Alma Mater Studiorum - Università di Bologna

DOTTORATO DI RICERCA IN MATEMATICA
Ciclo 34

Settore Concorsuale: 01/A5 - ANALISI NUMERICA
Settore Scientifico Disciplinare: MAT/08 - ANALISI NUMERICA

Tensor-Train decomposition for image classification problems

Presentata da: Domitilla Brandoni

Coordinatore Dottorato
Prof.ssa Valeria Simoncini

Supervisore
Prof.ssa Valeria Simoncini

Co-supervisore
Dott.ssa Margherita Porcelli

Esame finale 2022

Spero che un giorno tu abbia il coraggio di scappare da ogni cosa che ti rende infelice
(La città incantata, Hayao Miyazaki)

A chi mi ha sostenuta in questo percorso.

Contents

Introduction **i**

Notation and preliminary definitions **v**

1 Tensor tools **1**
1.1 Tensor computations 1
1.2 Tensor decompositions 6
1.2.1 Canonical Polyadic (CP) decomposition 6
1.2.2 The Higher-Order Singular Value Decomposition (HOSVD) 9
1.2.3 Tensor-Train (TT) decomposition 15

2 Image classification with tensor models **25**
2.1 Least Squares classification algorithm 26
2.2 HOSVD classification algorithm 27
2.3 Tensor-Train classification algorithm 30
2.3.1 Higher-order classification algorithm 32
2.4 Numerical experiments 33
2.4.1 Implementation details 34
2.4.2 Classification performance 34
2.4.3 A classification test in higher dimensional setting 36
2.4.4 Numerical experiments with truncated methods 37
2.4.5 Performance using statistical classification measures 38
2.4.6 Closing considerations 39

3 Proximal gradient methods **41**
3.1 The proximal gradient method 41
3.2 The block proximal gradient method 46
3.3 The Proximal Alternating Linearized Minimization (PALM) algorithm . 49

4 The spectral Proximal Alternating Linearized Minimization (sPALM) algorithm **57**
4.1 The spectral steplength 57
4.2 The spectral PALM algorithm 59

5 Matrix and tensor Dictionary Learning (DL) problem **63**
5.1 Matrix Dictionary Learning problem 65
5.1.1 Sparse coding 66
5.1.2 Dictionary update 68
5.2 Tensor methods 70
5.2.1 K-HOSVD 70
5.2.2 HO-SuKro 71
5.2.3 GRADTENSOR 73
5.2.4 K-CPD 75

6 Proximal methods for Dictionary Learning **79**
6.1 Proximal matrix methods 79
6.2 Tensor proximal methods for a new Tensor-Train formulation of the DL problem 86

7 Dictionary Learning for image classification **93**
7.1 The classification problem 93
7.2 Numerical experiments 96
7.2.1 Implementation details 96
7.2.2 Preliminary tests on the matrix DL formulation 97
7.2.3 Numerical experiments on the tensor DL formulation 99
7.2.4 Truncated approach for memory saving 100
7.2.5 A classification example in 4D setting 102
7.3 Closing considerations 105

Conclusions **107**

A Description of the databases **109**

Introduction

The interest around automatic object classification has been growing in these years thanks to the large amount of data available and the swift increase of computing performance [41]. A great effort has been put in reproducing the human ability of recognizing objects in different conditions (i.e. illumination, view point, occlusion, scale and background variations), with consequences in several applications such as medical imaging, driverless cars and security services [84, 75]. This is done by algorithmically *teaching* the computer how to automatically recognize different quantities, and it is at the core of machine learning methodologies. All the algorithms used to address this task are known as classification algorithms and are based on the detection of features that can help *classify* an object, i.e., recognize that a certain object belongs to a well defined class (e.g., an image represents a dog rather than an airplane). Object and image classification is part of the more general area of artificial intelligence, which consists of developing algorithmic and engineering strategies for making automatic decisions and operations. From a mathematical point of view, widely different approaches have been employed and further developed, from logistic regression to neural networks [36], and a major role is played by the underlying algebraic tools used to represent and approximate the processed data.

In this thesis we examine two different tensor-based models for image classification, in which tensors are used both to represent the data and to construct the reference dataset for further analysis and classification. It has been acknowledged in many applications that tensors allow to preserve the multidimensional structure of the data and the neighboring relations among image pixels [18, 45]. The intensive analytical and algorithmic work in tensor decompositions in the past few decades [48] has led to the development of different successful procedures that rely on different ways to represent and approximate a tensor. Among the available techniques, we have considered Tensor-Train (TT) decomposition [60], whose memory requirements do not seem to suffer from the so-called "curse of dimensionality", that is the exponentially growing memory requirements as the data dimensions increase. In addition, the quality of TT approximations does not significantly deteriorate when using truncation strategies to further limit memory consumptions.

Before tensors, matrix-based tools have been extensively used for image processing. Due to its optimality properties, the Singular Value Decomposition (SVD) has been the major workhorse for data compression and image classification (see [28] and references therein). However, transforming image datasets into matrices requires squeezing all information of the data into two dimensions: each image is vectorized so as to fit into

a single vector, and all features such as illumination, viewpoint, object class, etc. are mixed together. The use of multidimensional tensors allows one to preserve both the pixel two-dimensional structure of the image and the different inherent image properties, strongly motivating the broad use of tensor methodologies in image processing, and in particular for classification purposes, see, e.g., [28, 67, 78].

The first model we study in this thesis determines a reduced TT-based dataset that can be used to classify a new object. The TT decomposition determines basis vectors that allow to capture the main attributes of the original database in a reduced space, while being able to serve as an accurate reference for each object class. As an original contribution of this thesis, we derive a new Tensor-Train classification algorithm. In addition to coping with the curse of dimensionality, the approach shows great performance in terms of CPU time and classification success rate irrespective of the image size over the number of images in each class, overcoming related problems occurring with other (matrix) classification methods.

The second part of the thesis is devoted to the study of another mathematical model that can be used to address classification tasks: Dictionary Learning (DL). This process aims to determine a sufficiently descriptive reference set – called dictionary and described by a large matrix D – of the considered data population, and a representation of a data sample as a *sparse* linear combination of the columns of this dictionary matrix. The dictionary can be fixed or learned from data. In the first approach D can be determined using the characteristics of the problem and only the coefficients of the sparse linear combination need to be determined. In the second approach both the data and the variables are learnt from data. This makes the model more flexible and appropriate for different types of data going from signal to image processing. In the case one wants to represent a collection of p observations sets $Y \in \mathbb{R}^{n\times p}$, the sparse representations are collected in a matrix $X \in \mathbb{R}^{k\times p}$ so that

$$Y \approx DX,$$

where the approximation is required to be accurate according to some measure. Both the sparsity constraint and the nonlinearity due to the learnt dictionary make the underlying mathematical problem extremely difficult to treat. The most popular approaches known in literature, see [26] and references therein for a review, employ an iterative optimization strategy that alternates the updating of D by fixing X (*dictionary update*) and then the updating X by fixing D (*sparse coding*). The main drawback of these approaches is the lack of convergence guarantees mainly due to the use of greedy algorithms ([74]) for the sparse coding step. To overcome this problem, the Proximal Alternating Linearized Minimization (PALM) algorithm can be used. Developed in [9], it is specifically designed for general nonconvex and nonsmooth composite optimization problems that includes several DL formulations, such as [2]. PALM is an iterative block coordinate algorithm also relying on an alternating procedure. In each variable block, a gradient step is performed on the regular part of the function followed by a proximal step on the nonsmooth part. Since this gradient method is based on first order information, conver-

gence can be slow. Thus, we propose a new proximal approach, named spectral PALM (sPALM), with global guaranteed convergence and the use of second order information in the gradient step. The new approach is particularly well suited for the DL problem.

One of the main issues when dealing with large data is limiting memory requirements, since the dictionary D can be very memory consuming. To this end we propose a new Tensor-Train formulation of the DL problem in which the dictionary D is replaced with its Tensor-Train decomposition. This allows a more compact representation of data leading to memory savings without spoiling the classification performance. Once the sparse representation is available, it can be used for classification purposes with satisfactory success rate.

The results in this work open a venue to the use of more general, also additive, tensor structures in the dictionary term D, see,e.g., [22, 34], for early explorations, that may be able to better represent hidden properties in the data. This may also lead to new interpretations of the sparsity constraints and a more effective design of the current matrix X.

The original contribution of this thesis in the context of Tensor-Train based classification algorithms for image classification problems is based on the works [15, 16] and is contained in Chapters 2, 4, 6 and 7.

Notation and preliminary definitions

We introduce some notation and we recall definitions that will be used throughout the thesis.

a	vector
A	rectangular matrix
A^T	transpose of the matrix A
$\mathcal{A}$	higher-order tensor
$\text{int}(\Omega)$	interior of the set Ω
∇f	gradient of the function f
$\|\cdot\|_F$	Frobenius norm
$\|\cdot\|_2$	Euclidean norm
$\|\cdot\|_0$	zero norm
$\langle\cdot,\cdot\rangle$	scalar product

Table 1: Table of notation.

Definition 1. *Given two matrices $A \in \mathbb{R}^{m_1 \times m_2}$ and $B \in \mathbb{R}^{n_1 \times n_2}$ the Kronecker product of A and B is denoted by $A \otimes B \in \mathbb{R}^{m_1 n_1 \times n_2 m_2}$ and is defined as*

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \dots & a_{1m_2}B \\ a_{21}B & a_{22}B & \dots & a_{2m_2}B \\ \vdots & & & \vdots \\ a_{m_1 1}B & a_{m_1 2}B & \dots & a_{m_1 m_2}B \end{bmatrix}.$$

Definition 2 ([5, p. 14])**.** *Given a function $f : \mathbb{R}^n \to (-\infty, +\infty]$, its domain is defined as*

$$\text{dom}(f) = \{x \in \mathbb{R}^n \; s.t. \; f(x) < +\infty\}.$$

Definition 3 ([5, p. 14])**.** *A function $f : \mathbb{R}^n \to [-\infty, +\infty]$ is called proper if it does not attain the value $-\infty$ and if there exists at least one $x \in \mathbb{R}^n$ such that $f(x) < +\infty$, that is dom(f) is non-empty.*

Definition 4 ([5, Definition 5.1])**.** *Let $f : \mathbb{R}^n \to (-\infty, +\infty]$ and $L_f \geq 0$. Then f is said to be L_f-smooth over $\Omega \subseteq \mathbb{R}^n$ if it is differentiable over Ω and satisfies the following inequality*

$$\|\nabla f(x) - \nabla f(y)\|_2 \leq L_f \|x - y\|_2$$

for all x, $y \in \Omega$.

From this definition if a function f is L_f-smooth, then it is also $\bar{L}_f$-smooth with any $\bar{L}_f > L_f$. We refer to $\bar{L}_f$ as *smoothness parameter* of the function f and we will call *Lipschitz constant* the smallest possible smoothness parameter of a given function.

Definition 5. *Given a non-empty and closed set $\Omega \subset \mathbb{R}^n$, the indicator function $\delta_\Omega : \mathbb{R}^n \to (-\infty, +\infty]$ for all $a \in \mathbb{R}^n$ is given by*

$$\delta_\Omega(a) = \begin{cases} 0 & \text{if } a \in \Omega, \\ +\infty & \text{otherwise.} \end{cases}$$

Chapter 1

Tensor tools

In this chapter we describe some basic aspects of tensor computations and we explore three different tensor decompositions that will be used throughout this thesis.

1.1 Tensor computations

Definition 1.1.1 ([48, p. 455]). *A tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ is an element of the tensor product of N vector spaces, each of which has its own coordinate system. The order of a tensor is the number of its dimensions, also known as ways or modes.*

In a nutshell an Nth-order tensor can be defined as an N dimensional array. For example, a vector is a first-order tensor and a matrix is a second-order tensor. The *fibers* of a tensor are defined by fixing every index but one and correspond to matrix rows and columns in a higher-order setting. They are always considered as column vectors when extracted from the original tensor. Two-dimensional sections of a tensor are called *slices*, that are defined by fixing all but two indices [48]. The slices of a third-order tensor are usually denoted by $\mathcal{A}_{i,:,:}$ (horizontal slices), $\mathcal{A}_{:,j,:}$ (lateral slices), $\mathcal{A}_{:,:,k}$ (frontal slices).

Definition 1.1.2 ([48, p. 458]). *The inner product of two Nth-order tensors $\mathcal{A}$, $\mathcal{B} \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_N}$ is the sum of the products of their entries, i.e.*

$$\langle \mathcal{A}, \mathcal{B} \rangle = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_n=1}^{I_N} a_{i_1,i_2,\ldots,i_n} b_{i_1,i_2,\ldots,i_n} \tag{1.1}$$

The norm associated to this scalar product is the Frobenius norm $\|A\|_F^2 = \langle \mathcal{A}, \mathcal{A} \rangle$. Notice that if $\mathcal{A}$ and $\mathcal{B}$ are second-order tensors, i.e. matrices, (1.1) reduces to the standard definition of the matrix scalar product.

In several tensor decompositions the concept of *unfolding* a tensor into a matrix is crucial. The unfolding, also known as *matricization* or *flattening*, consists in reordering the elements of an N-way array into a matrix. In this thesis we consider two different

types of unfolding of a tensor $\mathcal{A} \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_N}$, that we will refer to as *mode-n unfolding* and *reshape*.

The mode-n unfolding

Given an Nth-order tensor $\mathcal{A} \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_N}$, its mode-$n$ unfolding ([48]), denoted by $A_{(n)}$, arranges the mode-n fibers to be the columns of the resulting matrix. More precisely, the tensor element $(i_1, i_2, \ldots, i_n)$ is mapped to the matrix element (i_n, j), where

$$j = 1 + \sum_{\substack{k=1 \\ k \neq n}}^{N} (i_k - 1) j_k \qquad j_k = \prod_{\substack{m=1 \\ m \neq n}}^{k-1} d_m. \tag{1.2}$$

Different orderings of the columns for the mode-n unfolding can be used as long as the operations defined on the flattened tensors are consistent [47, 48].

The reshape

The reshape ([18, 60]) of a tensor $\mathcal{A} \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_N}$, denoted here as $A_{[k]}$, is a $(d_1 d_2 \ldots d_k)$ by $(d_{k+1} d_{k+2} \ldots d_N)$ matrix, whose elements are taken columnwise from $\mathcal{A}$, that is

$$A_{[k]}(\overline{i_1 \ldots i_k}, \overline{i_{k+1} \ldots i_N}) = \mathcal{A}(i_1, \ldots, i_k, i_{k+1}, \ldots i_N). \tag{1.3}$$

The multi-index $\overline{i_1 \ldots i_N}$ is defined here as

$$\overline{i_1 \ldots i_N} = i_1 + (i_2 - 1) d_1 + \cdots + (i_N - 1) d_1 \ldots d_{N-1}.$$

Other conventions for defining the multi-index can be used, see for example [19]. Given an Nth-order tensor $\mathcal{A}$, it is possible to have N unfoldings and $N-1$ reshapes. Furthermore the mode-1 unfolding of an Nth-order tensor is equivalent to the reshape along the first mode.

Remark 1.1.1. *Given a tensor $\mathcal{A} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$, then*

$$\left(\mathcal{A}_{[N-1]}\right)^T = \mathcal{A}_{(N)}.$$

To prove this equivalence is sufficient to observe that the multi-index $\overline{i_1 \ldots i_{N-1}}$ is equivalent to the column index j associated to the mode-N unfolding defined in (1.2).

The next example illustrates the differences between the mode-n unfolding and the reshape of a third-order tensor $\mathcal{A} \in \mathbb{R}^{3 \times 2 \times 3}$.

Example 1.1.1. *Consider the tensor $\mathcal{A} \in \mathbb{R}^{3 \times 2 \times 3}$, whose frontal slices are given by*

$$A_1 = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{pmatrix}, \quad A_3 = \begin{pmatrix} 13 & 14 \\ 15 & 16 \\ 17 & 18 \end{pmatrix}.$$

Then the three mode-n unfoldings are given by

$$A_{(1)} = \begin{pmatrix} 1 & 2 & 7 & 8 & 13 & 14 \\ 3 & 4 & 9 & 10 & 15 & 16 \\ 5 & 6 & 11 & 12 & 17 & 18 \end{pmatrix},$$

$$A_{(2)} = \begin{pmatrix} 1 & 3 & 5 & 7 & 9 & 11 & 13 & 15 & 17 \\ 2 & 4 & 6 & 8 & 10 & 12 & 14 & 16 & 18 \end{pmatrix},$$

$$A_{(3)} = \begin{pmatrix} 1 & 3 & 5 & 2 & 4 & 6 \\ 7 & 9 & 11 & 8 & 10 & 12 \\ 13 & 15 & 17 & 14 & 16 & 18 \end{pmatrix}.$$

The reshapes along mode-1 and mode-2 are given by

$$A_{[1]} = \begin{pmatrix} 1 & 2 & 7 & 8 & 13 & 14 \\ 3 & 4 & 9 & 10 & 15 & 16 \\ 5 & 6 & 11 & 12 & 17 & 18 \end{pmatrix}, \quad A_{[2]} = \begin{pmatrix} 1 & 7 & 13 \\ 3 & 9 & 15 \\ 5 & 11 & 17 \\ 2 & 8 & 14 \\ 4 & 10 & 16 \\ 6 & 12 & 18 \end{pmatrix},$$

As already mentioned, we can observe that $A_{(1)} = A_{[1]}$ *and* $A_{(3)} = \left(A_{[2]}\right)^T$.

Now we introduce two important tensor operations: the n-mode product, which is a tensor by matrix operation and the $\binom{m}{n}$-product, which is a tensor by tensor operation.

Definition 1.1.3 ([48, p. 460])**.** *Given a tensor* $\mathcal{A} \in \mathbb{R}^{d_1 \times \cdots \times d_n \times \cdots \times d_N}$ *and a matrix* $U \in \mathbb{R}^{j \times d_n}$ *the n-mode matrix product of* $\mathcal{A}$ *with* U *is denoted by* $\mathcal{A} \times_n U \in \mathbb{R}^{d_1 \times \cdots \times d_{n-1} \times j \times d_{n+1} \times \cdots \times d_N}$ *and its entries are given by*

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

If $\mathcal{A} \in \mathbb{R}^{d_1 \times d_2}$ is a matrix and $U \in \mathbb{R}^{j \times d_1}$, then the 1-mode multiplication of $\mathcal{A}$ with U corresponds to the matrix multiplication

$$\mathcal{A} \times_1 U = U\mathcal{A} \qquad (UA)(i,j) = \sum_{i_1=1}^{d_1} u_{j,i_1} a_{i_1,i}.$$

Furthermore, if $\mathcal{A} \in \mathbb{R}^{d_1 \times d_2}$ and $V \in \mathbb{R}^{j \times d_2}$, the 2-mode multiplication is equivalent to matrix multiplication with V^T from the right:

$$\mathcal{A} \times_2 V = \mathcal{A}V^T \qquad (AV^T)(i,j) = \sum_{i_2=1}^{d_2} a_{i,i_2} v_{j,i_2}.$$

We also recall that the n-mode product satisfies the following important commutativity property ([48, p. 461])

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

where $\mathcal{A} \in \mathbb{R}^{d_1 \times \cdots \times d_n \times \cdots \times d_m \times \cdots \times d_N}$, $U \in \mathbb{R}^{j \times d_n}$ and $V \in \mathbb{R}^{j \times d_m}$. Therefore, in a series of multiplications the product order of distinct modes is irrelevant. If the modes are the same, i.e. $m = n$, then

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

Proposition 1.1.1 ([48]). *Let $\mathcal{A} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$ and $U_n \in \mathbb{R}^{j_n \times d_n}$ for $n = 1, \ldots, N$. Consider*

$$\mathcal{B} = \mathcal{A} \times_1 U_1 \times_2 \cdots \times_N U_N,$$

then the mode-n unfolding of $\mathcal{B}$ is given by

$$\mathcal{B}_{(n)} = U_n \mathcal{A}_{(n)} \left(U_N \otimes \cdots \otimes U_{n+1} \otimes U_{n-1} \otimes \cdots \otimes U_1\right)^T.$$

In a similar way to the n-mode product, in [18] the $\binom{m}{n}$-product of a tensor is introduced.

Definition 1.1.4 ([18]). *The $\binom{m}{n}$-product of a tensor $\mathcal{A} \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_N}$ with a tensor $\mathcal{B} \in \mathbb{R}^{j_1 \times j_2 \times \cdots \times j_M}$, such that $d_n = j_m$, is defined as*

$$\mathcal{C} = \mathcal{A} \times_n^m \mathcal{B},$$

where $\mathcal{C} \in \mathbb{R}^{d_1 \times \cdots \times d_{n-1} \times d_{n+1} \times \cdots \times d_N \times j_1 \times \cdots \times j_{m-1} \times j_{m+1} \times \cdots \times j_M}$. Its entries are given by

$$\mathcal{C}(i_1, \ldots, i_{n-1}, i_{n+1}, \ldots, i_N, j_1, \ldots j_{m-1}, j_{m+1}, \ldots, j_M) = \sum_{i=1}^{d_n} \mathcal{A}(i_1, \ldots, i_{n-1}, i, i_{n+1}, \ldots, i_N) \mathcal{B}(j_1, \ldots, j_{m-1}, i, j_{m+1}, \ldots, j_M).$$

The next result provides a relation between the n-mode product and the $\binom{m}{n}$-product.

Proposition 1.1.2. *Let $\mathcal{A} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ and $\mathcal{B} \in \mathbb{R}^{d_3 \times \cdots \times d_N}$, then*

$$\left(\mathcal{A} \times_3^1 \mathcal{B}\right)_{[N-1]} = \left(\mathcal{B} \times_1 \mathcal{A}_{[2]}\right)_{[N-2]}.$$

Proof. First, we write $\left(\mathcal{B} \times_1 \mathcal{A}_{[2]}\right)$ in the index form

$$\begin{aligned} \left(\mathcal{B} \times_1 \mathcal{A}_{[2]}\right)(\overline{i_1 i_2}, i_3, \ldots, i_N) &= \sum_{j=1}^{d_3} \mathcal{A}_{[2]}(\overline{i_1 i_2}, j) \mathcal{B}(j, i_3 \ldots, i_N) \\ &= \sum_{j=1}^{d_3} \mathcal{A}(i_1, i_2, j) \mathcal{B}(j, i_3 \ldots, i_N) = \left(\mathcal{A} \times_3^1 \mathcal{B}\right)(i_1, i_2, \ldots, i_N). \end{aligned} \tag{1.4}$$

The result follows by observing that $\overline{\overline{i_1 i_2} i_3 \ldots i_{N-1}} = \overline{i_1 \ldots i_{N-1}}$. □

Finally, we introduce one of the most important concepts in tensor computations: the *tensor rank*. Even if the definition of tensor rank is analogue to the definition of matrix rank, the properties of tensor and matrix ranks are quite different as stated in [48].

Definition 1.1.5 ([48, p. 458]). *Given N vectors $x^{(1)} \in \mathbb{R}^{d_1}, \ldots, x^{(N)} \in \mathbb{R}^{d_N}$, their outer product $x^{(1)} \circ x^{(2)} \circ \cdots \circ x^{(N)}$ is an Nth-order tensor $\mathcal{A} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$ whose entries are given by*

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

Definition 1.1.6 ([48, p. 458]). *An Nth-order tensor $\mathcal{A}$ is rank 1 if it can be written as the outer product of N vectors $x^{(1)}$, $x^{(2)}, \ldots, x^{(N)}$, i.e.*

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

Definition 1.1.7 ([24, Definition 2]). *The n-rank of $\mathcal{A}$ is the dimension of the vector space spanned by the n-mode vectors and is denoted by $\rho_n = rank_n(\mathcal{A})$.*

For the n-rank the following relation holds:

$$\operatorname{rank}_n(\mathcal{A}) = \operatorname{rank}(A_{(n)}). \tag{1.5}$$

Example 1.1.2. *Consider the tensor $\mathcal{A} \in \mathbb{R}^{6\times 2\times 3}$, whose frontal slices are given by*

$$A_1 = \begin{pmatrix} 1 & 2 \\ 2 & 4 \\ 3 & 6 \\ 4 & 8 \\ 5 & 10 \\ 6 & 12 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 11 & 12 \\ 13 & 14 \\ 15 & 16 \\ 17 & 18 \\ 19 & 20 \\ 21 & 22 \end{pmatrix}, \quad A_3 = \begin{pmatrix} 31 & 32 \\ 33 & 34 \\ 35 & 36 \\ 37 & 38 \\ 39 & 40 \\ 41 & 42 \end{pmatrix}.$$

Then the $1-$rank is not maximum and in particular $\rho_1 = 5$ since the first two columns of $\mathcal{A}_{(1)}$ are proportional. On the other hand, ρ_2 and ρ_3 are maximum and are equal to 2 and 3 respectively.

