Yale Face Database.* Sept. 1997. URL: `http://cvc.cs.yale.edu/cvc/projects/yalefaces/yalefaces.html`.

[10] L. De Lathauwer, B. De Moor, and J. Vandewalle. "A Multilinear Singular Value Decomposition". *SIAM J. Matrix Anal. Appl.* 21.4 (2000).

[11] L. Eldén. *Matrix Methods in Data Mining and Pattern Recognition.* Philadelphia: SIAM, 2007.

[12] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Baltimore: The Johns Hopkins University Press, 1996.

[13] Samantha Hansen, Todd Plantenga, and Tamara G. Kolda. "Newton-Based Optimization for Kullback-Leibler Nonnegative Tensor Factorizations". *Optimization Methods and Software* 30.5 (Apr. 2015), pp. 1002–1029. DOI: 10.1080/10556788.2015.1009977.

[14] Tamara G. Kolda. "Numerical Optimization for Symmetric Tensor Decomposition". *Mathematical Programming B* 151.1 (Apr. 2015), pp. 225–248. DOI: 10.1007/s10107-015-0895-0.

[15] Tamara G. Kolda and Jackson R. Mayo. "An Adaptive Shifted Power Method for Computing Generalized Tensor Eigenpairs". *SIAM Journal on Matrix Analysis and Applications* 35.4 (Dec. 2014), pp. 1563–1581. DOI: 10.1137/140951758.

[16] Tamara G. Kolda and Jackson R. Mayo. "Shifted Power Method for Computing Tensor Eigenpairs". *SIAM Journal on Matrix Analysis and Applications* 32.4 (Oct. 2011), pp. 1095–1124. DOI: 10.1137/100801482.

[17] Tamara G. Kolda and Jimeng Sun. "Scalable Tensor Decompositions for Multi-aspect Data Mining". *ICDM 2008: Proceedings of the 8th IEEE International Conference on Data Mining*. Dec. 2008, pp. 363–372. DOI: 10.1109/ICDM.2008.89.

[18] T.G. Kolda and B.W. Bader. "Tensor Decompositions and Applications". *SIAM Review* 51.3 (2009).

[19] Cambridge University Computer Laboratory. *The ORL Database of Faces*. 2002. URL: https://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html.

[20] I.V. Oseledets. "Tensor-Train Decomposition". *SIAM J. Sci. Comput.* 33.5 (2011).

[21] D. Palitta and Simoncini V. *Dispense del corso di Calcolo Numerico*. Dec. 2016.

[22] R.A. Sadek. "SVD Based Image Processing Applications: State of The Art, Contributions and Research Challenges". *International Journal of Advanced Computer Science and Applications* 3.7 (2012).

[23] D. Tock. *Tensor Decomposition and its Applications*. Sept. 2010.

[24] M. Turk and A. Pentland. "Eigenfaces for Recognition". *Journal of Cognitive Neuroscience* 3.1 (1991).

[25] M.A.O. Vasilescu and D. Terzopoulos, eds. *Multilinear Analysis of Image Ensembles: TensorFaces*. Copenhagen: European Conference on Computer Vision, 2002.

[26] M.A.O. Vasilescu and D. Terzopoulos, eds. *Multilinear Image Analysis for Facial Recognition*. Quebec City: International Conference on Pattern Recognition, 2002.