Definition 1.1.8 ([24, Definition 4]). *Given $\mathcal{A} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$ we define its rank as the minimal number of rank-one tensors that yield $\mathcal{A}$ as a linear combination and we denote it by $\rho = rank(\mathcal{A})$.*

Note that, as observed in [48], the problem of determining the rank of an Nth-order tensor is NP-hard. However for some tensors it is possible to compute exactly or estimate the rank (see [48, Tables 3.2-3.3]). The following remark provides an upper bound for the rank of a third-order tensor.

Remark 1.1.2 ([48]). *If $\mathcal{A} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ is a third-order tensor, then the following inequality holds*

$$rank(\mathcal{A}) \leq \min\{d_1 d_2,\ d_1 d_3,\ d_2 d_3\}.$$

1.2 Tensor decompositions

This section is devoted to the description of three different tensor decompositions: Canonical Polyadic decomposition (CP) which decomposes a tensor into a sum of rank one tensors, High-Order Singular Value Decomposition (HOSVD), which generalizes of the matrix SVD to N-mode tensors and Tensor-Train (TT) decomposition, which decomposes an N dimensional tensor in a product of third-order tensors.

1.2.1 Canonical Polyadic (CP) decomposition

The Canonical Polyadic (CP) decomposition decomposes an Nth-order tensor $\mathcal{X} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$ in a finite sum of rank one tensors, that is

$$\mathcal{X} = \sum_{r=1}^{\rho} a_r^{(1)} \circ a_r^{(2)} \circ \cdots \circ a_r^{(N)}, \tag{1.6}$$

where $a_r^{(i)} \in \mathbb{R}^{d_i}$, $\forall i = 1, \ldots, N$, $\forall r = 1, \ldots, \rho$. In several contexts, it can be useful to consider unit norm factors $a_r^{(i)}$. Thus, (1.6) can be written as follows.

$$\mathcal{X} = \sum_{r=1}^{\rho} \lambda_r a_r^{(1)} \circ a_r^{(2)} \circ \cdots \circ a_r^{(N)}, \tag{1.7}$$

where $\lambda_r \geq 0$ contains the information of the norms of $a_r^{(i)}$ and are decreasingly ordered (i.e. $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_\rho$). A visualization of the CP decomposition for a third-order tensor is given in Figure 1.1.

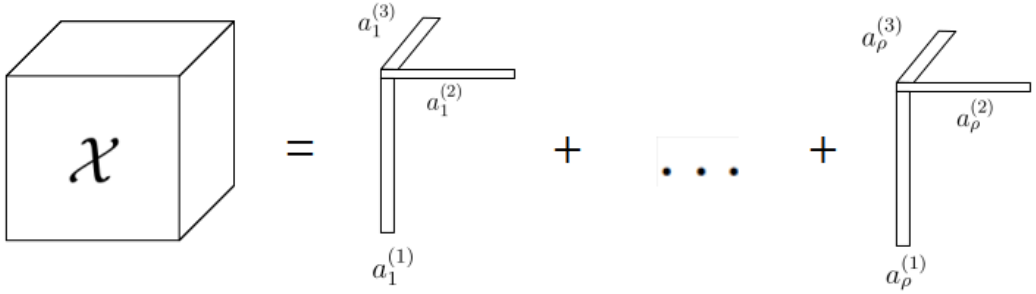


Figure 1.1: Visualization of the CP decomposition for a third-order tensor.

Recalling Definition 1.1.8, ρ is the rank of $\mathcal{X}$. Thus, determining the exact CP decomposition of a tensor $\mathcal{A}$ is strictly related to determining its rank which is an NP-hard problem. To avoid the exact computation of ρ, some procedures determine CP approximations with increasing ranks R until the approximation of $\mathcal{X}$ is sufficiently good ([48]). Before moving to the numerical strategies used to determine the CP Decomposition, we introduce two important matrix operations that will be useful in the discussion.

Definition 1.2.1 ([48, p. 462]). *Given two matrices $A \in \mathbb{R}^{d_1 \times j}$ and $B \in \mathbb{R}^{d_2 \times j}$ the Kathri-Rao product between A and B is a real $d_1 d_2 \times j$ matrix denoted by $A \odot B$ and its entries are given by*

$$A \odot B = (a_1 \otimes b_1, \ \dots \ , a_j \otimes b_j) .$$

The vectors a_k, b_k, $k = 1, \dots, j$ denote the columns of A and B, respectively.

Definition 1.2.2 ([48, p. 462]). *Given two matrices A, $B \in \mathbb{R}^{d_1 \times d_2}$, their Hadamard product is denoted as $A * B$ and is defined as follows*

$$A * B = \begin{pmatrix} a_{11}b_{11} & \dots & a_{1d_2}b_{1d_2} \\ \vdots & & \vdots \\ a_{d_1 1}b_{d_1 1} & \dots & a_{d_1 d_2}b_{d_1 d_2} \end{pmatrix} .$$

As previously mentioned, determining the exact CP decomposition for a tensor can be difficult or even not possible. Thus, we consider the following setting. Given a tensor $\mathcal{X}$ we want to approximate it with a tensor $\tilde{\mathcal{X}}$ that has a CP structure of rank ρ, i.e.

$$\min_{\tilde{\mathcal{X}}} \left\| \mathcal{X} - \tilde{\mathcal{X}} \right\|_F , \qquad \text{where} \quad \tilde{\mathcal{X}} = \sum_{r=1}^{\rho} \lambda_r a_r^{(1)} \circ a_r^{(2)} \circ \cdots \circ a_r^{(N)}. \tag{1.8}$$

One of the most used approaches to compute the CP Decomposition of a tensor $\mathcal{X}$ is the alternating least squares (ALS). The underlying idea of this algorithm is to update one term of the decomposition while keeping the others fixed. More precisely, let $A^{(i)}$ be the matrix obtained by collecting the vectors $a_r^{(i)}$, $\forall r = 1, \dots, \rho$.

$$A^{(i)} = \left[a_1^{(i)}, \dots, a_\rho^{(i)} \right] \quad i = 1, \dots, N,$$

then the mode-n unfolding of a tensor $\mathcal{X}$ can be written as

$$\mathcal{X}_{(n)} = A^{(n)} \Lambda \left(A^{(N)} \odot \cdots \odot A^{(n+1)} \odot A^{(n-1)} \odot \cdots \odot A^{(1)} \right), \tag{1.9}$$

where $\Lambda \in \mathbb{R}^{\rho \times \rho} = \mathrm{diag}(\lambda_1, \ \dots, \ \lambda_\rho)$. Thus, $A^{(n)}$ can be determined as the least squares solution of (1.9). The whole procedure is summarized in Algorithm 1.

Notice that the ALS procedure does not guarantee the convergence to a global minimum or a stationary point of (1.8) but only the decrease of the objective function at each iteration ([48]). For further properties and a more detailed discussion on the CP decomposition we refer to [48] and references therein.

Example 1.2.1. *Let $\mathcal{X} \in \mathbb{R}^{n_i \times n_e \times n_p}$ be the tensor representing a database of black and white images of n_p handwritten digits, reshaped as vectors with n_i components. The number of images per digit is denoted as n_e and we refer to this variable as "expression". In this example we consider the MNIST database (see Appendix A) with $n_i = 784$, $n_e = 2676$ and $n_p = 10$. We approximate its CP decomposition with $\rho = 100$. In*

Algorithm 1 CP-ALS [48]

Require: Tensor $\mathcal{X} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$, tensor rank ρ, initial values for $A^{(1)}, \ldots, A^{(N)}$

1: **for** $it = 1, \ldots,$ **do**
2: **for** $n = 1, \ldots, N$ **do**
3: Set $V \leftarrow \left(A^{(1)T}A^{(1)}\right) * \cdots * \left(A^{(n-1)T}A^{(n-1)}\right) * \left(A^{(n+1)T}A^{(n+1)}\right) * \cdots * \left(A^{(N)T}A^{(N)}\right)$
4: Set $A^{(n)} = X^{(n)}\left(A^{(N)} \odot \cdots \odot A^{(n+1)} \odot A^{(n-1)} \odot \cdots \odot A^{(1)}\right) V^{\dagger}$
5: Compute the norm of the columns of $A^{(n)}$ and update Λ
6: **end for**
7: **end for**

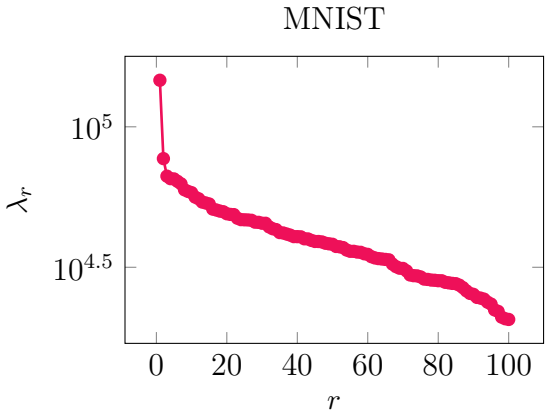


Figure 1.2: Value of λ_r, $r = 1, \ldots, \rho$ from the CP decomposition of the MNIST database $\mathcal{X}$.

Figure 1.2, the values of λ_r, for $r = 1, \ldots, \rho$ are summarized. A significant decrease can be observed between the first and the third value of λ, while for $r > 3$ the decrease is almost linear. However λ_1 and λ_ρ differ only by one order of magnitude suggesting that the original tensor $\mathcal{X}$ does not have a low rank structure. Recalling Remark 1.1.2, $rank(\mathcal{X}) \leq 7840$, which is much larger than the chosen rank ρ.

The CP decomposition of $\mathcal{X}$ can be used to approximate a specific digit $\hat{p}$ in an expression $\hat{e}$ as

$$\mathcal{X}(:, \hat{e}, \hat{p}) = \sum_{r=1}^{\rho} \lambda_r a_r^{(1)} \circ a_r^{(2)}(\hat{e}) \circ a_r^{(N)}(\hat{p}),$$

where $a_r^{(1)}$, $a_r^{(2)}$ and $a_r^{(3)}$ are unit norm vectors.

In Figure 1.3 we report the approximation of the zero digit ($\hat{p} = 1$) in the first expression ($\hat{e} = 1$) with $\rho = 5$, while in Figure 1.4a we report the approximation of the digit 8 ($\hat{p} = 9$) in expression 76 ($\hat{e} = 76$) using the same rank $\rho = 5$. It is worth to observe that the chosen value of the rank enables us to achieve a good visual approximation of the handwritten digit 0, that is easier to represent. On the other hand, for more difficult digits such as 8, a higher value of ρ should be considered, as shown in Figure 1.4b where

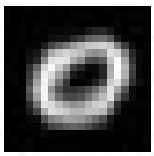
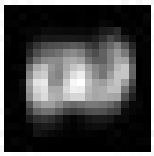
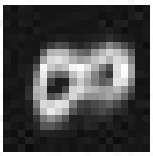


Figure 1.3: Approximation of the handwritten digit 0 in expression 1 using $\rho = 5$.



(a)



(b)

Figure 1.4: Approximation of the handwritten digit 8 in expression 76 using $\rho = 5$ (left) and $\rho = 10$ (right).

the approximation is computed with $\rho = 10$.

1.2.2 The Higher-Order Singular Value Decomposition (HOSVD)

The Higher-Order Singular Value Decomposition (HOSVD) decomposes an Nth-order tensor $\mathcal{A}$ into a core tensor $\mathcal{S}$, with the same order of $\mathcal{A}$, multiplied by N orthogonal matrices along the different modes. Thus, the HOSVD is a Tucker decomposition when the involved matrices are orthogonal and the core tensor has the property of all-orthogonality. More precisely, the Tucker decomposition, introduced in [77], decomposes an Nth-order tensor $\mathcal{A}$ into a core tensor $\mathcal{X}$ and N matrices $F_1, \ldots, F_N$ as follows

$$\mathcal{A} = \mathcal{X} \times_1 F_1 \times_2 \cdots \times_N F_N. \tag{1.10}$$

A well-known variant of this decomposition for third-order tensors is *Tucker2*, which corresponds to set a factor matrix equal to the identity (see [61, Definition 9]). For instance, given $\mathcal{A} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$, *Tucker2* can have the following form

$$\mathcal{A} = \mathcal{X} \times_1 F_1 \times_3 F_3, \tag{1.11}$$

where $\mathcal{X} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$. A complete description of the Tucker decomposition, its variants and the algorithms used to compute it can be found in [48]. In this Section we only

Algorithm 2 HOSVD [24]

Require: Tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$
1: **for** $n = 1, \ldots, N$ **do**
2: Consider the unfolding $A_{(n)}$ of $\mathcal{A}$
3: compute the SVD of $A_{(n)}$, $A_{(n)} = U\Sigma V^T$;
4: Set the n-th orthogonal matrix of the HOSVD $U^{(n)} = U$.
5: **end for**
6: Compute $\mathcal{S} = \mathcal{A} \times_1 \left(U^{(1)}\right)^T \times_2 \left(U^{(2)}\right)^T \times_3 \cdots \times_N \left(U^{(N)}\right)^T$

analyze the HOSVD and we refer to the work of De Lathawer, De Moor and Vandewalle in [24].

First, we report a result of [24], that provides the existence of the HOSVD for every N dimensional tensor and gives a constructive way to compute it. In the following $\mathcal{S}_{i_n=\alpha}$ denotes the subtensor obtained from $\mathcal{S}$ by fixing the index i_n equal to α. For example, if $\mathcal{S}$ is a third-order tensor, $\mathcal{S}_{i_2=\alpha} = \mathcal{S}_{:,\alpha,:}$.

Theorem 1.2.1 ([24, Theorem 2])**.** *Every Nth-order tensor $\mathcal{A} \in \mathbb{C}^{d_1 \times \cdots \times d_N}$ can be factorized as*

$$\mathcal{A} = \mathcal{S} \times_1 U^{(1)} \times_2 U^{(2)} \times_3 \cdots \times_N U^{(N)} \tag{1.12}$$

where

i. the matrices $U^{(n)} = (u_1^{(n)}\, u_2^{(n)}\, \ldots\, u_{I_n}^{(n)}) \in \mathbb{C}^{d_n \times d_n}$ are unitary and the vector $u_i^{(n)}$ is the i-th n-mode singular vector for $n = 1, \ldots, N$;

ii. the core tensor *$\mathcal{S} \in \mathbb{C}^{d_1 \times \cdots \times d_N}$ is a complex Nth-order tensor whose subtensors $\mathcal{S}_{i_n=\alpha}$, have the properties of*

(a) 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 (1.1)*

$$\langle \mathcal{S}_{i_n=\alpha}, \mathcal{S}_{i_n=\beta} \rangle = 0 \qquad \forall\, \alpha \neq \beta; \tag{1.13}$$

(b) ordering*:*

$$\|\mathcal{S}_{i_n=1}\|_F \geq \|\mathcal{S}_{i_n=2}\|_F \geq \cdots \geq \|\mathcal{S}_{i_n=I_n}\|_F \geq 0 \qquad \forall\, n = 1, \ldots, N, \tag{1.14}$$

where the Frobenius norms $\|\mathcal{S}_{i_n=i}\|_F$ are the n-mode singular values and are denoted by $\sigma_i^{(n)}$.

The proof of Theorem 1.2.1, reported in [24], relates the HOSVD of a tensor with the SVDs of its unfoldings. More precisely, each orthogonal matrix $U^{(n)}$, with $n = 1, \ldots, N$ is the matrix containing the left singular vectors of the ith unfolding of $\mathcal{A}$. This procedure can be used to derive Algorithm 2. The HOSVD of a third-order tensor is visualized in Figure 1.5.

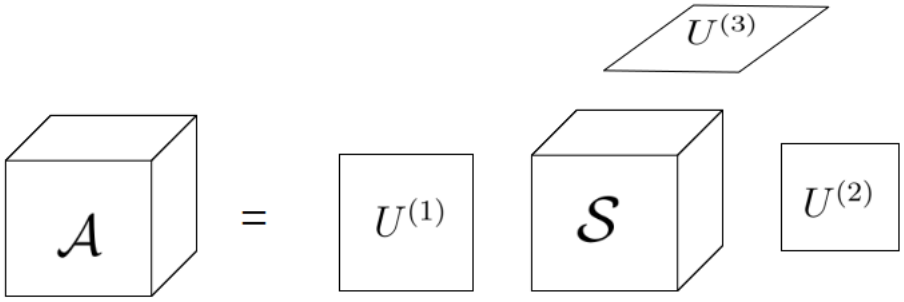


Figure 1.5: Visualization of the HOSVD for a third-order tenor.

Remark 1.2.1 ([24]). *Consider the HOSVD of a tensor $\mathcal{A}$ given in (1.12). Then, n-mode singular values are ordered in a decreasing order, i.e.*

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

Remark 1.2.2 ([24, Property 8]). *Consider the HOSVD of a tensor $\mathcal{A}$ given in (1.12). Then, $\|\mathcal{A}\|_F = \|\mathcal{S}\|_F$.*

Remark 1.2.3 ([24, Property 4]). *The n-mode singular values are uniquely defined. Furthermore for real-valued tensors, the n-mode singular vectors are unique up to a sign change or a multiplication with an orthogonal matrix.*

Remark 1.2.4 ([24, Property 5]). *If $A \in \mathbb{C}^{d_1 \times d_2}$, then (1.12) reduces to the SVD. From Theorem 1.2.1 we have*

$$A = S \times_1 U \times_2 V, \tag{1.15}$$

with $S \in \mathbb{C}^{d_1 \times d_2}$,$U \in \mathbb{C}^{d_1 \times d_1}$ and $V \in \mathbb{C}^{d_2 \times d_2}$. Now using the properties of the n-mode product we can rewrite (1.15) as

$$A = USV^T.$$

In particular, since S has the properties of all-orthogonality and ordering, S is a diagonal matrix containing the singular values of A in a decreasing order.

Remark 1.2.4 shows that the n-mode singular values and the n-mode singular vectors are the higher-order analogue of the left/right singular vectors and singular values, respectively.

It is also possible to express the HOSVD as an expansion of mutually orthogonal rank-one tensors ([24])

$$\mathcal{A} = \sum_{i_1=1}^{d_1} \sum_{i_2=1}^{d_2} \cdots \sum_{i_N=1}^{d_N} \mathcal{S}(i_1, i_2, \ldots, i_N) u_{i_1}^{(1)} u_{i_2}^{(2)} \ldots u_{i_N}^{(N)}. \tag{1.16}$$

Figure 1.6 shows one term of the sum (1.16) for a third-order tensor.

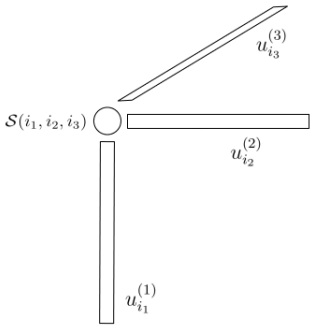


Figure 1.6: Visualization of one term of a triadic decomposition .

Consider a tensor $\mathcal{A} \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_N}$. As observed in [28], in some settings the dimension of one mode can be larger than the product of the dimensions of the other modes. For example, in image processing we may have that the dimension of the pixel space is larger than the product of the features dimensions, i.e. $d_1 > d_2 d_3 \dots d_N$. Then, it can be shown that the following condition for the core tensor holds

$$\mathcal{S}_{i_1=\alpha} = 0 \qquad \alpha > d_2 d_3 \dots d_N.$$

Thus, we can rewrite (1.12) by omitting the zero part of the core tensor

$$\mathcal{A} = \tilde{\mathcal{S}} \times_1 \tilde{U}^{(1)} \times_2 U^{(2)} \times_3 \cdots \times_N U^{(N)} \tag{1.17}$$

where $\tilde{\mathcal{S}} \in \mathbb{R}^{d_2 d_3 \dots d_N \times d_2 \times \cdots \times d_N}$ and $\tilde{U}^{(1)} \in \mathbb{R}^{d_1 \times d_2 d_3 \dots d_N}$. This variant of the HOSVD is known as *thin HOSVD*. Notice that if $A \in \mathbb{R}^{d_1 \times d_2}$ the thin HOSVD corresponds to the thin SVD. Indeed, when $d_1 > d_2$ the SVD can be written as

$$A = U \begin{pmatrix} S \\ 0 \end{pmatrix} V^T. \tag{1.18}$$

Then the matrix U can be partitioned in two blocks $U = (U_1, U_2)$, where $U_2^{(1)}$ in (1.18) is multiplied with the zero block of the diagonal matrix containing the singular values. Thus, using this observation, (1.18) can be written as

$$A = U_1 S V^T,$$

which is known as thin SVD of A.

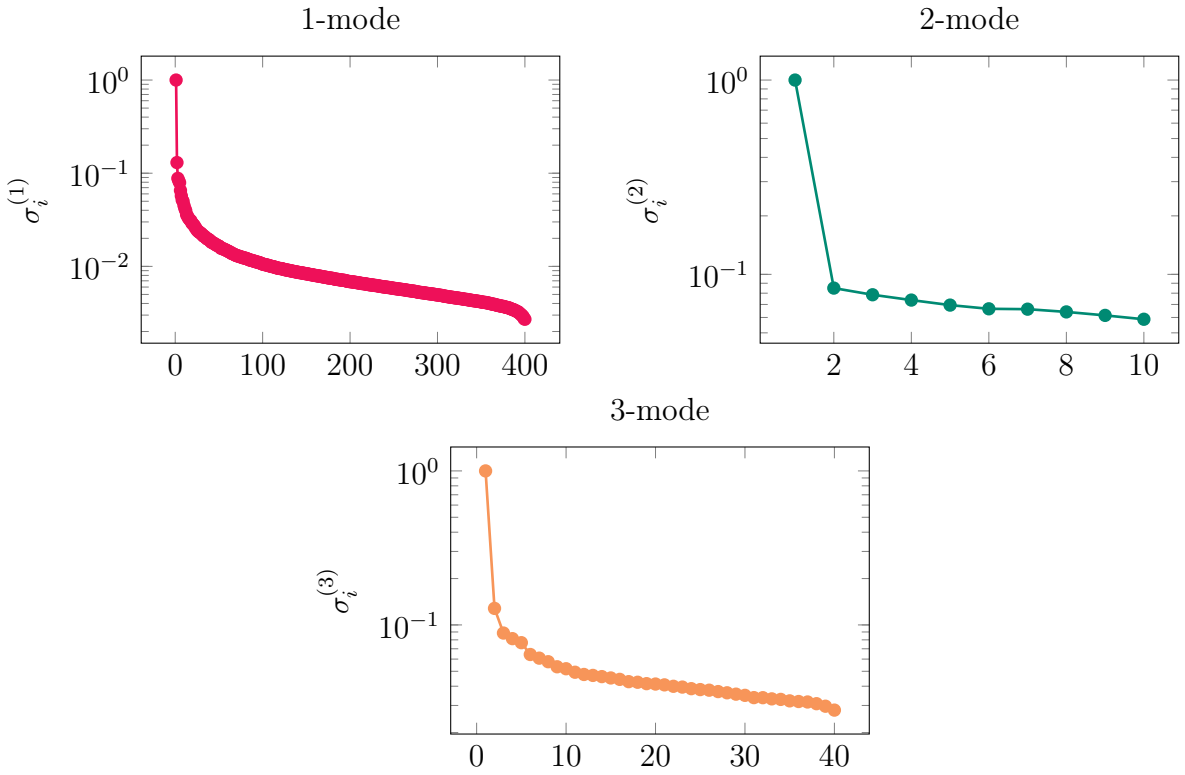


Figure 1.7: Singular values of the Orl database $\mathcal{A} \in \mathbb{R}^{10304\times10\times40}$.

Example 1.2.2. *Consider a third-order tensor* [1] $\mathcal{A} \in \mathbb{R}^{10304\times10\times40}$ *representing the Orl database (see Appendix A). We consider its HOSVD and we report in Figure 1.7 the singular values along the different modes. The number of singular values in the first mode is smaller than the number of pixels of each image* n_i*, because a thin HOSVD is considered. Thus, the number of 1-mode singular values is equal to* $n_e n_p$*. We also report in Figure 1.8 the first singular vector of* $\mathcal{A}$*,* $u_1^{(1)}$*. Similarly to the matrix setting, also in higher-order settings the first 1-mode singular vector contains the most important information of the database. Thus for a face database, such as the Orl, the first singular vector represents a face. This is due to the fact that* $U^{(1)}$ *contains the left singular vectors of the first unfolding of* $\mathcal{A}$ *which is indeed the matrix obtained by storing the vectorized version of each image in the database one after the other. Thus,* $u_1^{(1)}$ *is nothing but the first principal component of* $A_{(1)}$*, that is the linear combination of the images with the largest variance.*

[1] To further exploit the properties of a database of images a fourth-order tensor can be considered. In such a way each image can be represented as a matrix and the neighboring relations among pixels are better preserved. In the following chapters higher-order settings will be discussed, but in this example we just consider a third-order tensor by using only the first dimension for the pixel space, according to several works in the literature such as [78].

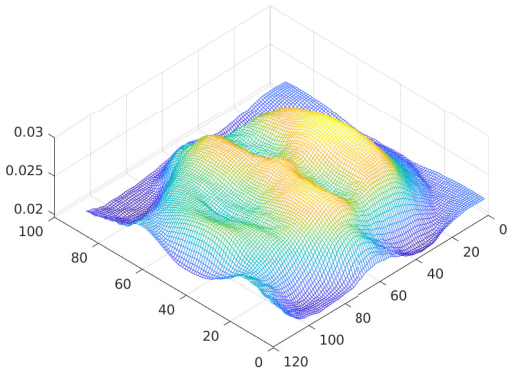


Figure 1.8: Matricization of the first 1-mode singular vector of $\mathcal{A} \in \mathbb{R}^{10304\times 10\times 40}$.

The truncated HOSVD

In several applications, we are interested in compressing information without loosing representative power. Thus, a truncated version of the HOSVD can be considered. Unfortunately, for the HOSVD the property of best approximation of the SVD cannot be derived. Nevertheless, the following theorem provides an upper bound on the approximation error.

Theorem 1.2.2 ([24, Property 10])**.** *Let $\mathcal{A} \in \mathbb{R}^{d_1\times d_2\times\cdots\times d_N}$ and consider its HOSVD defined in Theorem 1.2.1. Denote by $r_n (1 \leq n \leq N)$ the n-mode rank of $\mathcal{A}$ and define a tensor $\hat{\mathcal{A}}$ by discarding the n smallest n-mode singular values $\sigma^{(n)}_{d'_n+1}, \sigma^{(n)}_{d'_n+2}, \ldots, \sigma^{(n)}_{r_n}$ for given values of d'_n. Then*

$$\|\mathcal{A} - \hat{\mathcal{A}}\|_F^2 \leq \sum_{i_1=d'_1+1}^{r_1} \left(\sigma^{(1)}_{i_1}\right)^2 + \sum_{i_2=d'_2+1}^{r_2} \left(\sigma^{(2)}_{i_2}\right)^2 + \cdots + \sum_{i_N=d'_N+1}^{d_N} \left(\sigma^{(N)}_{i_N}\right)^2.$$

Proof. Denoting as $\hat{\mathcal{S}}$ the tensor obtained from $\mathcal{S}$ by setting equal to zero the n smallest

n-mode singular values, we have that

$$\begin{aligned}
\|\mathcal{A}-\hat{\mathcal{A}}\|_F^2 &= \|\mathcal{S}-\hat{\mathcal{S}}\|_F^2 \\
&= \sum_{i_1=1}^{r_1}\sum_{i_2=1}^{r_2}\cdots\sum_{i_N=1}^{r_N} s_{i_1,i_2,\ldots,i_N}^2 - \sum_{i_1=1}^{d_1'}\sum_{i_2=1}^{d_2'}\cdots\sum_{i_N=1}^{d_N'} s_{i_1,i_2,\ldots,i_N}^2 \\
&= \sum_{i_1=d_1'+1}^{r_1}\sum_{i_2=d_2'+1}^{r_2}\cdots\sum_{i_N=d_N'+1}^{r_N} s_{i_1,i_2,\ldots,i_N}^2 \\
&\leq \sum_{i_1=d_1'+1}^{r_1}\sum_{i_2=1}^{r_2}\cdots\sum_{i_N=1}^{r_N} s_{i_1,i_2,\ldots,i_N}^2 + \sum_{i_1=1}^{r_1}\sum_{i_2=d_2'+1}^{r_2}\cdots\sum_{i_N=1}^{r_N} s_{i_1,i_2,\ldots,i_N}^2 + \ldots \\
&+ \sum_{i_1=1}^{r_1}\sum_{i_2=1}^{r_2}\cdots\sum_{i_N=d_N'+1}^{r_N} s_{i_1,i_2,\ldots,i_N}^2 \\
&= \sum_{i_1=d_1'+1}^{r_1}\left(\sigma_{i_1}^{(1)}\right)^2 + \sum_{i_2=d_2'+1}^{r_2}\left(\sigma_{i_2}^{(2)}\right)^2 + \cdots + \sum_{i_N=d_N'+1}^{r_N}\left(\sigma_{i_N}^{(N)}\right)^2.
\end{aligned}$$

□

Example 1.2.3. *Consider the tensor $\mathcal{A}\in\mathbb{R}^{10304\times 10\times 40}$ described in Example 1.2.2. Define now $\hat{\mathcal{A}}$ by discarding the 50 smallest 1-mode singular values. The approximation error is $\|\mathcal{A}-\hat{\mathcal{A}}\|_F^2 = 3.796057\cdot 10^7$ and the sum of the discarded singular values is $3.796057\cdot 10^7$. In this example the approximation error is equal to the upper bound, since the truncation is performed just along one mode. Indeed, from the proof of Theorem 1.2.2 we can notice that if the truncation is performed just along one mode of the tensor, the upper bound becomes an equality.*

1.2.3 Tensor-Train (TT) decomposition

The Tensor-Train Decomposition was introduced by Oseledets in 2010 ([60]) to overcome one of the main drawbacks of the Tucker format. As already mentioned, for the HOSVD of a tensor $\mathcal{A}$ we have to store in memory the core tensor $\mathcal{S}$ and N unitary matrices. This becomes unattractive for large N. Thus, to prevent the curse of dimensionality, the *Tensor-Train Decomposition* decomposes an Nth-order tensor in a product of third-order tensors. Furthermore using an SVD based algorithm to determine the factors of the decomposition, the approximation properties of the HOSVD are preserved.

Let $\mathcal{A}\in\mathbb{R}^{d_1\times\cdots\times d_N}$, then its Tensor-Train Decomposition is given by ([60])

$$\mathcal{A}(i_1,\ldots,i_N) = G_1(i_1,:)\mathcal{G}_2(:,i_2,:)\ldots\mathcal{G}_{N-1}(:,i_{N-1},:)G_N(:,i_N) \tag{1.19}$$

where

i. $G_1 \in \mathbb{R}^{r_1 \times d_1}$, $G_N \in \mathbb{R}^{r_{N-1} \times d_N}$ and $\mathcal{G}_k \in \mathbb{R}^{r_{k-1} \times d_k \times r_k}$, for $k = 2, \ldots, N-1$ are called *TT-cores*,

ii. r_k for $k = 1, \ldots, N$ are called *TT-ranks* and $r_0 = r_N = 1$,

iii. $r = \max\limits_{1 \le k \le N} r_k$ is called *maximal TT-rank*.

In index form, the definition (1.19) can be written as

$$\mathcal{A}(i_1, \ldots, i_N) = \sum_{\alpha_1, \ldots, \alpha_{N-1}} G_1(i_1, \alpha_1) \mathcal{G}_2(\alpha_1, i_2, \alpha_2) \ldots G_N(\alpha_{N-1}, i_N)$$

Definition 1.2.3 ([60, p. 2297]). *A tensor $\mathcal{A} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$ is said to be in TT-format if its elements are given by (1.19).*

A classical visualization of the decomposition of a third-order tensor in index form is given in Figure 1.9.

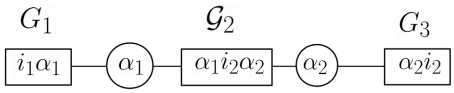


Figure 1.9: Visualization of the Tensor Train Decomposition in index form.

This graphical representation means the following. The rectangular nodes contain the indices i_k of the original tensor and at least one auxiliary index. The circular nodes 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 rectangular nodes and then perform the summation over all auxiliary indices. Since Figure 1.9 is similar to a train with carriages, the corresponding decomposition is called Tensor-Train Decomposition ([60]).

We next recall one of the main theorems for the Tensor-Train Decomposition, which also gives a constructive way to compute it.

Theorem 1.2.3 ([60, Theorem 2.1]). *If for each unfolding matrix $\mathcal{A}_{[k]}$ of form (1.3) of a tensor $\mathcal{A} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$ with*

$$rank(\mathcal{A}_{[k]}) = r_k, \tag{1.20}$$

then there exists a decomposition (1.19) with TT-ranks not higher than r_k.

Proof. Consider the first unfolding of $\mathcal{A}$ and its dyadic decomposition $\mathcal{A}_{[1]} = UV^T$, which can be written in the index form

$$\mathcal{A}_{[1]}(i_1, \overline{i_2 i_3 \ldots i_N}) = \sum_{\alpha_1 = 1}^{r_1} U(i_1, \alpha_1) V(\alpha_1, \overline{i_2 \ldots i_N}),$$

since $\text{rank}(\mathcal{A}_{[1]}) = r_1$. Then, the first TT-core is given by the matrix G_1 such that $G_1(i_1, \alpha_1) = U(i_1, \alpha_1)$. The matrix V can be expressed as

$$V = \mathcal{A}_{[1]}^T U (U^T U)^{-1} = \mathcal{A}_{[1]}^T W,$$

or in the index form

$$V(\alpha_1, \overline{i_2 \dots i_N}) = \sum_{i_1=1}^{d_1} \mathcal{A}(i_1, \dots, i_N) W(i_1, \alpha_1).$$

It is possible to treat the matrix V as an $(N-1)$th-order tensor

$$\mathcal{V}(\overline{\alpha_1 i_2}, i_3, \dots, i_N) = V(\alpha_1, \overline{i_2 \dots i_N}).$$

Now, consider the unfoldings $\mathcal{V}_{[k]}(\overline{\alpha_1 i_2 \dots i_k}, \overline{i_{k+1} \dots i_N})$ for $k = 2, \dots, N$. It can be shown that $\text{rank}(\mathcal{V}_{[k]}) \leq r_k$. Thus, since (1.20) holds,

$$\mathcal{A}_{[k]}(\overline{i_1 \dots i_k}, \overline{i_{k+1} \dots i_N}) = \sum_{\beta=1}^{r_k} F(\overline{i_1 \dots i_k}, \beta) G(\beta, \overline{i_{k+1} \dots i_N}).$$

Using the last expression we have

$$\begin{aligned} \mathcal{V}_{[k]}(\overline{\alpha_1 i_2 \dots i_{k+1}}, \overline{i_{k+2} \dots i_N}) &= \sum_{i_1=1}^{d_1} \mathcal{A}_{[k]}(\overline{i_1 \dots i_k}, \overline{i_{k+1} \dots i_N}) W(i_1, \alpha_1) \\ &= \sum_{i_1=1}^{d_1} \sum_{\beta=1}^{r_k} F(\overline{i_1 \dots i_k}, \beta) G(\beta, \overline{i_{k+1} \dots i_N}) W(i_1, \alpha_1) \qquad (1.21) \\ &= \sum_{\beta=1}^{r_k} H(\overline{\alpha_1 i_2 \dots i_k}, \beta) G(\beta, \overline{i_{k+1} \dots i_N}), \end{aligned}$$

where

$$H(\overline{\alpha_1 i_2 \dots i_k}, \beta) = \sum_{i_1=1}^{d_1} F(\overline{i_1 \dots i_k}, \beta) W(i_1, \alpha_1).$$

From (1.21) we have that $\text{rank}(\mathcal{V}_{[k]}) \leq r_k$, $\forall\, k = 1, \dots, N$. Now if we consider the unfolding $\mathcal{V}_{[1]}$, we have

$$\mathcal{V}_{[1]}(\overline{\alpha_1 i_2}, \overline{i_3 \dots i_N}) = \sum_{\alpha_2=1}^{r_2} G_2(\overline{\alpha_1 i_2}, \alpha_2) V'(\alpha_2, i_3 \dots i_N).$$

Then we set $(\mathcal{G}_2)_{[2]} = G_2$ and we iterate this process to find the other core tensors $\mathcal{G}_k$, for $k = 3, \dots, N$. □

Algorithm 3 TT-SVD [76, p.31-32]

Require: Tensor $\mathcal{A} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$

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

$$N_l = d_1, \qquad N_r = \prod_{k=2}^{N} d_k;$$

2: create a copy of the original tensor $\mathcal{M}_1 = \mathcal{A}$;
3: unfold $\mathcal{M}_1$ into the computed dimensions: $M_1 = \texttt{reshape}(\mathcal{M}, [N_l, N_r])$;
4: compute the thin SVD of M_1, $M_1 = U\Sigma V^T$;
5: set the first core tensor $G_1 \coloneqq U$;
6: recompute $M_2 = \Sigma V^T = U^T M$ and let $[\sim, r_1] = \texttt{size}(U)$;
7: **for** $k = 2, \ldots, N-1$ **do**
8: Compute

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

9: set $M_k = \texttt{reshape}(M_k, [r_{k-1} \cdot N_l, N_r])$;
10: compute the thin SVD of M_k, $M_k = U\Sigma V^T$;
11: Set $G_k = \texttt{reshape}(U, [r_{k-1}, d_k, r_k])$;
12: Compute $M_{k+1} = U^T M_k$ and let $[\sim, r_k] = \texttt{size}(U)$.
13: **end for**
14: $G_N \coloneqq M_N$

The proof of Theorem 1.2.3 gives a way to compute the TT decomposition of an Nth-order tensor $\mathcal{A}$ (see Algorithm 3).

Remark 1.2.5. *Let $G_1 \in \mathbb{R}^{d_1 \times r_1}$ be the first core of the TT decomposition of a tensor $\mathcal{A} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$ computed using Algorithm 3. Then $G_1 = U^{(1)}$ where $U^{(1)}$ is the first factor matrix of the HOSVD of $\mathcal{A}$. This follows directly from the equivalence between the mode-1 unfolding and the reshape along the first mode.*

As stated in [18], using Definition 1.1.4 for the $\binom{m}{n}$-mode product, (1.19) can be written as

$$\mathcal{A} = G_1 \times_2^1 \mathcal{G}_2 \times_3^1 \mathcal{G}_3 \times_3^1 \cdots \times_3^1 G_N. \tag{1.22}$$

Thus, if $\mathcal{A}$ is a third-order tensor, then

$$\mathcal{A} = \mathcal{G}_2 \times_1 G_1 \times_3 G_3^T, \tag{1.23}$$

since

$$\begin{aligned}\mathcal{A}(i_1,i_2,i_3) &= G_1(i_1,:)\times_2^1 \mathcal{G}_2(:,i_2,:)\times_3^1 G_3(:,i_3) = \sum_{\alpha_1,\alpha_2} G_1(i_1,\alpha_1)\mathcal{G}_2(\alpha_1,i_2,\alpha_2)G_3(\alpha_2,i_3)\\ &= \sum_{\alpha_2}(\mathcal{G}_2\times_1 G_1)(i_1,i_2,\alpha_2)G_3(\alpha_2,i_3) = (\mathcal{G}_2\times_1 G_1\times_3 G_3^T)(i_1,i_2,i_3).\end{aligned}$$

A standard visualization of (1.23) is given in Figure 1.10.

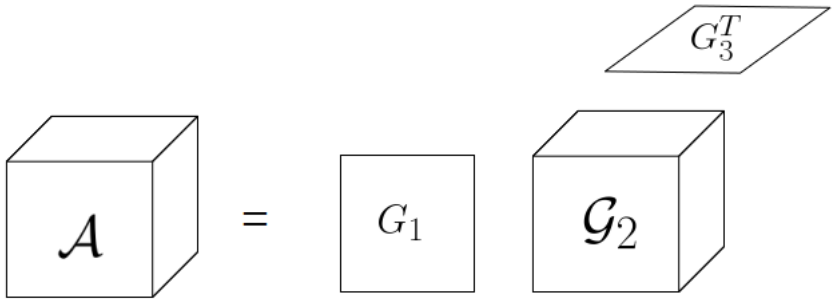


Figure 1.10: Visualization of the Tensor-Train Decomposition of a third-order tensor using the $\binom{m}{n}$-mode product.

We notice that for a third-order tensor the Tensor-Train Decomposition corresponds to the classical *Tucker2* decomposition (1.11). We recall that, in a third-order setting, this decomposition enables us to write a tensor $\mathcal{A}\in\mathbb{R}^{d_1\times d_2\times d_3}$ as the product of a core tensor and two matrices. In particular, using Tucker2, $\mathcal{A}$ can be decomposed as follows

$$\mathcal{A} = \mathcal{X}\times_1 U\times_3 V.$$

Thus, if we determine $\mathcal{X}$, U, V using the TT-SVD algorithm we obtain (1.23). We illustrate the Tensor-Train Decomposition of a third-order tensor for image data in the following example.

Example 1.2.4. *Let $\mathcal{A}\in\mathbb{R}^{10304\times 10\times 40}$ be the tensor representing the Orl database described in Example 1.2.2 and consider its Tensor-Train decomposition. Then the singular values of the matrices M_1 and M_2 defined in Algorithm 3, are summarized in Figure 1.11. According to Remark 1.2.5, the singular values of M_1 are equivalent to the 1-mode singular values of $\mathcal{A}$ reported in Example 1.2.2.*

We next derive some results on the relation between the TT decomposition of an Nth-order tensor and its reshapes.

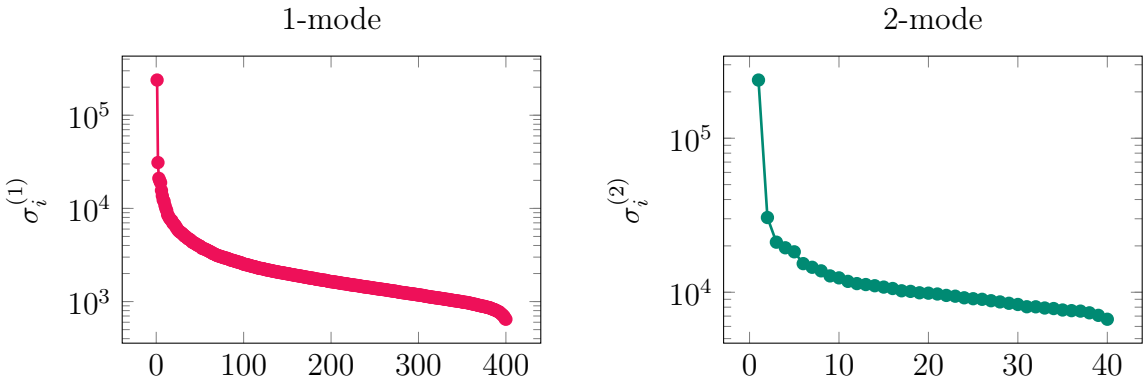


Figure 1.11: Singular values of the matrices M_1, M_2 for the Orl database $\mathcal{A} \in \mathbb{R}^{10304\times 10\times 40}$.

Proposition 1.2.1. *Let $\mathcal{A} \in \mathbb{R}^{d_1\times\cdots\times d_N}$ and consider its Tensor-Train decomposition with TT-cores $G_1, \mathcal{G}_2, \ldots, G_N$. Then the reshape along the mode $N-1$ of $\mathcal{A}$ is given by*

$$\mathcal{A}_{[N-1]} = (I_{d_2\ldots d_N} \otimes G_1)\left(I_{d_3\ldots d_{N-1}} \otimes G_2\right)\ldots\left(I_{d_3\ldots d_{N-1}} \otimes G_{N-2}\right) G_{N-1}G_N, \tag{1.24}$$

where $G_n = (\mathcal{G}_n)_{[2]}$, for $n = 2, \ldots, N-1$ and $I_k \in \mathbb{R}^{k\times k}$ denotes the identity matrix.

Proof. The Tensor-Train decomposition of $\mathcal{A}$ using the $\binom{m}{n}$-mode product as follows

$$\mathcal{A} = G_1 \times_2^1 \mathcal{G}_2 \times_3^1 \cdots \times_3^1 G_N.$$

Then denoting as $\mathcal{G}^{2,N-1} := \mathcal{G}_2 \times_3^1 \cdots \times_3^1 \mathcal{G}_{N-1}$, the reshape along the mode $N-1$ is given by

$$\begin{aligned}
\mathcal{A}_{[N-1]} &= \left(\mathcal{G}^{2,N-1} \times_1 G_1 \times_N G_N^T\right)_{[N-1]} = \left(\mathcal{G}^{2,N-1} \times_1 G_1 \times_2 I_{d_2} \times_3 \cdots \times_{N-1} I_{d_{N-1}} \times_N G_N^T\right)_{[N-1]} \\
&= \left(\left(\mathcal{G}^{2,N-1} \times_1 G_1 \times_2 I_{d_2} \times_3 \cdots \times_{N-1} I_{d_{N-1}} \times_N G_N^T\right)_{(N)}\right)^T \\
&= (I_{d_2\ldots d_{N-1}} \otimes G_1)\left(\left(\mathcal{G}^{2,N-1}\right)_{(N)}\right)^T G_N.
\end{aligned}$$

Now we consider the mode-N unfolding of $\mathcal{G}^{2,N-1}$. The idea is to separate one TT core at a time using Proposition 1.1.2.

$$\begin{aligned}
\left(\mathcal{G}_{(N)}^{2,N-1}\right)^T &= \left(\mathcal{G}_2 \times_3^1 \mathcal{G}^{3,N-1}\right)_{[N-1]} = \left(\mathcal{G}^{3,N-1} \times_1 G_2\right)_{[N-2]} \\
&= \left(\left(\mathcal{G}^{3,N-1} \times_1 G_2 \times_2 I_{d_3} \times_3 \cdots \times_{N-1} I_{d_{N-1}}\right)_{(N-1)}\right)^T \\
&= (I_{d_3\ldots d_{N-1}} \otimes G_2)\left(\left(\mathcal{G}^{3,N-1}\right)_{(N-1)}\right)^T.
\end{aligned}$$

Repeating this procedure for each TT-core, we obtain the result. □

In case of a third-order tensor $\mathcal{A} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$, using Proposition 1.2.1, we have that

$$\mathcal{A}_{[2]} = (I_{d_2} \otimes G_1)\, G_2 G_3, \tag{1.25}$$

where $G_2 = (\mathcal{G}_2)_{[2]}$ and $I_{d_2} \in \mathbb{R}^{d_2 \times d_2}$ is the identity matrix.

Proposition 1.2.2. *Let $\mathcal{A} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ and consider its Tensor-Train decomposition with TT-cores G_1, $\mathcal{G}_2$, G_3. Then the reshape along the first mode of $\mathcal{A}$ is given by*

$$\mathcal{A}_{[1]} = G_1\, (\mathcal{G}_2)_{[1]}\, (I \otimes G_3) \tag{1.26}$$

Proof. The result follows directly from Proposition 1.1.1 by observing that $\mathcal{A}_{(1)} = \mathcal{A}_{[1]}$. □

The truncated TT-SVD

As for the HOSVD, rank truncation strategies are commonly employed in the TT setting. Indeed, the best rank r_k approximation via SVD can be computed instead of the exact SVD. Then the introduced error can be estimated as shown in the following theorem.

Theorem 1.2.4 ([60, Theorem 2.2])**.** *Suppose that the unfoldings $\mathcal{A}_{[k]}$ of the tensor $\mathcal{A} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$ can be approximated by matrices of low rank $\hat{A}_k$*

$$\mathcal{A}_{[k]} = \hat{A}_k + E_k, \qquad \textit{rank}(\hat{A}_k) = r_k, \qquad \|E_k\| = \epsilon_k, \qquad k = 1, \ldots, N-1. \tag{1.27}$$

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

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

Following Theorem 1.2.4, we can consider a variant of Algorithm 3 in which the truncated SVD instead of the exact one is computed. This truncated TT-SVD is reported in Algorithm 4.

Example 1.2.5. *Consider a tensor $\mathcal{A} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$ and its TT approximation $\mathcal{B}$, computed via the TT-SVD algorithm by keeping the first m singular values of the unfolding matrices $\mathcal{A}_{[k]}$. Then, for the approximation property of the SVD the error obtained for each unfolding is equal to the sum of the discarded singular values, i.e. $\epsilon_k^2 = \sum_{s=m+1}^{r_k} (\sigma_s^k)^2$. Thus, using Theorem 1.2.4*

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

where σ_s^k, $k = 1, \ldots, r_k$ are the singular values of $\mathcal{A}_{[k]}$.

Algorithm 4 Truncated TT-SVD [60, Algorithm 1]

Require: Tensor $\mathcal{A} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$, accuracy ϵ

1: Compute the truncation parameter $\delta = \frac{\epsilon}{\sqrt{d-1}} \|\mathcal{A}\|_F$

2: Compute the size of the first unfolding of $\mathcal{A}$, $\mathcal{A}_{[1]}$:

$$N_l = d_1, \qquad N_r = \prod_{k=2}^{N} d_k;$$

3: create a copy of the original tensor $\mathcal{M}_1 = \mathcal{A}$;

4: unfold $\mathcal{M}_1$ into the computed dimensions: $M_1 = \texttt{reshape}(\mathcal{M}_1, [N_l, N_r])$;

5: compute the truncated SVD of M_1, $M_1 = U_1 \Sigma_1 V_1^T + E_1$, where $\|E_1\|_F \leq \delta$ and $r_1 = \text{rank}(U_1)$;

6: set the first core tensor $G_1 \coloneqq U$;

7: compute $M_2 = \Sigma_1 V_1^T = U_1^T M$;

8: **for** $k = 2, \ldots, N-1$ **do**

9: Compute

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

10: set $M_k = \texttt{reshape}(M_k, [r_{k-1} \cdot N_l, N_r])$;

11: compute the truncated SVD of M_k, $M_k = U_k \Sigma_k V_k^T + E_k$, where $\|E_k\|_F \leq \delta$ and $r_k = \text{rank}(U_k)$;

12: Set $\mathcal{G}_k = \texttt{reshape}(U_k, [r_{k-1}, d_k, r_k])$;

13: Compute $M_{k+1} = U_k^T M_k$.

14: **end for**

15: $G_N \coloneqq M_N$

16: **return** TT-cores $\mathcal{G}_k \in \mathbb{R}^{r_{k-1} \times d_k \times r_k}$ for $k = 1, \ldots, N$ s.t. $\|\mathcal{A} - G_1 \times_2^1 \mathcal{G}_2 \times_3^1 \cdots \times_3^1 G_N\|_F \leq \epsilon$

Corollary 1.2.1 ([60, Corollary 2.4])**.** *Given a tensor $\mathcal{A} \in \mathbb{R}^{d_1 \times d_N}$ 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*

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

An example on the approximation error follows.

Example 1.2.6. *Consider $\mathcal{A} \in \mathbb{R}^{10304 \times 10 \times 40}$ and let $\mathcal{B}$ be the tensor in the TT-format obtained by using the TT-SVD algorithm with $r_1 = 350$. The approximation error is $\|\mathcal{A} - \mathcal{B}\|_F^2 = 3.796057 \cdot 10^7$ and the sum of the discarded singular values is $3.796057 \cdot 10^7$. As for the HOSVD, when the truncation is performed just along the first mode the inequality*

in Theorem 1.2.4 becomes an equality. Indeed, computing the truncated SVD of the first unfolding of $\mathcal{A}$ we obtain

$$\mathcal{A}_{[1]} = U_1 \Sigma V_1^T + E_1 = G_1 M_2 + E_1,$$

where E_1 is the error matrix obtained by discarding the smallest 50 singular values of $\mathcal{A}_{[1]}$ and G_1 is the matrix containing the first 350 left singular vectors of $\mathcal{A}_{[1]}$. Now using the properties of the Frobenius norm we have

$$\sqrt{\sum_{s=351}^{400} (\sigma_s^1)^2} = \|E_1\|_F = \|\mathcal{A}_{[1]} - \mathcal{M}_2 \times_1 G_1\|_F = \|\mathcal{A} - \mathcal{M}_2 \times_1 G_1\|_F = \|\mathcal{A} - \mathcal{G}_2 \times_3 G_3^T \times_1 G_1\|_F,$$

where the last equality holds since no truncation is performed on the second mode.

In many image applications one may be interested in different truncations depending on the mode of the tensor. For example, consider a database of images with n_i pixels of n_p persons in n_e expressions $\mathcal{A} \in \mathbb{R}^{n_i \times n_e \times n_p}$. Then, for classification problems, the truncation performed on the first mode may be different from the truncation performed on the second mode, since the "pixel" mode is usually more redundant than the "expression" mode. In such situations instead of using the same δ for each mode as in Algorithm 4, one can consider different truncation parameters for each TT-core. One possible choice is to truncate each U_k to its first ℓ_k columns according to a parameter π_k

$$\ell_k = \max_J \left\{ \frac{\sum_{j=1}^{J} \sigma_j(M_k)}{\sum_{j=1}^{I_i} \sigma_j(M_k)} \leq \pi_k \right\} \qquad k = 1, \ldots, N-1,$$

where π_k is such that $\ell_k \geq 1$ for $k = 1, \ldots, N-1$. This selection is in agreement with the truncation guidelines discussed in [20, 39, 49].

Chapter 2

Image classification with tensor models

The material of this chapter is based on the publication [16].

Numerical linear algebra tools, and primarily the Singular Value Decomposition (SVD) have been extensively used to automatically process images in computations for classification purposes, see, e.g., [28] and its references. In this context, a database of n_p persons in n_e expressions is stored as n_p distinct matrices $A_p \in \mathbb{R}^{n_i \times n_e}$, $p = 1, \ldots, n_p$, where n_i is the number of pixels of each image. SVD-based classification algorithms work pretty well when the number of image pixels is greater than the number of features, otherwise the classification performance drops significantly. This suggests that alternative strategies should be used, since the latter condition often occurs in realistic applications where very many images need to be analyzed.

More recently, it has been shown that multilinear algebra and the algebra of higher-order tensors (see, e.g., [48]) can offer a more powerful mathematical framework for analyzing and addressing the multi-factor structure of image ensembles. For instance, the Weizmann database in [78] - where TensorFaces are introduced - is composed by 28 male subjects in 15 poses, 4 illumination conditions and 3 expressions, and it is represented by a 5-way tensor. Then the HOSVD is used to classify the image of an unknown person. Other tensor-based approaches have been proposed in a large variety of contexts; for instance, in [39] Face Recognition is performed using tensor-tensor decompositions, while in [14] the classification problem is expressed using the Kronecker Product Equation (KPE) in a randomization context.

In this chapter we describe some known tensor-based classification algorithms, and we introduce a classification strategy associated with the Tensor-Train decomposition. This tensor decomposition has been shown to have great potential in multivariate function approximation and compression, and it is particularly amenable to truncation strategies yielding low-rank tensor approximations.

In the tensor setting, a database of images with n_i pixels, n_p persons and n_e expressions for each person is splitted in a *training set* and a *test set*. The first is composed by all the persons in $\tilde{n}_e < n_e$ expressions and is represented either as a third-order tensor $\mathcal{A} \in \mathbb{R}^{n_i \times \tilde{n}_e \times n_p}$ or as a fourth-order tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times \tilde{n}_e \times n_p}$ with $n_1 n_2 = n_i$. The second

Algorithm 5 Least Squares

1: **Input**: $z \in \mathbb{R}^{n_i}$ input image and $\mathcal{A}$.
2: **for** $e = 1, \ldots, n_e$ **do**
3: Solve for x_e in (2.1) and set $r_e = z - A^e x_e$ [1]
4: **end for**
5: $\hat{e} = \underset{e}{\text{argmin}}(\|r_e\|)$
6: $\hat{p} = \underset{p}{\text{argmax}}|x_{\hat{e}}(p)|$
7: **Output**: Classify z as person $\hat{p}$

is composed by all the persons in the remaining expressions and is used to validate the classification algorithm. More precisely, given an image z from the test set, represented by a vector in $\mathbb{R}^{n_i}$, we want to determine which of the n_p persons, the new image is closest to. To this end we define a distance $dist(z, \mathcal{A}_{:,:,p})$ of z from each person p of the given database, for $p = 1, \ldots, n_p$. In the following discussion the space "*expression* e" spanned by all images of the n_p persons in expression e is denoted as $\mathbf{E}^e$.

2.1 Least Squares classification algorithm

The Least Squares classification algorithm is a simple minded strategy that merely focuses on the comparison between the new image $z \in \mathbb{R}^{n_i}$ to be classified and the matrices obtained by considering all images in each given expression. More precisely, for each expression $e = 1, \ldots, n_e$ let $A^e := \mathcal{A}_{:,e,:}$, where the columns of A^e have been scaled to have unit Euclidean norm. Consider the following least squares problem

$$\min_x \|z - A^e x\|_2. \tag{2.1}$$

The solution of (2.1), x_e represents the coordinates in the space $\mathbf{E}^e$. Let $r_e = z - A^e x_e$ be the associated residual that indicates how far z is from being a linear combination of the columns of A^e and the distance of z from $\mathbf{E}^e$ is measured by $\|r_e\|_2$. Then compute

$$\hat{e} = \text{argmin}_e \|r_e\|_2,$$

to determine the space $\mathbf{E}^{\hat{e}}$ in which z is better represented and classify z as person $\hat{p}$, where $\hat{p} = \text{argmax}_p |x_{\hat{e}}(p)|$. Note that $\hat{p}$ indicates the column of $A^{\hat{e}}$ that contributes most in the linear combination $A^{\hat{e}} x_{\hat{e}}$. A version of this classification algorithm is given in Algorithm 5.

Problem (2.1) has a unique solution when $n_i > n_p$ and A^e has full column rank. This is quite common in image databases (see Appendix A). On the other hand when this condition is not fulfilled one can for example consider the unique solution $x_e^{(LS)}$ of (2.1)

[1] Each slice of $\mathcal{A}$, that is A^e, is normalized before solving (2.1).

that minimizes the Euclidean norm of $\|x\|_2$. In particular, $x_e^{(LS)}$ can be computed as $x_e^{(LS)} = (A^e)^T y$, where y is the solution of

$$A^e (A^e)^T y = z.$$

2.2 HOSVD classification algorithm

Using the HOSVD described in section 1.2.2, the tensor $\mathcal{A} \in \mathbb{R}^{n_i \times n_e \times n_p}$ can be written as follows

$$\mathcal{A} = \mathcal{S} \times_i F \times_e G \times_p H, \tag{2.2}$$

where $\times_i$, $\times_e$, $\times_p$ are the 1-mode, 2-mode, 3-mode multiplications, respectively[2]; F, G and H are the factor matrices of the HOSVD and $\mathcal{S}$ is the core tensor. Notice that $G \in \mathbb{R}^{n_e \times n_e}$, $H \in \mathbb{R}^{n_p \times n_p}$ and, if $n_i > n_e n_p$ then $\mathcal{S} \in \mathbb{R}^{n_e n_p \times n_e \times n_p}$, $F \in \mathbb{R}^{n_i \times n_e n_p}$, otherwise, $\mathcal{S} \in \mathbb{R}^{n_i \times n_e \times n_p}$, $F \in \mathbb{R}^{n_i \times n_i}$. Recalling that the aim of the algorithm is to classify a new image, we separate the information on the pixels and the expressions from the information on the persons, i.e we consider $\mathcal{C} = \mathcal{S} \times_i F \times_e G$. Then for a fixed expression e

$$A^e = C^e H^T \qquad\qquad e = 1, 2, \ldots, n_e, \tag{2.3}$$

where A^e and C^e are the matrices obtained by fixing the second index of $\mathcal{A}$ and $\mathcal{C}$ equal to e. Hence each column p of A^e can be written as a linear combination of the columns of C^e as follows

$$a_p^{(e)} = C^e h_p^T, \tag{2.4}$$

where h_p^T, the p-th column of H^T, are the coordinates of $a_p^{(e)}$ in the basis C^e. Note that the coordinates of person p in different bases C^e are the same. Now let $z \in \mathbb{R}^{n_i}$ be the new image to be classified. The coordinates of z in the expression basis can be found by solving a least squares problem

$$\hat{\alpha}_e = \text{argmin}_{\alpha_e} \|C^e \alpha_e - z\|_2, \qquad e = 1, \ldots, n_e. \tag{2.5}$$

Then, for each $e = 1, \ldots, n_e$ and for each $p = 1, \ldots, n_p$

$$D_{HO}(e, p) = \|\hat{\alpha_e} - h_p^T\|_2.$$

Note that if C^e is full column rank, then the least squares problem (2.5) has a unique solution, otherwise the solution is not unique, but the solution with smallest norm can be obtained by using the SVD of C^e for its computation ([35, section 5.5]).

A preliminary version of the classification algorithm, described in [28], is given in Algorithm 6. Since the solution of n_e least squares problems can be computationally demanding, in [28] a variant of Algorithm 6 is presented. Assume that $n_i \gg n_e n_p$ and

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

Algorithm 6 Face Recognition (preliminary version)[28, p.173]

Require: z test image.
1: **for** $e = 1, 2, \ldots, n_e$ **do**
2: Compute $\hat{\alpha}_e = \min_{\alpha_e} \|C_e\alpha_e - z\|_2$
3: **for** $p = 1, 2, \ldots, n_p$ **do**
4: $D_{HO}(e, p) = \|\hat{\alpha}_e - h_p^T\|_2$
5: **end for**
6: **end for**
7: $d_1 = \min(D_{HO})$;
8: $[d_2, \hat{p}] = \min(d_1)$.
9: **Output**: Classify z as person $\hat{p}$

consider $F \in \mathbb{R}^{n_i \times n_e n_p}$ obtained by means of a thin HOSVD. Then, enlarge F so that it becomes square and orthogonal, that is

$$\hat{F} = (F \;\; F^{\perp}) \tag{2.6}$$

If we set $B_e = \mathcal{S} \times_i F \times_e G$, we have

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

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

$$\min_{\alpha_e} \|B_e\alpha_e - \hat{z}\|_2 \qquad e = 1, \ldots, n_e. \tag{2.7}$$

Since the matrix B_e is smaller than C_e, it is cheaper to solve (2.7) instead of (2.5). To further reduce work, the thin QR-Decomposition of B_e can be computed.

This procedure is summarized in Algorithm 7. 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 (2.6) we consider $\hat{F} = F$ and both Algorithm 6 and Algorithm 7 can be used.

There is a tight connection between the LS-based method and the HOSVD classification, given that for each expression e the matrix A^e is the same.

Proposition 2.2.1. *Let x_e be the solution to (2.1) in the LS-based method, and let $\hat{\alpha}_e$ be the solution to (2.5) in the HOSVD method. Then $\hat{\alpha}_e = H^T x_e$. Letting $I_{:,p}$ be the p-th column of the canonical basis, it then holds*

$$D_{HO}(e, p) = \|\hat{\alpha}_e - h_p^T\|_2 = \|x_e - I_{:,p}\|_2.$$

Algorithm 7 Face Recognition [28, p.174]

Require: z test image.
1: Compute $\hat{z} = F^T z$:
2: **for** $e = 1, \ldots, n_e$ **do**
3: Compute the thin QR-Decomposition of B_e
4: Denote as $\hat{\alpha}_e$ the solution of $R_e \alpha_e = Q_e^T \hat{z}$
5: **for** $p = 1, \ldots, n_p$ **do**
6: $D_{HO}(e, p) = \|\hat{\alpha}_e - h_p\|_2$.
7: **end for**
8: **end for**
9: $d_1 = \min(D_{HO})$;
10: $[d_2, \hat{p}] = min(d_1)$
11: **Output**: Classify z as person $\hat{p}$

Proof. The least squares solution $\hat{\alpha}_e$ is given by $\hat{\alpha}_e = ((C_e)^T C_e)^{-1}(C_e)^T z$, while the least squares solution x_e is given by $x_e = ((A^e)^T A^e)^{-1}(A^e)^T z$. Substituting $A^e = C^e H^T$ we obtain

$$\begin{aligned} x_e &= ((A^e)^T A^e)^{-1}(A^e)^T z = (H(C^e)^T C^e H^T)^{-1} H(C^e)^T z \\ &= H^{-T}((C^e)^T C^e)^{-1}(C^e)^T z = H^{-T}\hat{\alpha}_e, \end{aligned}$$

which proves the first assertion. For the second assertion, since $\hat{\alpha}_e = H^T x_e$, we have $\|\hat{\alpha_e} - h_p^T\|_2 = \|H^T x_e - H^T I_{:,p}\|_2 = \|H^T(x_e - I_{:,p})\|_2$. From the orthogonality of H the result follows. □

Assume that z and A^e have both been scaled to have unit norm columns, so that the entries of x_e are all not greater than one in absolute value. It follows that for a fixed e, finding the minimum of $D_{HO}(e, p)$ corresponds to finding the largest entry of x_e, the $\hat{p}$-th entry, which is closest to one. Hence, the difference between the two methods lays in the way the "best" expression is chosen. In the basic least squares method the expression corresponding to the smallest residual norm is selected, whereas in HOSVD all is based on the quantity $\|x_e - I_{:,p}\|$. This implies that the HOSVD and LS methods compute the same quantities, but the stopping criterion changes, which is thus responsible of possible discrepancies in the performance of the two approaches. Table 2.1 displays the differences and similarities between the least-squares and the HOSVD formulations for the classification of a single image: the test set is composed by the person p in the last expression, while all the other n_p persons in the remaining expressions are used as training set. We report the classification result of each method by displaying $\hat{p}$ for each database. The method LS2 corresponds to Algorithm 5, in which lines 5 and 6 are replaced with $[\hat{e}, \hat{p}] = \text{argmin} D(e, p) = \text{argmin}\|x_e - I_{:,p}\|$. According to Proposition 2.2.1, the same classification results are obtained using either HOSVD or LS2.

Another classification algorithm based on HOSVD is presented in [67], where the training set is decomposed as in (2.2). The tensor $\mathcal{L} = \mathcal{S}(1 : r, 1 : s, :) \times_3 H$ is then

Database	p	LS	LS2	HOSVD
Orl	14	18	14	14
COIL-20	1	1	1	1
Faces95	5	50	57	57
Faces96	59	77	59	59
Ext'd Yale shrunk	9	10	6	6

Table 2.1: Value of $\hat{p}$ for least squares based algorithms and the algorithm based on HOSVD with respect to different subjects p of five different databases.

Algorithm 8 HOSVD2[67, p.999]

Require: z test image, $\mathcal{A}$ training sets

1: Compute the HOSVD of $\mathcal{A} = \mathcal{S} \times_1 F \times_2 G \times_3 H$.
2: **for** $p = 1, \ldots, n_p$ **do**
3: Compute $\mathcal{L} = \mathcal{S}(1:r, 1:s, :) \times_3 H$ and store the first k singular vectors of $L^p = \mathcal{L}(:, :, p)$ in a matrix B^p
4: Compute $d(p) = \|z_p - B^p(B^p)^T z_p\|_2$
5: **end for**
6: **Output**: Classify z as person $\hat{p} = \text{argmin}_p(d)$

computed[3], and for every person p the first k singular vectors of $L^p = \mathcal{L}_{:,:,p}$ are stored in a matrix B^p to take the most relevant characteristic of each class. Let $z \in \mathbb{R}^{n_i}$ be the image of an unknown person in an unknown expression to be classified. The following least squares problem measures how far z is from being a linear combination of the columns of B^p for each $p = 1, \ldots, n_p$:

$$d(p) := \min_x \|z_p - B^p x\|_2, \quad \text{where} \quad z_p = F(1:r, :)^T z. \tag{2.8}$$

Using the orthogonality of the columns of B^p, the solution of (2.8) is given by $x_p = (B^p)^T z_p$ and $d(p) = \|z_p - B^p(B^p)^T z_p\|_2$. As for the previous algorithms z is classified as person $\hat{p} = \text{argmin}_p(d)$. The whole procedure is summarized in Algorithm 8. In Section 2.4 we will refer to this algorithm as HOSVD2. We conclude this brief description by noticing that HOSVD2 requires the computation of an HOSVD of the training tensor and n_p SVDs to determine the matrices B^p, $p = 1, \ldots, n_p$. This can be computationally expensive when n_p is large.

2.3 Tensor-Train classification algorithm

In this section, we derive a new Tensor-Train classification algorithm following the scheme of Algorithm 6. We first write the data tensor for each expression e, by means of a TT-

[3]The parameters r, s can be set arbitrarily and depend on the desired data compression.

based basis for $\mathbf{E}^e$. Then we define the associated distance to be minimized.

Using (1.23), $\mathcal{A}_{:,e,:}$ can be written as

$$\mathcal{A}_{:,e,:} = G_2^e \times_i G_1 \times_p G_3^T \qquad \forall \ e = 1, \dots, n_e, \tag{2.9}$$

where $G_2^e = (\mathcal{G}_2)_{:,e,:}$ is a matrix. Thus, the image of a person p in the expression e is

$$\mathcal{A}_{:,e,p} = G_1 G_2^e g_3^p, \qquad \text{where} \quad g_3^p = (G_3)_{:,p}\,. \tag{2.10}$$

As in the HOSVD setting we want to separate the information on the person from the information on the pixels and on the expressions. Thus, we set $\mathcal{C} = \mathcal{G}_2 \times_i G_1$ so that (2.10) becomes

$$\mathcal{A}_{:,e,p} = C^e g_3^p, \qquad \text{with} \quad C^e = \mathcal{C}_{:,e,:} = G_1 G_2^e. \tag{2.11}$$

This can be interpreted as follows. The columns of C^e are a basis for $\mathbf{E}^e$ while g_3^p, the p-th column of G_3, holds the coordinates of person p in this basis. Note that the same g_3^p holds the coordinates of all the images of person p in the different expression bases [28, p.173].

Given the image z of an unknown person in an unknown expression, we want to find the coordinates α_e of z in all the n_e bases $\{C^e\}_{e=1,\dots,n_e}$ and then compare each α_e for $e = 1, \dots, n_e$ with the coordinates of all n_p persons in the same basis, which are represented by the columns of G_3. More precisely, for each $e = 1, \dots, n_e$ compute

$$\min_{\alpha_e} \|C^e \alpha_e - z\|_2, \tag{2.12}$$

and then, for each $e = 1, \dots, n_e$ and for each $p = 1, \dots, n_p$ define the distance between z and the person p in expression e as

$$D_{TT}(e, p) := \|\hat{\alpha_e} - g_3^p\|_2, \tag{2.13}$$

where $\hat{\alpha_e}$ is the solution of (2.12). Hence, the distance between z and person p is given by

$$dist(z, \mathcal{A}_{:,:,p}) = \min_e D_{TT}(e, p).$$

The computation in (2.12) can be performed by means of the reduced QR decomposition of G_2^e, that is $G_2^e = Q^e R^e$. The coordinates of z in $\mathbf{E}^e$ are thus given by $\hat{\alpha_e} = {R^{(e)}}^{-1} {Q^{(e)}}^T G_1^T z$ and so (2.13) becomes

$$D_{TT}(e, p) = \|{R^{(e)}}^{-1} {Q^{(e)}}^T G_1^T z - g_3^p\|_2 \qquad p = 1, \dots, n_p,$$

where $g_3^p = (G_3)_{:,p}$. The overall procedure is summarized in Algorithm 9.

Remark 2.3.1. *Let $A^e = \mathcal{A}(:, e, :)$ in (2.10). Then a result analogous to that in Proposition 2.2.1 can be derived, that is the distance D_{TT} satisfies*

$$D_{TT}(e, p) = \|x_e - I_{:,p}\|$$

where $x_e = \operatorname{argmin}_x \|A^e x - z\|$.

Algorithm 9 TT3D

Input: z test image G_1, $\mathcal{G}_2$, G_3.
Compute $\hat{z} = G_1^T z$:
for $e = 1, \ldots, n_e$ **do**
  Solve $G_2^e \alpha_e = \hat{z}$ for α_e
  for $p = 1, \ldots, n_p$ **do**
    $D_{TT}(e,p) = \|\alpha_e - g_3^p\|_2$ where $g_3^p = G_3(:,p)$.
  end for
end for
$(\hat{e}, \hat{p}) = \underset{e,p}{\text{argmin}}(D_{TT})$
Output: Classify z as person $\hat{p}$

2.3.1 Higher-order classification algorithm

Several authors in the literature have observed that image classification can be improved by considering images as matrices $I \in \mathbb{R}^{n_1 \times n_2}$ instead of vectors, thus adding a forth dimension to the problem; see, e.g., [39]. Here we represent the same database as a fourth-order tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_e \times n_p}$ written in TT-form as

$$\mathcal{A} = G_1 \times_2^1 \mathcal{G}_2 \times_3^1 \mathcal{G}_3 \times_3^1 G_4,$$

where $G_1 \in \mathbb{R}^{n_1 \times n_1}$, $\mathcal{G}_2 \in \mathbb{R}^{n_1 \times n_2 \times n_e n_p}$, $\mathcal{G}_3 \in \mathbb{R}^{n_e n_p \times n_e \times n_p}$, and $G_4 \in \mathbb{R}^{n_p \times n_p}$ (see Section 1.2.3). Thus, the image of a person p in expression e in the database is given by

$$\mathcal{A}_{:,:,e,p} = \mathcal{G}_2 \times_1 G_1 \times_3 (G_3^{(e)})^T \times_3 g_4^{(p)}, \tag{2.14}$$

where $G_3^{(e)} = (\mathcal{G}_3)_{:,e,:} \in \mathbb{R}^{n_e n_p \times n_p}$ and $g_4^{(p)} = (G_4)_{:,p} \in \mathbb{R}^{n_p}$. Let $\mathcal{C}^{(e)} = \mathcal{G}_2 \times_1 G_1 \times_3 (G_3^{(e)})^T$. Then (2.14) becomes

$$\mathcal{A}_{:,:,e,p} = \mathcal{C}^{(e)} \times_3 g_4^{(p)}.$$

The classification strategy is then analogous to that for the three-dimensional case. In particular, given an image $z \in \mathbb{R}^{n_1 n_2}$ of an unknown person in an unknown expression, we define the distance of z from person p as

$$D_{TT4}(e,p) = \|\hat{\alpha_e} - g_4^{(p)}\|_2,$$

where

$$\hat{\alpha_e} = \text{argmin}_{\alpha_e} \|(\mathcal{C}_{(3)}^{(e)})^T \alpha_e - z\|_2.$$

A version of the classification algorithm is given in Algorithm 10. We recall that $\mathcal{C}_{(3)}^{(e)}$ denotes the mode-3 unfolding of $\mathcal{C}^{(e)}$ defined in Section 1.1.

The TT format naturally extends to the higher-dimensional setting. For instance, suppose that $\mathcal{A}$ is an Nth-order tensor representing our database, further assume that

Algorithm 10 TT 4D

Input: z test image, G_1, $\mathcal{G}_2$, $\mathcal{G}_3, G_4$.
Compute $\mathcal{G}_{12} = \mathcal{G}_2 \times_1 G_1$:
for $e = 1, \ldots, n_e$ **do**
  Compute $\mathcal{C}^{(e)} = \mathcal{G}_{12} \times_3 (G_3^{(e)})^T$, where $G_3^{(e)} = \mathcal{G}_3(:, e, :)$
  Solve $\min_{\alpha_e} \|(\mathcal{C}_{(3)}^{(e)})^T \alpha_e - z\|_2$ for α_e
  for $p = 1, \ldots, n_p$ **do**
    $D_{TT4}(e, p) = \|\alpha_e - g_4^p\|_2$ where $g_4^p = G_4(:, p)$.
  end for
end for
$(\hat{e}, \hat{p}) = \underset{e,p}{\operatorname{argmin}}(D_{TT4})$
Output: Classify z as person $\hat{p}$

the first two modes are the pixel modes (as in TT4D) and the last one is the person mode. All the other TT-cores $(\mathcal{G}_i)_{i=2,\ldots,N-1}$ represent the variation of the images of the database in terms of facial expression, view angles, illumination,etc. According to (3.3) $\mathcal{A}$ can be written in the following way:

$$\mathcal{A} = G_1 \times_2^1 \mathcal{G}_2 \times_3^1 \mathcal{G}_3 \times_3^1 \cdots \times_3^1 \mathcal{G}_{N-1} \times_3^1 G_N.$$

Thus, person p in a specific combination of all the other $(N-3)$ features can be expressed as

$$\mathcal{A}_{:,:,i_3,\ldots,i_{N-1},p} = G_1 \times_2^1 \mathcal{G}_2 \times_3^1 \mathcal{G}_3^{(i_3)} \times_3^1 \cdots \times_3^1 \mathcal{G}_{N-1}^{(i_{N-1})} \times_3^1 (G_N)_{:,p}. \tag{2.15}$$

Let $C^{(i_3,\ldots,i_{N-1})} = G_1 \times_2^1 \mathcal{G}_2 \times_3^1 \mathcal{G}_3^{(i_3)} \times_3^1 \cdots \times_3^1 \mathcal{G}_{N-1}^{(i_{N-1})}$. Then, (2.15) becomes

$$\mathcal{A}_{:,:,i_3,\ldots,i_{N-1},p} = C^{(i_3,\ldots,i_{N-1})} \times_3 g_N^{(p)}, \tag{2.16}$$

where $g_N^{(p)} = G_N(:, p)$.

Given an image vector $z \in \mathbb{R}^{n_1 n_2}$ of an unknown person in an unknown combination of the $(N-3)$ features, we define the distance between z and the person p as

$$D_{TTN}(i_3, \ldots, i_{N-1}, p) = \|\hat{\alpha}_{i_3,\ldots,i_{N-1}} - g_N^{(p)}\|,$$

where

$$\hat{\alpha}_{i_3,\ldots,i_{N-1}} = \underset{\alpha_{i_3,\ldots,i_{N-1}}}{\operatorname{argmin}} \|(C_{(3)}^{(i_3,\ldots,i_{N-1})})^T \alpha_{i_3,\ldots,i_{N-1}} - z\|.$$

2.4 Numerical experiments

In this section we compare all the tensor classification algorithms and we explore the advantages in using a Tensor-Train formulation in terms of memory requirements, CPU

time and classification performance on several image databases. Moreover, in Section 2.4.3 a classification example in higher dimensional setting for the TT-based algorithm is reported. Before showing classification results, some implementation issues are discussed.

2.4.1 Implementation details

In the following numerical experiments we compare all the aforementioned tensor classification algorithms implemented in Matlab. All the numerical experiments were conducted on a 4 × Intel(R) Core(TM) i7-4500U CPU @ 1.80 GHz and 8GB RAM using Matlab R2019b. For each run the training set and the test set are defined as follows. The $s\%$ of the n_e expressions, denoted as $\tilde{n}_e$, is used for each person as training set and the remaining ones as test set. The expressions used as training set are chosen randomly. The training set is stored in a tensor $\mathcal{A} \in \mathbb{R}^{n_i \times \tilde{n}_e \times n_p}$ for LS, HOSVD, HOSVD2 and TT3D and in a $4th$-order tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times \tilde{n}_e \times n_p}$ for TT4D. As performance measure we consider either the median success rate or the median values of *precision*, *accuracy* and *recall* among 20 consecutive runs.

2.4.2 Classification performance

In this section we first report on our numerical experiments where we use the success rate as performance measure. We report results for $s = 80\%, 50\%$; except for HOSVD2, the tensor methods were not overly sensitive to the choice of this parameter. In Table 2.2 we report the success rate of all considered algorithms, namely matrix SVD, tensor SVD (HOSVD, HOSVD2), three- and four-dimensional TT (TT 3D and TT 4D) and simple least squares (LS) method. For all datasets except the large MNIST ones, all percentages correspond to the average of 20 consecutive runs.

Table 2.2 shows that all the tensor methods work well when $n_i > n_e$, i.e. with the Orl, COIL-20, Faces95 and Faces96 datasets.

However, in many applications it happens that $n_i < n_e$ because of the huge number of features available. In such situations HOSVD2 cannot be used if r and s are set equal to n_i and n_e respectively (i.e. no truncation is performed on $\mathcal{S}$) or as long as B^p is a tall matrix. It is also interesting to observe that LS works quite well for the datasets where the third variable does not provide a real classification feature, that is for the two MNIST sets.

Memory requirements of the tensor methods can be quite different. Given the tensor $\mathcal{A} \in \mathbb{R}^{n_i \times n_e \times n_p}$, the following table summarizes the maximum storage requirements:

Method	Memory allocations
HOSVD	$n_i n_e n_p + n_i^2 + n_e^2 + n_p^2$
TT 3D	$n_i^2 + n_i n_e n_p + n_p^2$
LS	$n_i n_e n_p$
HOSVD2	$n_i n_e n_p + n_p^2 + n_e^2 n_p$

$s\%$	Database	HOSVD	TT 3D	TT 4D	LS	HOSVD2
80	Orl	94.44%	96.65%	96.69%	95.81%	**97.63**%
	COIL-20	97.50%	99.35%	99.35%	99.17%	**99.95**%
	Faces95	86.56%	87.15%	**91.03**%	86.44%	88.87%
	Faces96	97.68%	**100**%	98.50%	97.23%	99.36%
	Ext'd Yale shrunk	99.08%	99.24%	98.72%	**99.78**%	69.04%
	MNIST	92.27%	91.05%	91.03%	**92.69**%	87.95%
	Fashion MNIST	78.63%	79.20%	77.74%	**84.76**%	45.91%
	MIT-CBCL	100%	**100**%	100%	100%	39.77%
50	Orl	91.38%	92.96%	92.75%	89.93%	**95.20**%
	COIL-20	96.10%	**98.90**%	96.71%	98.04%	99.24%
	Faces95	**96.25**%	83.90%	87.50%	80.24%	88.87%
	Faces96	97.34%	97.94%	**99.33**%	96.87%	98.04%
	Ext'd Yale shrunk	98.70%	98.74%	98.99%	99.64%	**99.91**%
	MNIST	**90.87**%	89.12%	89.12%	89.83%	37.68%
	Fashion MNIST	74.58%	75.22%	74.57%	**81.31**%	33.16%
	MIT-CBCL	100%	**100**%	100%	100%	99.98%

Table 2.2: Success rate (as percentage) of all classification algorithms.

In particular, HOSVD requires to store the core tensor $\mathcal{S} \in \mathbb{R}^{n_i \times n_e \times n_p}$ and three orthonormal matrices $U^{(1)} \in \mathbb{R}^{n_i \times n_i}$, $U^{(2)} \in \mathbb{R}^{n_e \times n_e}$ and $U^{(3)} \in \mathbb{R}^{n_p \times n_p}$. HOSVD2 requires to store in memory the core tensor $\mathcal{S} \in \mathbb{R}^{n_i \times n_e \times n_p}$, the factor matrix $U^{(3)} \in \mathbb{R}^{n_p \times n_p}$ and the first $k \leq n_e$ singular vectors of each L^p. On the other hand, TT 3D needs to store $G_1 \in \mathbb{R}^{n_i \times n_i}$, $\mathcal{G}_2 \in \mathbb{R}^{n_i \times n_e \times n_p}$ and $G_3 \in \mathbb{R}^{n_p \times n_p}$, thus leading to lower memory requirements with respect to the HOSVD based methods when $n_i^2 < n_e^2 n_p$ which is quite common for large datasets. Furthermore, if a truncated TT is considered, the memory requirements for TT3D can be lower than for LS.

Figure 2.1 and Figure 2.2 report on the computational costs (in logarithmic scale) of the tensor-based methods on four of the most time consuming datasets, i.e. for MIT-CBCL, COIL-20, Faces95 and MNIST. In Figure 2.2 the reported CPU time is the time required for the classification of a person p, in an expression e whenever appropriate. The plot shows that the CPU time of TT 3D and HOSVD2 is lower than for all other tensor-based methods. In Figure 2.1 the training time for all the tensor algorithms is reported. As we can observe the Tensor-Train algorithms require less CPU time than the other methods. Hence, this performance measure can help discriminate among methods whenever the classification success rate and memory requirements are comparable. Notice that the LS algorithm is not reported because there is no training time for this algorithm.

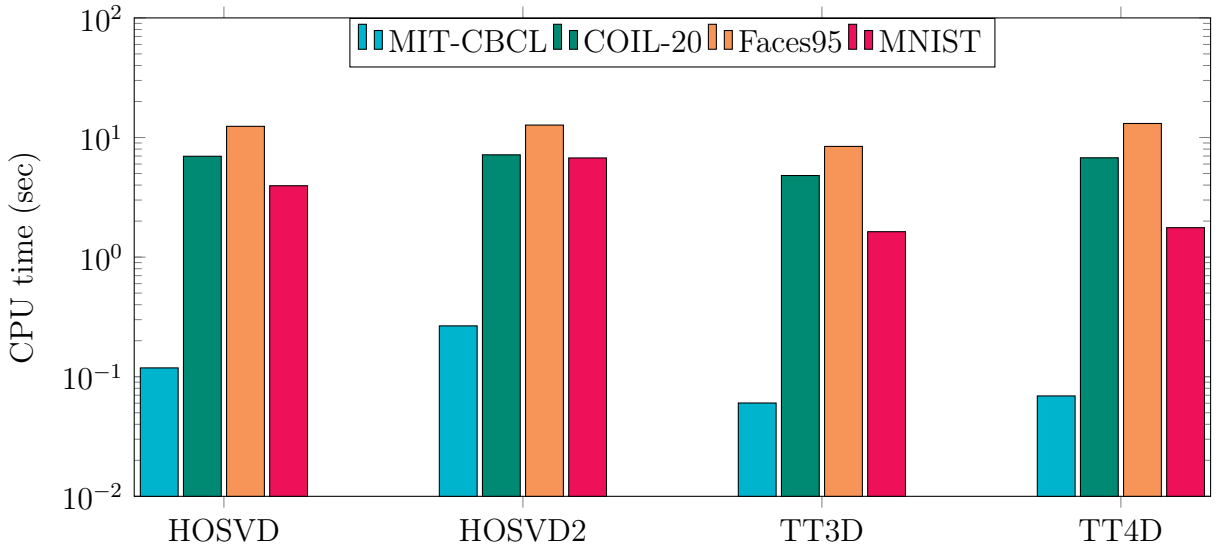


Figure 2.1: CPU training time for MIT-CBCL, COIL-20, Faces95 and MNIST for all the tensor methods.

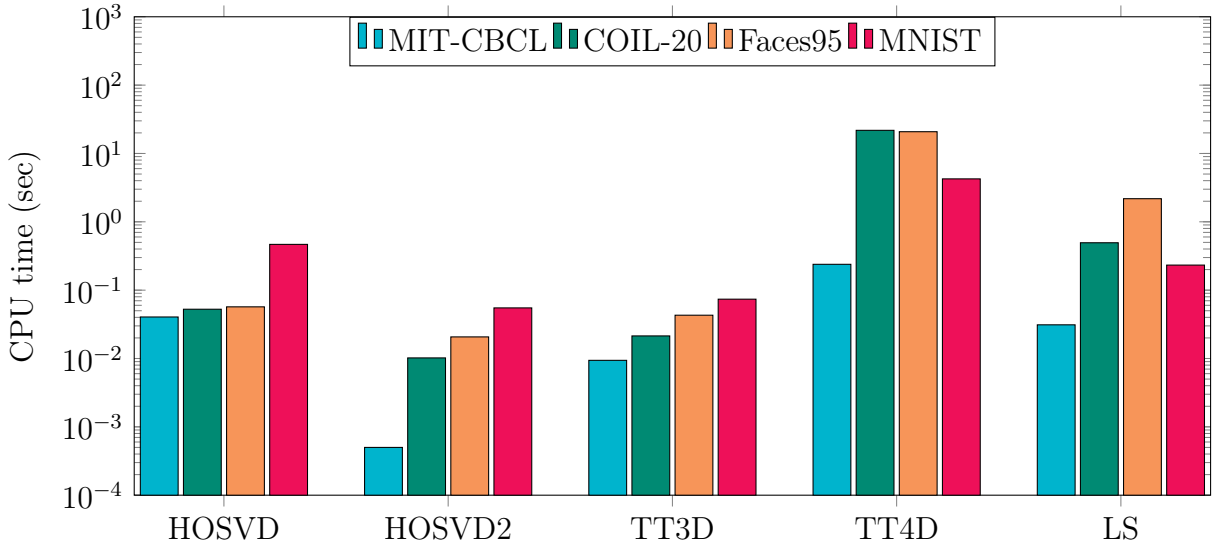


Figure 2.2: CPU time to classify a test image for MIT-CBCL, COIL-20, Faces95 and MNIST for all the tensor methods.

2.4.3 A classification test in higher dimensional setting

In the previous experiments we have considered only two features per person, however in realistic applications a higher number of features may be available. In such situations the Tensor-Train format enables one to still store only third-order tensors, whereas the HOSVD requires to allocate memory for a core tensor of the same order as $\mathcal{A}$.

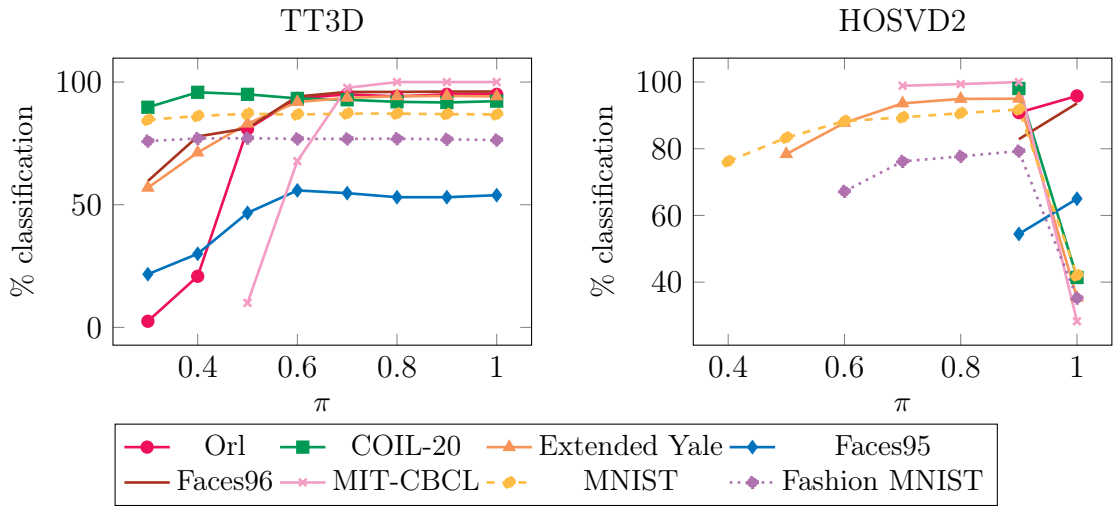


Figure 2.3: Percentage of success of the truncated TT 3D (left) and HOSVD2 (right), for different values of the truncation parameter π, tested on different databases.

In this section we analyze the classification performance of the Tensor-Train algorithm for a fifth-order tensor. To this end, we consider the Weizmann face image database (see Appendix A). The training set is stored as the tensor $\mathcal{A} \in \mathbb{R}^{n_i \times n_v \times n_{ill} \times n_e \times n_p}$. In a first classification test the test set is composed by n_p persons in n_e facial expressions, n_{ill} illuminations and 2 view angles. This means that 80% of the database is used for training and the remaining 20% is used as test set. In this setting 87.05% of the test images are correctly classified. Notice that TensorFaces with this dataset only achieves 80% success (for further details see [79]). In a second classification test the split between training and test sets is set to 90%/10% (i.e., just one view angle is used as test set). In this setting 91.07% of the examples are correctly classified.

2.4.4 Numerical experiments with truncated methods

In this section we report on our experiments with a truncated version of the 3D classification algorithms. We compare the truncated TT-SVD described in Section 1.2.3 with the classification algorithm based on truncated HOSVD2. Figure 2.3 shows the percentage of success of the truncated procedure on five databases, when different values of the truncation threshold π are used (see Section 1.2.3). The displayed curves only report results for π such that $\ell \geq 1$. In HOSVD2 the truncation is applied to the columns of B^p, $p = 1, \ldots, n_p$ according to the truncation parameter π. Furthermore, following [67], $\mathcal{S}$ in (2.2) is replaced with $\hat{\mathcal{S}} = \mathcal{S}(1:48, 1:64, :)$. Several comments are in order. We observe that for some databases such as Orl, methods can behave quite differently for the same value of π. The decrease in the HOSVD2 performance for high values of π is related to the resulting dimensions of the singular vector matrix B^p: to express 90% of the variance (i.e. $\pi \geq 0.9$) of the MIT-CBCL, COIL-20, MNIST and Fashion

MNIST datasets we have to take all columns of B^p, making it square. Thus, for this algorithm there is an additional constraint on the truncation parameter, to ensure a good classification performance for several databases.

Figure 2.3 shows that the classification performance is not monotonic with respect to π. Nonetheless, for TT3D any $\pi \geq 0.6$ leads to a good classification performance.

The higher efficiency in terms of computational costs (CPU time), memory requirements together with the good recognition rate, favor the TT3D truncated version compared to the untruncated one. For instance, for $\pi = 0.9$ in the Extended Yale Database only 32% of the singular values are retained, yielding large memory savings.

2.4.5 Performance using statistical classification measures

Using an applied statistics terminology, the Face Recognition problem can be thought of as a multiclass classification problem, where the classes are the different persons of a specific database. For a database of n_p persons in n_e different expressions, we consider the splitting in a training set (n_p persons in $0.75n_e$ expressions) and a test set (n_p persons in $0.25n_e$ expressions). After the classification of all images in the test set has been completed, for each person p the following quantities can be computed, giving rise to the so-called "confusion matrix":

- *true positive*(tp_p): number of subjects correctly classified as person p;
- *true negative* (tn_p): number of correctly recognized subjects that are not person p;
- *false positive* (fp_p): number of subjects incorrectly classified as person p;
- *false negative* (fn_p): number of subjects not recognized as person p.

Using these four quantities, the following classifier parameters can be computed:

$$Accuracy = \frac{1}{n_p}\sum_{p=1}^{n_p}\frac{tp_p + tn_p}{tp_p + tn_p + fp_p + fn_p}, \qquad Precision = \frac{1}{n_p}\sum_{p=1}^{n_p}\frac{tp_p}{tp_p + fp_p},$$

$$Recall = \frac{1}{n_p}\sum_{p=1}^{n_p}\frac{tp_p}{tp_p + fn_p},$$

These parameters provide a measure of reliability of the employed algorithm: the closer the parameter is to 100% the more robust the corresponding classification strategy is. Depending on the realistic application for which we want to use the classification algorithm, one can be interested in maximizing one of these three quantities. For example, in situations such as border controls or medical tests, one may be interested in maximizing the recall, since this means minimizing the number of images of person p that are not recognized as person p.

In Table 2.3 *precision*, *accuracy* and *recall* are displayed, using macro-averaging (for further details see [71]). The results show an overall different ranking of the methods when these measures are considered, with respect to the percentages in Table 2.2. HOSVD appears to be the most reliable strategy for all three parameters for the MNIST database, whereas for all other datasets tensor-train based algorithms have better performance. It is also interesting that, although the three parameters measure truly different things, the relative ranking of all algorithms does not change much going from, say, Accuracy to Recall. HOSVD2 was not included in the tests in Table 2.3, because its classification performance strongly depends on the truncation parameter.

measure	method	Orl	COIL -20	Faces95	Faces96	Ext'd Yale	MNIST	Fashion MNIST
Accuracy	HOSVD	99.71%	**99.77%**	99.66%	99.97%	99.93%	**98.18%**	95.26%
	TT 3D	**99.78%**	**99.77%**	**99.71%**	**99.98%**	**99.94%**	97.83%	**95.39%**
	TT 4D	99.77%	**99.77%**	99.60%	**99.98%**	99.94%	97.67%	**95.39%**
Precision	HOSVD	76.62%	**98.03%**	91.10%	98.60%	99.06%	**91.18%**	78.30%
	TT 3D	**96.73%**	**98.03%**	**92.53%**	99.07%	99.17%	89.89%	**78.53%**
	TT 4D	96.58%	**98.03%**	90.01%	**99.20%**	**99.20%**	89.00%	**78.53%**
Recall	HOSVD	94.29%	97.67%	87.67%	98.29%	99.01%	**90.90%**	76.31%
	TT 3D	**95.62%**	**97.69%**	**89.60%**	98.85%	99.12%	89.10%	**76.93%**
	TT 4D	95.50%	**97.69%**	85.62%	**99.01%**	**99.16%**	88.32%	**76.93%**

Table 2.3: Accuracy, precision and recall for the HOSVD, TT3D and TT4D classification algorithms.

2.4.6 Closing considerations

The proposed classification algorithm based on the Tensor-Train decomposition seems to exhibit good performance in terms of success rate, CPU time and memory requirements. Indeed, the use of the TT form allows one to easily treat truncation, which reduces both CPU time and memory requirements, without sacrificing the recognition success rate. Moreover, the TT-based algorithm is preferable to those based on HOSVD since it does not suffer from the curse of dimensionality; in particular, it naturally extends to more than three dimensions, thus allowing for the inclusion of additional features as extra dimensions, as we did for the Weizmann database. Indeed, for face recognition other features could be considered, such as different age or backdrop image sets. Our computational experiments on nine different datasets seem to show that using the Tensor-Train form allows one to achieve good classification success for comparable memory requirements (in the full case) and smaller CPU time with respect to the now classical tensor based HOSVD. Furthermore for the training the Tensor-Train algorithms require just the computation of the TT-cores while HOSVD2 requires the computation of the HOSVD of the training tensor and n_p SVD to determine the matrices B^p, $p = 1, \ldots, n_p$. This can be computationally expensive when n_p is large.

Chapter 3

Proximal gradient methods

In this chapter we review the class of proximal gradient methods for solving composite optimization problems, that is problems where the objective is the sum of a smooth function and non-smooth (possibly nonconvex) one. We first present the proximal gradient method and the block proximal gradient method to address composite optimization problems in one or p blocks of variables, respectively, focusing on the case when the non-smooth term is convex. Then we discuss the nonconvex case by introducing the Proximal Alternating Linearized Minimization (PALM) algorithm.

3.1 The proximal gradient method

In this section we consider the following composite optimization problem

$$\min_{x\in\mathbb{R}^n} \Psi(x) \quad \text{with} \quad \Psi(x) = H(x) + f(x), \tag{3.1}$$

where H and f satisfy the following assumption.

Assumption 3.1.1 ([5, Assumption 10.1])**.**

1. *$f : \mathbb{R}^n \to (-\infty, +\infty]$ is proper, closed (see [5, Definition 2.2]) and convex.*
2. *$H : \mathbb{R}^n \to (-\infty, +\infty]$ is proper and closed,* $\mathrm{dom}(H)$ *is convex,* $\mathrm{dom}(f) \subseteq \mathrm{int}(\mathrm{dom}(H))$ *and H is L_H-smooth over* $\mathrm{int}(\mathrm{dom}(H))$*.*
3. *The optimal set of problem (3.1) is non-empty and denoted by X^*. The optimal function value is denoted by Ψ_{opt}.*

The numerical solution of the optimization problem (3.1) can be carried out by using *proximal methods*, that is iterative algorithms based on the following kth iteration: Given the current iterate x_k and the current stepsize $\alpha_k > 0$, a new iterate x_{k+1} is given by

$$x^{k+1} = \mathrm{prox}^f_{1/\alpha_k}(x^k - \alpha_k \nabla H(x^k)),$$

where $\mathrm{prox}^f_{1/\alpha_k}(\cdot)$ denotes the proximal map defined as follows.

Definition 3.1.1 ([9, p. 465]). *Given a proper and lower semicontinuous function $f : \mathbb{R}^m \to (-\infty, +\infty]$, $x \in \mathbb{R}^m$ and a scalar $\alpha > 0$, the proximal map associated to f is defined as*

$$\text{prox}_\alpha^f(x) = \underset{w \in \mathbb{R}^m}{\text{argmin}} \left\{ f(w) + \frac{\alpha}{2}\|w - x\|_2^2 \right\}. \tag{3.2}$$

Remark 3.1.1 ([9]). *The proximal map of an indicator function δ_Ω over a non-empty and closed set $\Omega \subset \mathbb{R}^m$ is the multi-valued projection $P_\Omega : \mathbb{R}^m \rightrightarrows \Omega$ defined, for all $x \in \mathbb{R}^m$, by*

$$P_\Omega(x) = \underset{y \in \Omega}{\text{argmin}} \|x - y\|_2. \tag{3.3}$$

The symbol $\rightrightarrows$ is used to denote that, for a given $x \in \mathbb{R}^m$, $P_\Omega(x)$ is not uniquely defined when Ω is nonconvex. For example, if we consider Ω as the set of vectors with at most $0 < \tau < m$ nonzero elements and $x = [1, \dots, 1]^T$, it is easy to observe that $P_\Omega(x)$ is not uniquely defined. On the other hand if Ω is a convex set the projection map is single-valued, that is $P_\Omega(x)$ is uniquely defined for any $x \in \mathbb{R}^m$.

When $f = \delta_\Omega$, with Ω non-empty, closed and convex set, then (3.1) is equivalent to the following convex constrained smooth optimization problem

$$\min_{x \in \Omega} H(x),$$

and, using Remark 3.1.1, the associated proximal iteration is

$$x^{k+1} = P_\Omega(x^k - \alpha_k \nabla H(x^k)), \tag{3.4}$$

which is indeed the iteration of the *projected gradient descent* method.

Indeed, the proximal gradient method performs a gradient step in the smooth part of the function $\Psi(x)$, i.e. $H(x)$, followed by a proximal mapping that takes into account the non-smooth part of $\Psi(x)$, i.e. $f(x)$. The whole procedure is summarized in Algorithm 11.

We now give some preliminary results that anticipate the convergence theorem for the proximal gradient method. We start with the following theorem that will be used to prove the *sufficient decrease* lemma.

Theorem 3.1.1 (Second prox theorem, [5, Theorem 6.39]). *Let $f : \mathbb{R}^n \to (-\infty, +\infty]$ be a proper closed and convex function. Then for any x, $y \in \mathbb{R}^n$, the following conditions are equivalent.*

(i) $y = prox_\alpha^f(x)$ *with* $\alpha = 1$,

(ii) $x - y \in \partial f(y)$,

(iii) $\langle x - y, z - y \rangle \leq f(z) - f(y)$ *for any* $z \in \mathbb{R}^n$.

The following lemma reports an important property of L_H smooth functions.

Algorithm 11 The Proximal Gradient method [5, p. 271]

1: **Input**: $x^0 \in \text{int}(\text{dom}(H))$
2: **for** $k = 0, \ldots,$ **do**
3: Choose $\alpha_k > 0$
4: set $x^{k+1} = prox^f_{1/\alpha_k}\left(x^k - \alpha_k \nabla H(x^k)\right)$
5: **end for**
6: **Output**: x^{k+1}

Lemma 3.1.1 (Descent lemma, [9, Lemma 1])**.** *Consider an L_H-smooth function $H : \mathbb{R}^m \to \mathbb{R}$. Then for any x, $y \in \mathbb{R}^m$,*

$$H(y) \leq H(x) + \langle \nabla H(x), y - x \rangle + \frac{L_H}{2}\|x - y\|_2^2.$$

This property of H plays an important role in the convergence analysis of the proximal gradient methods due to the following lemma.

Lemma 3.1.2 (Sufficient decrease lemma, [5, Lemma 10.4])**.** *Let H and f be real valued functions that satisfy Assumption 3.1.1. Consider $\Psi = H + f$ and set*

$$\sigma_L^{H,f}(x) = prox_L^f\left(x - \frac{1}{L}\nabla H(x)\right). \tag{3.5}$$

Then for any $x \in \text{int}(\text{dom}(H))$ and $L \in \left(\frac{L_H}{2}, +\infty\right)$ the following inequality holds:

$$\Psi(x) - \Psi(\sigma_L^{H,f}(x)) \geq \left(L - \frac{L_H}{2}\right)\left\|x - \sigma_L^{H,f}(x)\right\|_2^2.$$

Proof. Let $x^+ = \sigma_L^{H,f}(x)$, then using Lemma 3.1.1 for H we obtain

$$H(x^+) \leq H(x) + \langle \nabla H(x), x^+ - x \rangle + \frac{L_H}{2}\|x^+ - x\|_2^2. \tag{3.6}$$

Then using the *second prox theorem* (Theorem 3.1.1) we have that

$$\langle \nabla H(x), x^+ - x \rangle \leq -L\|x^+ - x\|_2^2 + f(x) - f(x^+)$$

for any $L \in \left(\frac{L_H}{2}, +\infty\right)$. Thus, using this inequality in (3.6)

$$H(x^+) \leq H(x) - L\|x^+ - x\| + f(x) - f(x^+) + \frac{L_H}{2}\|x - x^+\|$$

and

$$H(x^+) + f(x^+) - H(x) - f(x) \leq \left(\frac{L_H}{2} - L\right)\|x^+ - x\|_2^2.$$

The result follows from the definition of $\Psi(x)$. □

Now we give the definition of the proximal gradient which is a generalization of the classical gradient.

Definition 3.1.2 ([5, Definition 10.5]). *Suppose that H and f satisfy the Assumption 3.1.1 items 1 and 2. Then the gradient mapping is the operator $G_L^{H,f} : \mathrm{int}(\mathrm{dom}(H)) \to \mathbb{R}$ defined by*

$$G_L^{H,f}(x) = L(x - \sigma_L^{H,f}(x)),$$

with $\sigma_L^{H,f}$ defined in (3.5) for any $x \in \mathrm{int}(\mathrm{dom}(H))$.

The following theorem describes the relation between the classical gradient and the proximal gradient.

Theorem 3.1.2 ([5, Theorem 10.7]). *Let H and f satisfy Assumption 3.1.1 items 1 and 2 and let $L > 0$. Then*

- *$G_L^{H,f_0}(x) = \nabla H(x)$ for any $x \in \mathrm{int}(\mathrm{dom}(H))$, where $f_0(x) = 0$;*
- *for $x^* \in \mathrm{int}(\mathrm{dom}(H))$, it holds that $G_L^{H,f}(x^*) = 0$ if and only if x^* is a stationary point of (3.1).*

A crucial ingredient of all gradient descent methods is the selection of the stepsize α_k that yield a sufficient decrease in the objective function. At each step k one can choose a constant stepsize $\alpha_k = \frac{1}{L}$ where $L \in \left(\frac{L_H}{2}, +\infty\right)$ or a stepsize given by a backtracking rule. A backtracking rule is an iterative procedure commonly used to estimate the stepsize starting from an initial guess s. The procedure consists of decreasing s until a sufficient decrease condition in the objective function is satisfied. For the proximal gradient method, using the backtracking rule results in setting $L_k = \rho^{i_k} s$, where $\rho > 1$ and i_k is the smallest non negative integer for which the condition

$$\Psi(x^k) - \Psi(\sigma_{L_k}^{H,f}(x^k)) \geq \frac{\gamma}{L_k} \|G_{L_k}^{H,f}(x^k)\|_2,$$

with $\gamma \in (0, 1)$ is satisfied. Then the stepsize α_k is set equal to the reciprocal of L_k. The backtracking procedure is reported in Algorithm 12. Note that, under the Assumption 3.1.1, the backtracking procedure has a finite termination as highlighted in the following remark.

Remark 3.1.2 ([5, Remark 10.13]). *In the backtracking procedure of Algorithm 12 the parameter L_k is bounded from above as*

$$L_k \leq \max\left\{s, \frac{\rho L_H}{2(1-\gamma)}\right\} \quad \text{with} \quad \rho > 1, \ \ \gamma \in (0, 1)\ .$$

The following convergence analysis for the proximal gradient method holds both for a constant stepsize and for a stepsize given by a backtracking rule. It is based on the sufficient decrease lemma which shows not only that the sequence $\{\Psi(x^k)\}_{k\geq 0}$ is nonincreasing, but also that the decrease of the function values is bounded by the norm of the proximal gradient.

Algorithm 12 Backtracking procedure

1: **Input**: Initial stepsize $s > 0$, $0 < \gamma < 1$, $\rho > 1$, and an integer $i_{\max}$ for the maximum number of backtracks.
2: Set $L_k = s$
3: **for** $i_k = 1, \ldots, i_{\max}$ **do**
4: Compute $x^k = prox^f_{L_k}\left(x^k - \frac{1}{L_k}\nabla H(x^k)\right)$
5: **if** $\Psi(x^k) - \Psi(\sigma^{H,f}_{L_k}(x^k)) \geq \frac{\gamma}{L_k}\|G^{H,f}_{L_k}(x^k)\|_2$, **return**
6: Set $L_k = \rho L_k$
7: **end for**
8: **Output**: L_k

Lemma 3.1.3 (Sufficient decrease of the proximal gradient method [5, Lemma 10.14])**.** *Assume that Assumption 3.1.1 holds and let $\{x^k\}_{k\geq 0}$ be the sequence generated by the proximal gradient method to solve (3.1) either with a constant stepsize $L \in \left(\frac{L_H}{2}, +\infty\right)$ or with a stepsize obtained using the backtracking procedure with parameters $s > 0$, $0 \leq \gamma < 1$ and $\rho > 1$. Then for any $k \geq 0$*

$$\Psi(x^k) - \Psi(x^{k+1}) \geq M\|G^{H,f}_d(x^k)\|_2, \tag{3.7}$$

where

$$M = \begin{cases} \frac{L - \frac{L_H}{2}}{L^2} & \text{constant stepsize,} \\ \frac{\gamma}{\max\left\{s, \frac{\rho L_H}{2(1-\gamma)}\right\}} & \text{backtracking,} \end{cases} \tag{3.8}$$

and

$$d = \begin{cases} L & \text{constant stepsize,} \\ s & \text{backtracking.} \end{cases} \tag{3.9}$$

Now we report the main convergence result that ensures the convergence of the proximal gradient method to a stationary point of problem (3.1).

Theorem 3.1.3 ([5, Theorem 10.15])**.** *Assume that Assumption 3.1.1 holds and let $\{x^k\}_{k\geq 0}$ be the sequence generated by the proximal gradient method to solve (3.1) either with a constant stepsize $L \in \left(\frac{L_H}{2}, +\infty\right)$ or with a stepsize obtained using the backtracking procedure with parameters $s > 0$, $0 \leq \gamma < 1$ and $\rho > 1$. Then*

1. *the sequence $\left\{\Psi(x^k)\right\}_{k\geq 0}$ is nonincreasing. In addition, $\Psi(x^{k+1}) < \Psi(x^k)$ if and only if x^k is not a stationary point of (3.1);*
2. *$\lim_{k\to\infty} G^{H,f}_d(x^k) = 0$ where d is given in (3.9);*
3. *$\min_{n=0,1,\ldots,k} \|G^{H,f}_d(x^n)\|_2 \leq \frac{\sqrt{\Psi(x^0) - \Psi_{opt}}}{\sqrt{M(k+1)}}$, where M is defined in (3.8);*

4. all limits points of the sequence $\{x^k\}_{k\geq 0}$ *are stationary points of problem (3.1).*

Several variations of the proximal gradient method have been considered for problem (3.1). For example, in [58] the authors describe iPiano which is a proximal gradient descent algorithm enriched with an inertial proximal step based on the linear combination of the previous iterates. In the more recent literature, the use of inexact variable metric [11, 12] and Newton-like steps [46] have also been explored.

3.2 The block proximal gradient method

In this section we extend the results described in the previous section to the following optimization problem

$$\min_{x_1\in\mathbb{R}^{n_1},\dots,x_p\in\mathbb{R}^{n_p}} \Psi(x_1,\dots,x_p) \quad \text{with} \quad \Psi(x_1,\dots,x_p) := H(x_1,\dots,x_p) + f(x_1,\dots,x_p), \tag{3.10}$$

where the function $f : \mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_p} \to (-\infty, +\infty]$ has a block separable structure, i.e.

$$f(x_1,\dots,x_p) = \sum_{i=1}^{p} f_i(x_i).$$

In the following discussion we denote by $\mathbb{R}^*$ the product space $\mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_p}$ and by $\nabla_i H$ for $i = 1,\dots,p$ the gradient with respect to the ith block. We also define $\omega_i : \mathbb{R}^{n_i} \to \mathbb{R}^*$ to be the linear transformation

$$\omega_i(d) = \begin{bmatrix} \mathbf{0} \\ d \\ \mathbf{0} \end{bmatrix} \; i\text{th-block}, \quad d \in \mathbb{R}^{n_i}.$$

We now report the assumptions on the model (3.10).

Assumption 3.2.1 ([5, Assumption 11.1])**.**

1. $f_i : \mathbb{R} \to (-\infty, +\infty]$ *is proper, closed and convex for any* $i = 1,\dots,p$.
2. $H : \mathbb{R} \to (-\infty, +\infty]$ *is proper and closed, and* $\mathrm{dom}(H)$ *is convex;* $\mathrm{dom}(f) \subseteq \mathrm{int}(\mathrm{dom}(H))$*, and* H *is differentiable over* $\mathrm{int}(\mathrm{dom}(H))$.
3. H *is* L_H*-smooth over* $\mathrm{int}(\mathrm{dom}(H))$ *with* $L_H > 0$.
4. *There exist* $L_1,\dots,L_p > 0$ *such that for any* $i = 1,\dots,p$ *it holds that*

$$\|\nabla_i H(x) - \nabla_i H(x + \omega_i(d))\|_2 \leq L_i \|d\|_2$$

for all $x \in \mathrm{int}(\mathrm{dom}(H))$ *and* $d \in \mathbb{R}^{n_i}$ *for which* $x + \omega_i(d) \in \mathrm{int}(\mathrm{dom}(H))$.

5. *The set of optimal solutions of problem (3.10) is non-empty and denoted by X^*. The optimal function value is denoted by Ψ_{opt}.*

Definition 3.2.1 ([5, Definition 11.3])**.** *Assume that H and f_i for $i = 1, \ldots, p$ satisfy Assumption 3.2.1 and let $L > 0$. Then the ith partial prox-grad mapping is the operator $\sigma_L^i : \text{int}(\text{dom}(H)) \to \mathbb{R}^{n_i}$ defined by*

$$\sigma_L^i(x) = prox_L^{f_i}\left(x_i - \frac{1}{L}\nabla_i H(x)\right).$$

Definition 3.2.2 ([5, Definition 11.4])**.** *Assume that H and f_i for $i = 1, \ldots, p$ satisfy Assumption 3.2.1 and let $L > 0$. Then the ith partial gradient mapping is the operator $G_L^i : \text{int}(\text{dom}(H)) \to \mathbb{R}^{n_i}$ defined by*

$$G_L^i(x) = L(x_i - \sigma_L^i(x)).$$

Using the block structure of the problem, the stationary conditions for (3.10) reduce to p different conditions on the partial gradients as will be clear from the following theorem.

Theorem 3.2.1 ([5, Theorem 11.6])**.** *Assume that H and f_i for $i = 1, \ldots, p$ satisfy Assumption 3.2.1 items 1 and 2. Then*

(i) $x^* \in \text{dom}(f)$ is a stationary point of problem (3.10) if and only if

$$-\nabla_i H(x*) \in \partial f_i(x_i^*), \quad i = 1, \ldots, p.$$

(ii) For any $M_1, \ldots, M_p > 0$, $x^* \in \text{dom}(f)$ is a stationary point of (3.10) if and only if

$$G_{M_i}^i(x^*) = 0, \quad i = 1, \ldots, p.$$

The following result is a generalization of the *sufficient decrease* lemma (Lemma 3.1.3) for problem (3.10).

Lemma 3.2.1 (block sufficient decrease lemma, [5, Lemma 11.9])**.** *Assume that H and f_i for $i = 1, \ldots, p$ satisfy Assumption 3.2.1 items 1 and 2 and that there exists $L_i > 0$ with $i \in \{1, \ldots, p\}$ for which*

$$\|\nabla_i H(y) - \nabla_i H(y + \omega_i(d))\|_2 \leq L_i \|d\|_2,$$

for any $y \in \text{int}(\text{dom}(H))$ and $d \in \mathbb{R}^{n_i}$ for which $y + \omega_i(d) \in \text{int}(\text{dom}(H))$. Then

$$\Psi(x) - \Psi(x + \omega_i(\sigma_{L_i}^i(x) - x_i)) \geq \frac{1}{2L_i}\|G_{L_i}^i(x)\|_2^2,$$

for all $x \in \text{int}(\text{dom}(H))$.

Problem (3.10) can be solved numerically using the Cyclic Block Proximal Gradient (CBPG) method, which is an alternating method that in each variable block performs a proximal gradient iteration. More precisely, at each iteration the method performs p proximal gradient iterations one for each variable block. For these p "subiterations" the following auxiliary index will be used.

$$\begin{aligned} x^{k,0} &= x^k = (x_1^k, x_2^k \dots, x_p^k), \\ x^{k,1} &= (x_1^{k+1}, x_2^k, \dots, x_p^k), \\ &\vdots \\ x^{k,p} &= x^{k+1} = (x_1^{k+1}, x_2^{k+1}, \dots, x_p^{k+1}). \end{aligned}$$

The whole procedure is summarized in Algorithm 13.

Algorithm 13 CBPG [5, p. 338]

Input: $x^0 = (x_1^0, \dots, x_p^0) \in \text{int}(\text{dom}(H))$
for $k = 0, 1, \dots,$ **do**
  set $x^{k,0} = x^k$
  for $i = 1, \dots, p$ **do**
    Use a fixed value for L_i or compute L_i using a backtracking strategy
    Set $x^{k,i} = x^{k,i-1} + \omega_i(\sigma_{L_i}^i(x^{k,i-1}) - x_i^{k,i-1})$.
  end for
  Set $x^{k+1} = x^{k,p}$.
end for
Output: $x^k + 1$

Note that one can choose either fixed values for L_i, $i = 1, \dots, p$ or use a backtracking rule similar to the one described in Algorithm 12.

As for the proximal gradient, the following result is needed to derive the convergence of the CBPG method.

Lemma 3.2.2 (Sufficient decrease of the CBPG method-version I [5, Lemma 11.11])**.** *Assume that Assumption 3.2.1 holds and consider the sequence $\{x^k\}_{k\geq 0}$ generated by the CBPG method for solving (3.10). Then*

1. *For all $k \geq 0$*

$$\Psi(x^{k,j}) - \Psi(x^{k,j+1}) \geq \frac{L_{j+1}}{2}\|x^{k,j} - x^{k,j+1}\|_2^2, \quad j = 0, \dots, p-1 \tag{3.11}$$

2. *for all $k \geq 0$*

$$\Psi(x^k) - \Psi(x^{k+1}) \geq \frac{L_{\min}}{2}\|x^k - x^{k+1}\|_2^2,$$

with $L_{\min} = \min_i L_i$.

Now we are ready to state the main convergence result for the CBPG method.

Theorem 3.2.2 ([5, Theorem 11.14]). *Assume that Assumption 3.2.1 holds and consider the sequence $\{x^k\}_{k\geq 0}$ generated by the CBPG method for solving (3.10). Denoting as $L_{\min} = \min_i L_i$, $L_{\max} = \max_i L_i$ and $C = \frac{L_{\min}}{2\left(L_H + 2L_{\max} + \sqrt{L_{\min} L_{\max}}\right)^2}$, then*

1. *the sequence $\left\{\Psi(x^k)\right\}_{k\geq 0}$ is nonincreasing. In addition, the equality holds if and only if $x^k = x^k + 1$;*
2. *$G_{L_{\min}}(x^k) \to 0$ as $k \to \infty$;*
3. *$\min_{n=0,\dots,k} \|G_{L_{\min}}(x^n)\|_2 \leq \frac{\sqrt{p(\Psi(x^0) - \Psi_{opt})}}{\sqrt{C(k+1)}}$;*
4. *all limit points of the sequence $\{x^k\}_{k\geq 0}$ are stationary points of (3.10).*

Different variants of the CBPG have been explored in the recent literature. In [10] the CBPG is applied to nonnegative matrix factorization employing a stepsize based on an adaptive alternation of the Barzilai-Borwein rule (see Section 4.1). In [13] the convergence results of the CBPG have been extended also for projection operations based on non Euclidean distances, while an application of this approach to blind deconvolution is presented in [65].

3.3 The Proximal Alternating Linearized Minimization (PALM) algorithm

As discussed in the previous section, the CBPG method is a powerful strategy to minimize a finite sum of functions that satisfy Assumption 3.2.1. However in several problems, such as the Dictionary Learning problem analyzed in this thesis (see Chapters 5 and 6), the convexity assumption on the functions f_i is not satisfied. In such situations, the Proximal Alternating Linearized Minimization (PALM) algorithm [9] provides a viable strategy to solve the composite optimization problem with nonconvex non-smooth terms maintaining convergence guarantees. Indeed PALM gives a general setting for solving non-smooth nonconvex optimization problems of the form

$$\min_{x,y} \Psi(x,y) \quad \text{with} \quad \Psi(x,y) := H(x,y) + f_1(x) + f_2(y), \tag{3.12}$$

where the functions f_1 and f_2 are extended valued (i.e., allowing the inclusion of constraints) and H is a smooth coupling function, only required to have partial Lipschitz continuous gradients $\nabla_x H$ and $\nabla_y H$ with Lipschitz constants $L_1(y)$ and $L_2(x)$. In the following discussion we refer to the constants $L_1'(y)$ and $L_2'(x)$ as *partial smoothness parameters* (see forthcoming Assumption 3.3.1).

For each block of coordinate in (3.12), PALM performs one gradient step on the smooth part, followed by a proximal step on the non-smooth part. The method belongs to the class of alternating minimization schemes, and generalizes to the nonconvex non-smooth case well-known and widely used alternating algorithms such as CBPG method and many others [4, 10, 38, 62]. The PALM algorithm is reported in Algorithm 14.

Algorithm 14 PALM (with constant stepsize) [9, p. 468]

1: **Input**: $(x_0, y_0) \in \mathbb{R}^n \times \mathbb{R}^m$, $\eta_1, \eta_2 > 1$, $\mu_1, \mu_2 > 0$
2: **for** $k = 0, 1, \ldots$ **do**
3: *Update* x: Set $L_1''(y^k) = \max\{\eta_1 L_1'(y^k), \mu_1\}$ and $\bar{\alpha}_{k,1} = 1/L_1''(y^k)$ and compute

$$x^{k+1} = \operatorname{prox}^{f_1}_{1/\bar{\alpha}_{k,1}}\left(x^k - \bar{\alpha}_{k,1}\nabla_x H(x^k, y^k)\right) \tag{3.13}$$

4: *Update* y: Set $L_2''(x^{k+1}) = \max\{\eta_2 L_2'(x^{k+1}), \mu_2\}$ and $\bar{\alpha}_{k,2} = 1/L_2''(x^{k+1})$ and compute

$$y^{k+1} = \operatorname{prox}^{f_2}_{1/\bar{\alpha}_{k,2}}\left(y^k - \bar{\alpha}_{k,2}\nabla_y H(x^{k+1}, y^k)\right) \tag{3.14}$$

5: **end for**

As clear from Algorithm 14, PALM makes explicit use of the partial smoothness parameters $L_1''(y^k)$ and $L_2''(x^k)$. When these parameters are not available, they can be approximated by using a backtracking strategy similar to the one described in Algorithm 12 for the proximal gradient method. Indeed, setting $\Psi_1(x, y) := H(x, y) + f_1(x)$ and $L_{0,1} = 1$, at each iteration $k > 1$, the procedure starts with $L_{k,1} = L_{k-1,1}$ and then $L_{k,1}$ is increased by a constant factor, typically doubled, until the following *sufficient decrease condition* is met

$$\Psi_1(x^{k+1}, y^k) < \Psi_1(x^k, y^k) + \langle \nabla_x H(x^k, y^k), x^{k+1} - x^k\rangle + \frac{L_{k,1}}{2}\|x^{k+1} - x^k\|_2^2. \tag{3.15}$$

Then, the stepsize $\bar{\alpha}_{k,1}$ is taken as the reciprocal of the found value (analogously for $\bar{\alpha}_{k,2}$, where we set $\Psi_2(x, y) := H(x, y) + f_2(y)$). The above *sufficient decrease condition* is a generalization of the *block sufficient decrease condition* (Lemma 3.2.1) when f_1 and f_2 are nonconvex. We will refer to this algorithm as btPALM .

An important contribution to the success of PALM was the convergence proof strategy obtained in [9]. This allowed the design of new convergent alternating minimization algorithms, consisting of a sequence converging to critical points of (3.12). In [9] the function Ψ is assumed to satisfy the so-called Kurdyka-Lojasiewicz property ([9, Definition 3]). However we limit our analysis to the subset of semi-algebraic functions.

We consider problems of the form (3.12), for which we assume that the functions H, f_1 and f_2 satisfy the following minimal assumptions set.

Assumption 3.3.1 ([9, Assumption 2])**.**

(A1) $f_1 : \mathbb{R}^n \to (-\infty, +\infty]$ *and* $f_2 : \mathbb{R}^m \to (-\infty, +\infty]$ *are proper and lower semicontinuous functions such that* $\inf_{\mathbb{R}^n} f_1 > -\infty$ *and* $\inf_{\mathbb{R}^m} f_2 > -\infty$.

(A2) $H : \mathbb{R}^{n\times m} \to \mathbb{R}$ *is continuously differentiable and* $\inf_{\mathbb{R}^{n\times m}} \Psi > -\infty$.

(A3) H *is* L_M *smooth over bounded subsets of* $\mathbb{R}^{n\times m}$*, i.e. for each bounded subsets* $B_1 \times B_2$ *of* $\mathbb{R}^n \times \mathbb{R}^m$ *there exists* $M > 0$ *such that for all* $(x_i, y_i) \in B_1 \times B_2$*,* $i = 1, 2$ *such that*

$$\|(\nabla_x H(x_1, y_1) - \nabla_x H(x_2, y_2), \nabla_y H(x_1, y_1) - \nabla_y H(x_2, y_2))\|_2 \leq M \|(x_1 - x_2, y_1 - y_2)\|_2$$

(A4) The partial gradients $\nabla_x H(x, y)$ *and* $\nabla_y H(x, y)$ *are globally Lipschitz continuous, i.e. there exist nonnegative* $L_1'(y)$ *and* $L_2'(x)$ *such that*

$$\textit{fixed } y, \quad \|\nabla_x H(u, y) - \nabla_x H(v, y)\|_2 \leq L_1'(y) \|u - v\|_2, \ \forall u, v \in \mathbb{R}^n,$$
$$\textit{fixed } x, \quad \|\nabla_y H(x, u) - \nabla_y H(x, v)\|_2 \leq L_2'(x) \|u - v\|_2, \ \forall u, v \in \mathbb{R}^m.$$

(A5) There exist $\lambda_i^+ > 0$*,* $i = 1, 2$ *such that*

$$\sup\{L_1'(y^k);\, k \in \mathbb{N}\} \leq \lambda_1^+, \qquad \sup\{L_2'(x^k);\, k \in \mathbb{N}\} \leq \lambda_2^+. \tag{3.16}$$

(A6) There exist $\lambda_i^- > 0$*,* $i = 1, 2$ *such that*

$$\inf\{L_1'(y^k);\, k \in \mathbb{N}\} \geq \lambda_1^-, \qquad \inf\{L_2'(x^k);\, k \in \mathbb{N}\} \geq \lambda_2^-. \tag{3.17}$$

Remark 3.3.1. *Assumption 3.3.1 (A6) can be avoided by choosing the partial smoothness parameters* L_1' *and* L_2' *safely bounded away from zero, as suggested in [9, Remark 3]. More precisely, given two positive constants* μ_1^- *and* μ_2^- *we set* $L_1'' = \max\{L_1', \mu_1^-\}$ *and* $L_2'' = \max\{L_2', \mu_2^-\}$ *(see Algorithm 14).*

Remark 3.3.2 ([9, Remark 3])**.** *For* H *being twice continuously differentiable, if the sequence generated is bounded, Assumption 3.3.1 (A5) is always satisfied. Furthermore under these conditions also Assumption 3.3.1 (A3) is satisfied as a consequence of the Mean Value Theorem.*

The following result, that will be crucial in the convergence analysis, extends the *sufficient decrease condition* (see Lemma 3.1.3) to the nonconvex setting.

Lemma 3.3.1 (Sufficient decrease, [9, Lemma 2])**.** *Let* $h : \mathbb{R}^m \to \mathbb{R}$ *be an* L_h*-smooth function. Consider a proper and lower semicontinuous function* $\sigma : \mathbb{R}^m \to \mathbb{R}$ *such that* $inf\sigma > -\infty$*. Let* $t > L_h$ *fixed. Then, for any* $u \in dom(\sigma)$ *and any* $u^+ \in \mathbb{R}^m$ *defined by*

$$u^+ \in prox_t^\sigma \left(u - \frac{1}{t} \nabla h(u) \right)$$

we have

$$h(u^+) + \sigma(u^+) \leq h(u) + \sigma(u) - \frac{1}{2}(t - L_h)\|u^+ - u\|_2^2.$$

Proof. By the definition of proximal map we have that

$$u^+ \in \operatorname*{argmin}_{v\in\mathbb{R}^m} \xi(v)$$

where $\xi(v) = \langle v-u, \nabla h(u)\rangle + \frac{t}{2}\|v-u\|_2^2 + \sigma(v)$. Since $\xi(u^+) \leq \xi(v)$, $\forall\ v \in \mathbb{R}^d$, then $\xi(u^+) \leq \xi(u)$, which implies

$$\langle u^+ - u, \nabla h(u)\rangle + \frac{t}{2}\|u^+ - u\|_2^2 + \sigma(u^+) \leq \sigma(u).$$

Combining the latter inequality with the *descent Lemma* [9, Lemma 1], we obtain

$$\begin{aligned} h(u^+) + \sigma(u^+) &\leq h(u) + \langle u^+ - u, \nabla h(u)\rangle + \frac{L_h}{2}\|u^+ - u\|_2^2 + \sigma(u^+) \\ &\leq h(u) + \frac{L_h}{2}\|u^+ - u\|_2^2 + \sigma(u) - \frac{t}{2}\|u^+ - u\|_2^2 \\ &= h(u) + \sigma(u) - \frac{1}{2}(t - L_h)\|u^+ - u\|_2^2. \end{aligned}$$

□

Using the *sufficient decrease Lemma* the convergence of PALM is proved in [9] by using a general methodology which consists in three steps.

Proof methodology 3.3.1 ([62, Theorem 3.1])**.** *Consider a proper, lower semicontinuous and semi-algebraic function $\Psi : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ such that $\Psi > -\infty$. Let $\{z^k\}_{k\geq 0}$ be a bounded sequence generated by an algorithm **A** starting from z^0. Then z^k converges to a critical point of (3.12) if the following conditions are satisfied.*

(C1) There exists a positive scalar $\bar{\rho}_1$ such that

$$\bar{\rho}_1\|z^{k+1} - z^k\|^2 \leq \Psi(z^k) - \Psi(z^{k+1}).$$

(C2) There exists a positive scalar $\bar{\rho}_2$ such that for some $w^k \in \partial\Psi(z^k)$ we have

$$\|w^k\| \leq \bar{\rho}_2\|z^k - z^{k-1}\|,$$

where $\partial\Psi$ denotes the limiting-subdifferential *of Ψ (see [9, Definition 1]).*

(C3) Each limit point in the set $\omega(z^0)$ defined as

$$\begin{aligned} \omega(z^0) = \{\bar{z} \in \mathbb{R}^n \times \mathbb{R}^m :\ &\exists \text{ an increasing sequence of integers } \{k_l\}_{l\geq 0} \text{ such that} \\ &z^{k_l} \to \bar{z} \text{ as } l \to \infty\} \end{aligned}$$

is a critical point of Ψ.

In the sequel we use the notation $z^k = (x^k, y^k)$ for all $k \geq 0$. The following results prove that PALM satisfies conditions (C1)-(C2)-(C3).

Lemma 3.3.2 ([9, Lemma 3])**.** *Assume that Assumption 3.3.1 holds. Let $\{z^k\}_{k\geq 0}$ be the sequence generated by PALM. Then*

(i) the sequence $\{\Psi(z^k)\}_{k\geq 0}$ is nonincreasing and in particular

$$\frac{\rho_1}{2}\|z^{k+1} - z^k\|_2^2 \leq \Psi(z^k) - \Psi(z^{k+1}), \quad \forall k \geq 0,$$

where $\rho_1 = \min\left\{(\gamma_1 - 1)\gamma_1^-, (\gamma_2 - 1)\gamma_2^-\right\}$.

(ii) $\sum_{k=1}^{\infty} \|x^{k+1} - x^k\|_2^2 + \|y^{k+1} - y^k\|_2^2 = \sum_{k=1}^{\infty} \|z^{k+1} - z^k\|_2^2 < \infty$, *and hence* $\lim_{k\to\infty} \|z^{k+1} - z^k\| = 0$.

Proof.

(i) For Assumption 3.3.1, the functions $x \to H(x, y)$ and $y \to H(x, y)$ are L-smooth with moduli $L_1(y)$ and $L_2(x)$ respectively. Then fixing $k \geq 0$ and applying Lemma 3.3.1 with $h(\cdot) = H(\cdot, y^k)$, $\sigma = f_1$ and $t = \eta_1 L_1(y^k) > L_1(y^k)$ we obtain

$$H(x^{k+1}, y^k) + f_1(x^{k+1}) \leq H(x^k, y^k) + f_1(x^k) - \frac{1}{2}\left(\eta_1 L_1(y^k) - L_1(y^k)\right)\|x^{k+1} - x^k\|_2^2.$$

Then applying Lemma 3.3.1 $h(\cdot) = H(x^{k+1}, \cdot)$, $\sigma = f_2$ and $t = \eta_2 L_2(x^{k+1}) > L_2(x^{k+1})$ we have

$$H(x^{k+1}, y^{k+1}) + f_2(y^{k+1}) \leq H(x^{k+1}, y^k) + f_2(y^k) - \frac{1}{2}\left(\eta_2 L_2(x^{k+1}) - L_2(x^{k+1})\right)\|y^{k+1} - y^k\|_2^2.$$

Summing the last two inequalities gives the following

$$\begin{aligned}\Psi(z^k) - \Psi(z^{k+1}) &\geq \frac{1}{2}(\eta_1 - 1)\, L_1(y^k)\|x^{k+1} - x^k\|_2^2 + \frac{1}{2}(\eta_2 - 1)\, L_2(x^{k+1})\|y^{k+1} - y^k\|_2^2 \\ &\geq \frac{1}{2}(\eta_1 - 1)\, \lambda_1^-\|x^{k+1} - x^k\|_2^2 + \frac{1}{2}(\eta_2 - 1)\, \lambda_2^-\|y^{k+1} - y^k\|_2^2,\end{aligned}$$

where for the last inequality we used the fact that the Lipschitz constants are bounded from above. Now choosing $\rho_1 = \min\left\{(\gamma_1 - 1)\gamma_1^-, (\gamma_2 - 1)\gamma_2^-\right\}$ gives the result.

(ii) Since

$$\frac{\rho_1}{2}\|z^{k+1} - z^k\|_2^2 \leq \Psi(z^k) - \Psi(z^{k+1}),$$

holds for every $k \geq 0$, by summing from $k = 0, \ldots, N-1$ we obtain

$$\sum_{k=0}^{N-1} \left\|x^{k+1} - x^k\right\|_2^2 + \left\|y^{k+1} - y^k\right\|_2^2 = \sum_{k=0}^{N-1} \left\|z^{k+1} - z^k\right\|_2^2 \leq \frac{2}{\rho_1} \left(\Psi(z^0) - \Psi(z^N)\right).$$

Since the sequence $\left\{\Psi(z^k)\right\}_{k\geq 0}$ is nonincreasing and is assumed to be bounded from below, it converges to a real number $\bar{\Psi}$. Thus, taking the limit for $N \to \infty$ we obtain that

$$\sum_{k=1}^{\infty} \|z^{k+1} - z^k\|_2^2 < \infty.$$

□

Lemma 3.3.3 ([9, Lemma 4])**.** *Assume that Assumption 3.3.1 hold. Let $\{z^k\}_{k\geq 0}$ be the sequence generated by PALM and assume that the sequence is bounded. For each positive integer k define*

$$a_x^k := c_{k-1}\left(x^{k-1} - x^k\right) + \nabla_x H(x^k, y^k) - \nabla_x H(x^{k-1}, y^{k-1}),$$

and

$$a_y^k := d_{k-1}\left(y^{k-1} - y^k\right) + \nabla_y H(x^k, y^k) - \nabla_y H(x^k, y^{k-1}),$$

where $c_{k-1} = \eta_1 L_1(y^{k-1})$ and $d_{k-1} = \eta_2 L_2(x^k)$ Then $\left(a_x^k; a_y^k\right) \in \partial\Psi(x^k, y^k)$ and there exists $M > 0$ s.t.

$$\left\|\left(a_x^k; a_y^k\right)\right\|_2 \leq \left\|a_x^k\right\|_2 + \left\|a_y^k\right\|_2 \leq (2M + 3\rho_2)\|z^k - z^{k-1}\|_2, \quad \forall k \geq 1, \tag{3.18}$$

with $\rho_2 = \max\{\gamma_1\lambda_1^+, \gamma_2\lambda_2^+\}$.

Lemma 3.3.4 ([9, Lemma 5])**.** *Assume that Assumption 3.3.1 holds. Let $\{z^k\}_{k\geq 0}$ be the sequence generated by PALM and assume that the sequence is bounded. Then*

$$\emptyset \neq \omega(z^0) \subset crit\Psi,$$

where critΨ denotes the set of critical points of Ψ, i.e. the set of points whose subdifferential contains 0.

We report the proof of Lemma 3.3.4 since it will be useful in the following chapter to prove the convergence of the spectral PALM.

Proof. Let $z^* = (x^*, y^*)$ be a limit point of $\{z^k\}_{k\geq 0}$ and let $\left\{x^{k_q}, y^{k_q}\right\}_{q\geq 0}$ a subsequence of $\{z^k\}_{k\geq 0}$ converging to z^*. Now consider the update rule of PALM. Given $k \geq 0$ and denoting as $c_k = \eta_1 L_1(y^k)$, we have that

$$x^{k+1} \in \operatorname*{argmin}_{x\in\mathbb{R}^n} \left\{\langle x - x^k, \nabla_x H(x^k, y^k)\rangle + \frac{c_k}{2}\|x - x^k\|_2^2 + f_1(x)\right\}.$$

Thus,

$$\begin{aligned}&\langle x^{k+1}-x^k,\nabla_x H(x^k,y^k)\rangle+\frac{c_k}{2}\|x^{k+1}-x^k\|_2^2+f_1(x^{k+1})\\&\leq\langle x^*-x^k,\nabla_x H(x^k,y^k)\rangle+\frac{c_k}{2}\|x^*-x^k\|_2^2+f_1(x^*)\end{aligned}$$

Setting $k=k_q-1$ in the above inequality and letting q goes to ∞ we get

$$\lim_{q\to\infty} f_1(x^q)\leq\lim_{q\to\infty}\sup\left(\langle x^*-x^k,\nabla_x H(x^k,y^k)\rangle+\frac{c_k}{2}\|x^*-x^k\|_2^2\right)+f_1(x^*),$$

since $\{x^k\}_{k\geq 0}$ and $\{c^k\}_{k\geq 0}$ are bounded, ∇H is continuous and the difference between successive iterates tends to zero. Now observing that $\lim_{q\to\infty} x^{k_q-1}=x^*$, we obtain that $\lim_{q\to\infty}\sup f_1(x^{k_q})\leq f_1(x^*)$. This, together with the fact that f_1 is lower semicontinuous (i.e. $\lim_{q\to\infty} f_1(x^{k_q})\geq f_1(x^*)$), implies that $\lim_{q\to\infty} f_1(x^{k_q})=f_1(x^*)$. Using the same procedure for f_2 we obtain that

$$\lim_{q\to\infty}\Psi(x^{k_q},y^{k_q})=\Psi(x^*,y^*).$$

From Lemmas 3.3.2 and 3.3.3, $(a_x^k,a_y^k)\in\partial\Psi(x^k,y^k)$ and $\lim_{k\to\infty}(a_x^k,a_y^k)=(0,0)$ which implies that $(0,0)\in\partial\Psi(x^*,y^*)$ (see [9, Remark 1]). This completes the proof. □

Remark 3.3.3 ([9, Lemma 5])**.** *Under the assumptions of the previous lemma we also have that*

(i) $\lim_{k\to\infty} dist(z^k,\omega(z^0))=0$*;*

(ii) $\omega(z^0)$ *is a non-empty, compact and connected set.*

(iii) The objective function Ψ *is finite and constant on* $\omega(z^0)$*.*

The following theorem gives the main convergence result proved for the PALM algorithm.

Theorem 3.3.1 ([9, Theorem 1])**.** *Suppose that* Ψ *is a semi-algebraic function such that Assumption 3.3.1 holds. Let* $\{z^k\}=\{(x^k,y^k)\}$ *be a bounded sequence generated by PALM in Algorithm 14. Then the sequence* $\{z_k\}$ *has finite length, that is* $\sum_{k=1}^{\infty}\left\|z^{k+1}-z^k\right\|<\infty$*, and converges to a critical point* z^* *of* Ψ*.*

Remark 3.3.4 ([9, Section 3.6])**.** *Algorithm 14 can be extended to the general setting involving* $p>2$ *blocks, that is problems of the form*

$$\min_{x_i\in\mathbb{R}^{n_i}}\Psi(x_1,\ldots,x_p)\quad with\quad\Psi(x_1,\ldots,x_p)=H(x_1,\ldots,x_p)+\sum_{i=1}^{p}f_i(x_i),\tag{3.19}$$

for which Theorems 3.3.1 holds.

Inertial variants of PALM have been later proposed with the aim of accelerating the convergence of the original algorithm [32, 42, 62]. A first version, named iPALM [62] combines the last two iterates using the so-called heavy-ball method, and has numerically been applied to solve sparse NMF, BID and image deconvolution using Dictionary Learning. A further inertial version based on the use of surrogate functions was developed in [42] and has been used for solving sparse NMF and matrix completion problems. All these variants enjoy the convergence properties of the original PALM, and are based on the Lipschitz constants explicitly available for all the addressed applications. When these constants are not explicitly known, then a backtracking scheme can be employed to approximate their action [5, 6], so that convergence results still hold.

Chapter 4

The spectral Proximal Alternating Linearized Minimization (sPALM) algorithm

In this chapter we briefly describe the main properties of a special class of gradient-type methods for smooth optimization, i.e. the spectral gradient methods, and discuss how to exploit them into the PALM algorithmic scheme. As a result we present the new spectral Proximal Alternating Linearized Minimization (sPALM) algorithm which enjoys the same convergence properties of PALM but is expected to have a better practical behavior.

4.1 The spectral steplength

Spectral gradient methods are well-known optimization strategies for the solution of large scale unconstrained and constrained smooth optimization problems [7, 8]. These algorithms are rather appealing for their simplicity, low-cost per iteration (gradient-type algorithms) and good practical performance due to a clever choice of the steplength. The key to the success of these approaches, also known as Barzilai-Borwein methods from the pioneering work [3], lies in the explicit use of first-order information of the cost function on the one hand and, on the other hand, in the implicit use of second-order information embedded in the steplength through a rough approximation of the cost function Hessian. This speeds up the convergence and avoids the explicit computation of the Hessian, typical of second-order methods, that would enhance the computational complexity. While spectral gradient methods were first proposed for convex quadratic problems, they have been widely used in a large variety of more general contexts [44, 54, 68].

Let $H : \mathbb{R}^n \to \mathbb{R}$ be a real valued differentiable function and consider the following

optimization problem

$$\min_{x\in\mathbb{R}^n} H(x). \tag{4.1}$$

Then the Barzilai-Borwein (BB) iteration is defined as

$$x_{k+1} = x_k - \alpha_k \nabla H(x_k),$$

where α_k is the solution of one of the following least-squares problems:

$$\alpha_{k+1}^{BB1} = \operatorname*{argmin}_{\alpha} \left\| \frac{1}{\alpha} s_k - g_k \right\|_2 \quad \text{and} \quad \alpha_{k+1}^{BB2} = \operatorname*{argmin}_{\alpha} \left\| s_k - \alpha g_k \right\|_2, \tag{4.2}$$

with $s_k = x_{k+1} - x_k$ and $g_k = \nabla H(x_{k+1}) - \nabla H(x_k)$. The definitions in (4.2) correspond to use a Quasi-Newton secant method for the solution of problem (4.1) where the Hessian matrix at the current iterate is approximated by a multiple of the identity matrix [8]. Indeed, the Quasi-Newton method is based on the following iteration

$$x_{k+1} = x_k - B_k^{-1} \nabla H(x^k), \tag{4.3}$$

and, given $B_0 \in \mathbb{R}^{n\times n}$ as an initial data, $B_k \in \mathbb{R}^{n\times n}$, $k \geq 1$ satisfies the secant equations i.e.,

$$B_k s_{k-1} = g_{k-1} \quad \text{with} \quad s_{k-1} = x_k - x_{k-1},\ \ g_{k-1} = \nabla H(x_k) - \nabla H(x_{k-1}). \tag{4.4}$$

Letting $B_k = \alpha_k^{-1} I$ where I denotes the identity matrix and imposing condition (4.4), Barzilai-Borwein derived the two steplengths in (4.2) which have the following explicit forms:

$$\alpha_{k+1}^{BB1} = \frac{\langle s_k, s_k\rangle}{\langle s_k, g_k\rangle} \quad \text{and} \quad \alpha_{k+1}^{BB2} = \frac{\langle s_k, g_k\rangle}{\langle g_k, g_k\rangle}. \tag{4.5}$$

Note that α_k^{BB1} is the inverse of the Rayleigh quotient corresponding to the average of the Hessian matrix $\int_0^1 \nabla^2 H(x_k + t s_k) dt$, that is always bounded above and below by the minimum and the maximum eigenvalues of the Hessian of H [7, 8]. Thus the BB methods are also known as *spectral* methods.

To make the stepsizes α_k^{BB1} and α_k^{BB2} at each step safely bounded away from zero and bounded above, the BB method uses one of the following stepsizes

$$\alpha_{k+1} = \max\left\{\alpha_{\min}, \min\left\{\alpha_{k+1}^{BB1}, \alpha_{\max}\right\}\right\} \tag{4.6}$$

$$\alpha_{k+1} = \max\left\{\alpha_{\min}, \min\left\{\alpha_{k+1}^{BB2}, \alpha_{\max}\right\}\right\} \tag{4.7}$$

where α_{k+1}^{BB1} and α_{k+1}^{BB2} are given in (4.5). A variety of different rules based on suitable adaptive combinations of α_{k+1}^{BB1} and α_{k+1}^{BB2} have been proposed in the literature. For example the Adaptive Barzilai-Borwein (ABB) method [85, 31] selects one of the two stepsizes according to a given parameter $\tau > 0$. More precisely, if $\frac{\alpha_{k+1}^{BB2}}{\alpha_{k+1}^{BB2}} < \tau$, then

$\alpha_{k+1} = \alpha_{k+1}^{BB2}$, otherwise $\alpha_{k+1} = \alpha_{k+1}^{BB1}$. In general it was observed experimentally that alternating the two stepsizes along iterations is beneficial for the performance. For further details on an alternating strategy for spectral stepsize selection we refer to [54] and references therein.

Differently from other choices of the stepsizes, the BB ones do not guarantee a decrease of the objective function at each step. However global convergence is guaranteed for convex quadratic problems ([63]) and in the non quadratic case when combined with a (generally non monotone) globalization scheme ([64]).

4.2 The spectral PALM algorithm

Starting from the PALM framework, we propose spectral PALM (sPALM) that, for each coordinate block, employs a *spectral* gradient step in the smooth part of the operator, while maintaining a proximal step for the non-smooth part. More precisely, sPALM uses a spectral stepsize in each variable bock in combinations with an Armijo-type backtracking strategy ensuring the overall convergence.

The use of enriched proximal steps for solving composite optimization problems is not new, see, e.g., [81]. The Cyclic Block Coordinate Gradient Projection algorithm, proposed in [10] and mentioned in the previous chapter, makes use of spectral stepsizes when applied to non-negative matrix factorizations. However, these steps are used to determine an approximate solution of the minimization problem for each variable block, and they do not use information from the previous iteration of the alternating algorithm. To the best of our knowledge, the use of spectral stepsizes embedded in an alternating algorithm for nonconvex non-smooth problems of the form (3.12) has remained so far unexplored.

We now discuss how to extend the derivation of spectral stepsizes introduced in the previous section for problem (4.1), to the more general nonconvex and non-smooth problem (3.12). For the sake of clarity, we rewrite here problem (3.12):

$$\min_{x,y} \Psi(x,y) \quad \text{with} \quad \Psi(x,y) := H(x,y) + f_1(x) + f_2(y), \tag{4.8}$$

where the functions H, f_1 and f_2 satisfy Assumption 3.3.1. Consider the case of the partial Hessian $\nabla_{xx}H$ (the case of $\nabla_{yy}H$ is analogous). For a given iteration k, let

$$s_k = x_{k+1} - x_k \quad \text{and} \quad g_k = \nabla_x H(x_{k+1}, y_k) - \nabla_x H(x_k, y_k)$$

be the difference between two consecutive iterates and corresponding gradient values. Then $\nabla_{xx}H(x_{k+1}, y_k)$ is approximated by $\alpha_{k+1}^{-1} I$ where the positive scalar α_{k+1} is defined by one of the following BB values

$$\alpha_{k+1}^{BB1} = \operatorname*{argmin}_{\alpha} \left\| \frac{1}{\alpha} s_k - g_k \right\|_2 \quad \text{and} \quad \alpha_{k+1}^{BB2} = \operatorname*{argmin}_{\alpha} \|s_k - \alpha g_k\|_2$$

that is

$$\alpha_{k+1}^{BB1} = \frac{\langle s_k, s_k \rangle}{\langle s_k, g_k \rangle} \quad \text{and} \quad \alpha_{k+1}^{BB2} = \frac{\langle s_k, g_k \rangle}{\langle g_k, g_k \rangle}. \tag{4.9}$$

We report in Algorithm 15 a simple alternating rule based on [37] that gives the best results in our numerical experiments (see Section 7.2). Clearly, other rules can be equally adapted within sPALM. The inclusion of threshold values in Algorithm 15 ensures that the α's remain bounded. The overall sPALM scheme is reported in Algorithm 16, where the following notation is used:

$$\Psi_1(x, y) := H(x, y) + f_1(x) \quad \text{and} \quad \Psi_2(x, y) := H(x, y) + f_2(y). \tag{4.10}$$

Algorithm 15 Computation of the spectral stepsize $\alpha_{k+1,i}$, $i = 1$ or $i = 2$

Input: $0 < \alpha_{\min} < \alpha_{\max}$,

$$s_{k,i} = \begin{cases} x_{k+1} - x_k & \text{if } i = 1 \\ y_{k+1} - y_k & \text{if } i = 2, \end{cases} \quad \text{and} \quad g_{k,i} = \begin{cases} \nabla_x H(x_{k+1}, y_k) - \nabla_x H(x_k, y_k) & \text{if } i = 1 \\ \nabla_y H(x_{k+1}, y_{k+1}) - \nabla_y H(x_{k+1}, y_k) & \text{if } i = 2, \end{cases}$$

if $\langle s_{k,i}, g_{k,i} \rangle > 0$ **then**
  if k is odd **then**
    $\alpha_{k+1,i} = \max\left\{\alpha_{\min}, \min\left\{\alpha_{k+1,i}^{BB1}, \alpha_{\max}\right\}\right\}$ with $\alpha_{k+1,i}^{BB1} = \frac{\langle s_{k,i}, s_{k,i} \rangle}{\langle s_{k,i}, g_{k,i} \rangle}$
  else
    $\alpha_{k+1,i} = \max\left\{\alpha_{\min}, \min\left\{\alpha_{k+1,i}^{BB2}, \alpha_{\max}\right\}\right\}$ with $\alpha_{k+1,i}^{BB2} = \frac{\langle s_{k,i}, g_{k,i} \rangle}{\langle g_{k,i}, g_{k,i} \rangle}$
  end if
else
  $\alpha_{k+1,i} = 1$
end if

Remark 4.2.1. *Conditions (4.12) and (4.14) in Algorithm 16 are satisfied in a finite number of backtracking steps. For instance, from the* descent lemma*, see e.g. [9, Lemma 1], for any x defined by $x = \operatorname{prox}_{1/\alpha}^{f_1}(x_k - \alpha \nabla_x H(x_k, y_k))$ we have that*

$$\Psi_1(x, y_k) \leq \Psi_1(x_k, y_k) - \frac{1}{2}\left(\frac{1}{\alpha} - L_1(y_k)\right)\|x - x_k\|_2^2,$$

and then condition (4.12) is satisfied when $\alpha \leq \frac{1-\delta_1}{L_1(y_k)}$. Therefore, the the backtracking terminates with $\bar{\alpha}_k \geq \min\left\{\alpha_{k,1}, \frac{\rho_1(1-\delta_1)}{L_1(y_k)}\right\} \geq \min\left\{\alpha_{\min}, \frac{\rho_1(1-\delta_1)}{L_1(y_k)}\right\}$. Analogously for (4.14).

In the following we set up the theoretical tools for proving a convergence result analogous to that of Theorem 3.3.1 by exploiting the Proof methodology 3.3.1 discussed in the previous chapter.

Algorithm 16 sPALM

1: **Input**: $(x_0, y_0) \in \mathbb{R}^n \times \mathbb{R}^m$, $\rho_1, \delta_1, \rho_2, \delta_2 \in (0,1)$, $0 < \alpha_{\min} < \alpha_{\max}$, $\alpha_{0,1}, \alpha_{0,2} \in [\alpha_{\min}, \alpha_{\max}]$.
2: **for** $k = 0, 1, \ldots,$ **do**
3: *Update* x: Set

$$x_{k+1} = \text{prox}^{f_1}_{1/\bar{\alpha}_{k,1}}\left(x_k - \bar{\alpha}_{k,1}\nabla_x H(x_k, y_k)\right) \tag{4.11}$$

where $\bar{\alpha}_{k,1} = \rho_1^{i_k}\alpha_{k,1}$ and i_k is the smallest nonnegative integer for which the following condition is satisfied,

$$\Psi_1(x_{k+1}, y_k) < \Psi_1(x_k, y_k) - \frac{\delta_1}{2\bar{\alpha}_{k,1}}\|x_{k+1} - x_k\|_2^2 \tag{4.12}$$

4: Compute $\alpha_{k+1,1} \in [\alpha_{\min}, \alpha_{\max}]$ using Algorithm 15 with $s_{k,1} = x_{k+1} - x_k$ and $g_{k,1} = \nabla_x H(x_{k+1}, y_k) - \nabla_x H(x_k, y_k)$.
5: *Update* y: Set

$$y_{k+1} = \text{prox}^{f_2}_{1/\bar{\alpha}_{k,2}}\left(y_k - \bar{\alpha}_{k,2}\nabla_y H(x_{k+1}, y_k)\right) \tag{4.13}$$

where $\bar{\alpha}_{k,2} = \rho_2^{j_k}\alpha_{k,2}$ and j_k is the smallest nonnegative integer for which the following condition is satisfied,

$$\Psi_2(x_{k+1}, y_{k+1}) < \Psi_2(x_{k+1}, y_k) - \frac{\delta_2}{2\bar{\alpha}_{k,2}}\|y_{k+1} - y_k\|_2^2 \tag{4.14}$$

6: Compute $\alpha_{k+1,2} \in [\alpha_{\min}, \alpha_{\max}]$ using Algorithm 15 with $s_{k,2} = y_{k+1} - y_k$ and $g_{k,2} = \nabla_y H(x_{k+1}, y_{k+1}) - \nabla_y H(x_{k+1}, y_k)$.
7: **end for**

Lemma 4.2.1. *Suppose Assumption 3.3.1 holds. Let* $\{z_k\} = \{(x_k, y_k)\}$ *be a bounded sequence generated by sPALM from a starting point* z_0 *and let* $\omega(z_0)$ *be the set of all limit points of* $\{z_k\}$*. Then the following conditions hold.*

C1) *There exists a positive scalar* γ_1 *such that* $\gamma_1\|z_{k+1} - z_k\|_2^2 < \Psi(z_k) - \Psi(z_{k+1})$*;*

C2) *There exists a positive scalar* γ_2 *such that for some* $w_k \in \partial\Psi(z_k)$ *we have* $\|w_k\|_2 < \gamma_2\|z_k - z_{k-1}\|_2$*, for* $k = 0, 1, \ldots$*;*

C3) *Each limit point in the set* $\omega(z_0)$ *is a critical point for* Ψ*.*

Proof. We first observe that the stepsizes $\bar{\alpha}_{k,1}$ and $\bar{\alpha}_{k,2}$ remain bounded for all k. Indeed, since $\alpha_{k,1} \in [\alpha_{\min}, \alpha_{\max}]$ we have that $\bar{\alpha}_{k,1} = \rho_1^{i_k}\alpha_{k,1} < \alpha_{\max}$ as $\rho_1 \in (0,1)$ and $i_k \geq 0$. Moreover, $\bar{\alpha}_{k,1}$ is at least $\rho_1^{i_k}\alpha_{\min}$ since condition (4.12) is satisfied in a finite number of steps (see Remark 4.2.1); similarly for $\bar{\alpha}_{k,2}$.

Item C1) can be proved as follows. Fix $k \geq 0$ and sum the inequalities (4.12) and (4.14), so as to obtain

$$\begin{aligned}\Psi_1(x_{k+1}, y_k) + \Psi_2(x_{k+1}, y_{k+1}) &< \Psi_1(x_k, y_k) + \Psi_2(x_{k+1}, y_k) \\ &\quad - \frac{\delta_1}{2\bar{\alpha}_{k,1}}\|x_{k+1} - x_k\|_2^2 - \frac{\delta_2}{2\bar{\alpha}_{k,2}}\|y_{k+1} - y_k\|_2^2.\end{aligned}$$

Recalling the definition of Ψ_1 and Ψ_2 (4.10) we obtain

$$\begin{aligned}\Psi(x_{k+1}, y_{k+1}) &< \Psi(x_k, y_k) - \frac{\delta_1}{2\bar{\alpha}_{k,1}}\|x_{k+1} - x_k\|_2^2 - \frac{\delta_2}{2\bar{\alpha}_{k,2}}\|y_{k+1} - y_k\|_2^2 \\ &< \Psi(x_k, y_k) - \frac{\delta_1}{2\alpha_{\max}}\|x_{k+1} - x_k\|_2^2 - \frac{\delta_2}{2\alpha_{\max}}\|y_{k+1} - y_k\|_2^2,\end{aligned}$$

from which Lemma 4.2.1 - C1) follows, with $\gamma_1 = \frac{1}{2\alpha_{\max}} \min\{\delta_1, \delta_2\}$.

The proofs of items C2) and C3) follow the lines of the proofs of Lemmas 3.3.3 and 3.3.4. Indeed, from the definition of the proximal map and the iterative steps (4.11) and (4.13), we have that

$$x_k = \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} \left\{ \langle \nabla_x H(x_{k-1}, y_{k-1}), x - x_{k-1} \rangle + \frac{1}{2\bar{\alpha}_{k,1}}\|x - x_{k-1}\|_2^2 + f_1(x) \right\}$$

and

$$y_k = \underset{y \in \mathbb{R}^m}{\operatorname{argmin}} \left\{ \langle \nabla_y H(x_k, y_{k-1}), y - y_{k-1} \rangle + \frac{1}{2\bar{\alpha}_{k,2}}\|y - y_{k-1}\|_2^2 + f_2(y) \right\}.$$

Using the fact that $\bar{\alpha}_{k,1}$ and $\bar{\alpha}_{k,2}$ are bounded for all k the results follow. □

We can now state a convergence result for sPALM.

Theorem 4.2.1. *Suppose that Ψ is semi-algebraic such that Assumption 3.3.1 (A1)-(A4) hold. Let $\{z_k\} = \{(x_k, y_k)\}$ be a bounded sequence generated by sPALM. Then the sequence $\{z_k\}$ has finite length and converges to a critical point z^* of Ψ.*

Proof. The proof follows by using Lemma 4.2.1 and applying the Proof methodology 3.3.1 described in Section 3.3. □

Remark 4.2.2. *Algorithms 15-16 can be extended to the general setting involving $p > 2$ blocks, that is problems of the form*

$$\min_{x_i \in \mathbb{R}^{n_i}} \Psi(x_1, \ldots, x_p) \quad \textit{with} \quad \Psi(x_1, \ldots, x_p) := H(x_1, \ldots, x_p) + \sum_{i=1}^{p} f_i(x_i), \tag{4.15}$$

for which Theorem 4.2.1 holds. When variable blocks are matrices, all PALM-type algorithms can be extended to the matrix optimization setting by using the trace matrix scalar product and the Frobenius norm in place of the vector scalar product and the vector 2-norm, respectively.

Chapter 5

Matrix and tensor Dictionary Learning (DL) problem

Sparse representation of data has become an important tool in a variety of contexts such as image classification and compression, observation denoising and equation solving. In the context of image classification, Dictionary Learning (DL) is among the leading sparsity promoting techniques, and we refer to [26] and [53] for an overview of all applications of Dictionary Learning in image processing in general. More precisely, in image denoising the DL model is able to separate the original image from noise, in image classification it is able to select the most important features of each class of images.

Given a set of data Y Dictionary Learning aims to find a matrix D called dictionary and a sparse matrix X to represent $Y \approx DX$, as shown in Figure 5.1, under specific constraints on D and X.

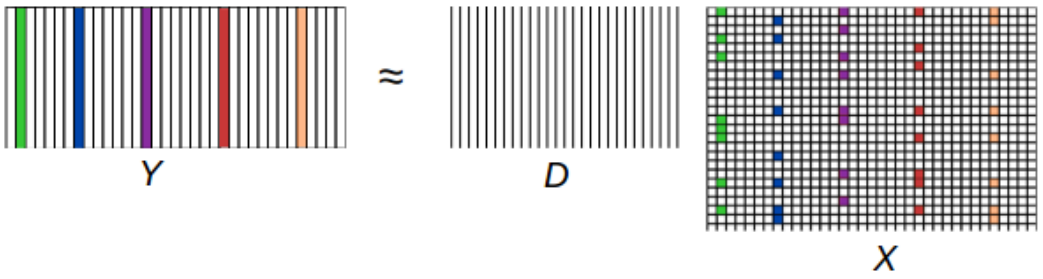


Figure 5.1: Dictionary learning problem.

A distinct feature of this approximate factorization is that the dictionary is usually overcomplete, i.e. the number of columns (called *atoms*) is greater than the number of rows. Overcompleteness has important drawbacks in terms of shiftability, i.e. the property of invariance of the dictionary under specific geometric transformations, as observed in [70].

As an example in image processing, Figure 5.2 shows a typical matrix D, obtained

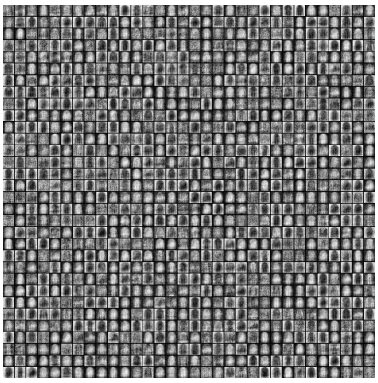


Figure 5.2: Dictionary D after 10 iterations for the Fashion MNIST dataset using PALM.

here for the Fashion MNIST data set (see Appendix A). In the figure each column of D is reported as a 28×28 image, displaying the prototype of a corresponding dressing item in the dataset. The sparse matrix X has the ability to cluster the data by selecting for each image contained in Y the columns of D that are more significant in terms of representation.

Originally fixed dictionaries were used and the Dictionary Learning problem reduced to find a sparse matrix X given a set of data Y and a dictionary D. However despite the simplicity of fixed dictionaries, they suffered from limited expression. For example, a dictionary based on Discrete Cosine Transform (DCT) may represent in a sparse manner some images, but it can fail with others (see [26] and references therein). All these considerations led to the development of overcomplete dictionaries learnt from data, as in the seminal work [1] where Ahron, Elad and Bruckstein proposed the K-SVD algorithm. From a computational view point going from fixed to learnt dictionaries poses several challenges, especially when further constraints, that will be discussed in the following sections, are included.

The increased need to analyze multidimensional data brought to various tensor formulations of the DL problem with the aim of preserving data structure and feature heterogeneity. For example, in image processing, preserving the neighboring relations could be useful to enhance the classification performance of the algorithm as will be clear from the following discussion. Unfortunately numerical methods based on these tensor formulations are usually not supported with global convergence analysis. In this chapter, we review some of the most important matrix and tensor Dictionary Learning formulations and related algorithms.

5.1 Matrix Dictionary Learning problem

Dictionary Learning (DL) consists of solving a two variable optimization problem, which can be formulated in several ways, not necessarily equivalent. The first formulation introduced by Olshausen and Field [59] enforces the sparse representation of the data Y using a penalty function $\Phi(X)$ as follows

$$\min_{D,X} \|Y - DX\|_F^2 + \Phi(X) \quad s.t. \quad D \in \Omega_{n,k}, \tag{5.1}$$

where

$$\Omega_{n,k} = \{D \in \mathbb{R}^{n\times k} : \ \|d_i\|_2 = 1, i = 1, \dots, k\}, \tag{5.2}$$

and $\Phi(X)$ can be chosen e.g. as the ℓ_0 or ℓ_1-norm. In general the number of column of Y is much larger than the number of atoms k of the dictionary, nevertheless (5.1) can be considered even when this condition is not satisfied. Notice that without the constraint on the norm of the columns of D, the penalization becomes ineffective since the norm of the dictionary can go to infinity [53]. Depending on the application, other variants of (5.1) can be considered. For example, in denoising applications one is not interested in recovering the data Y exactly due to the presence of noise. Thus, the following variant of (5.1) can be considered.

$$\min_{X} \Phi(X) \quad s.t. \ \ \|Y - DX\|_F^2 \leq \epsilon \quad D \in \Omega_{n,k}, \tag{5.3}$$

where ϵ is a positive constant depending on the noise level of the original data, when known. In our image classification context we use a predefined maximum number τ of atoms to represent each image contained in Y. This corresponds to selecting from the dictionary the most important information for each class. In this setting, the DL formulation becomes

$$\min_{D,X} \|Y - DX\|_F^2 \quad s.t. \quad D \in \Omega_{n,k} \ \ X \in \Gamma_{k,p}^{(\tau)}, \tag{5.4}$$

where

$$\Gamma_{k,p}^{(\tau)} = \{X \in \mathbb{R}^{k\times p} : \|x_i\|_0 < \tau, i = 1, \dots, p\}, \tag{5.5}$$

is the set of matrices with at most τ non zero elements per column and usually with $p > k$.

This minimization problem is NP-hard, see e.g. [26, 53], and nonconvex. Non-convexity comes from two sources: the sparsity promoting functional ℓ_0-norm and the bi-linearity between the dictionary D and the sparse representation X. In addition, the ℓ_0-norm makes the problem non-smooth. Another important property of (5.4) is that it admits multiple global minima $(D^*P, P^{-1}X^*)$, where (D^*, X^*) is a solution of (5.4) and P is a signed permutation matrix. A complete analysis on the properties of the Dictionary Learning problem can be found in [26] and [53]. A numerical solution is

Algorithm 17 Dictionary Learning [26, Algorithm 3.1]

1: **Input**: data matrix $Y \in \mathbb{R}^{n\times p}$, initial dictionary matrix $D \in \mathbb{R}^{n\times k}$
2: **for** $it = 1, \dots,$ **do**
3: *Sparse coding*: (Approximately) solve the problem

$$\min_X \|Y - DX\|_F^2 \quad s.t \quad X \in \Gamma_{k,p}^{(\tau)}$$

4: *Dictionary update*: (Approximately) solve the problem

$$\min_D \|Y - DX\|_F^2 \quad s.t. \quad D \in \Omega_{n,k}$$

5: **end for**
6: **Output**: D, X

usually computed using alternating minimization approaches. More precisely, the minimization problem in D (dictionary update) is solved while keeping X fixed and then the minimization problem in X (sparse coding) is solved while D is fixed. The general scheme of this procedure is reported in Algorithm 17. Note that, as will be explained later, some DL algorithms, like K-SVD, perform an update of the sparse matrix X also in the dictionary update step. In the next sections we report some of the most relevant strategies to solve both the sparse coding and the dictionary update step.

5.1.1 Sparse coding

The sparse coding step consists in solving the minimization problem

$$\min_X \|Y - DX\|_F^2 \quad s.t \quad X \in \Gamma_{k,p}^{(\tau)},$$

which is nonconvex and NP-hard. It can be solved in several ways, for example considering convex relaxations of the problem [26, Chapter 1]. However one of the most popular strategies consists in using greedy algorithms such as OMP (Orthogonal Matching Pursuit). The latter despite the lack of convergence guarantees under general assumptions has good practical behaviour. For each column of Y, named y_i, we compute its sparse representation x in the overcomplete "basis" D. The idea is to build the support of X incrementally, as explained in Algorithm 18. We start from an empty support $S = \emptyset$. Then at each iteration the atom j^* which is most correlated with the residual is added to the support $S = S \cup j^*$. The new coefficient of the sparse representation of y are recomputed as the solution of the following least-squares problem

$$\min_x \|y - D_S x\|_2, \tag{5.6}$$

Algorithm 18 OMP [26, Algorithm 1.1]

Input: data vector $y_i \in \mathbb{R}^n$, dictionary matrix $D \in \mathbb{R}^{n \times k}$, approximation precision ϵ, maximum number of non zero elements τ.
Initialize the vector $x = 0$, $r^{(0)} = y_i$, $S^{(0)} = support(x) = \emptyset$
for $it = 0, \ldots, \tau - 1$ **do**
  Select Atom:
  $j^* = argmax_j \frac{|d_j^T r^{(it)}|}{\|d_j\|_2}$
  $S^{(it+1)} = S^{(it)} \cup j^*$:
  Update the solution estimate:
  $x^{(it+1)}_{S^{(it+1)}} = D^{\dagger}_{S^{(it+1)}} y_i$
  $r^{(it+1)} = y_i - D_{S^{(it+1)}} x^{(it+1)}_{S^{(it+1)}}$
  if $\|r^{(it+1)}\|_2 < \epsilon$ **stop**
end for
Output: vector $x \in \mathbb{R}^k$.

where $D_S \in \mathbb{R}^{n \times |S|}$ is the restriction of the dictionary D to the current support S. The procedure ends either when the maximum number of non zero elements in x is achieved or when the norm of the residual is below a threshold ϵ ([26]). In order to reduce the

Algorithm 19 OMP-QR [74]

Input: data vector $y_i \in \mathbb{R}^n$, dictionary matrix $D \in \mathbb{R}^{n \times k}$, approximation precision ϵ, maximum number of non zero elements τ.
Initialize the vector $x = 0$, $r^{(0)} = y_i$, $S^{(0)} = support(x) = \emptyset$
for $it = 0, \ldots, \tau - 1$ **do**
  Select Atom:
  $j^* = argmax_j \frac{|d_j^T r^{(it)}|}{\|d_j\|_2}$
  $S^{(it+1)} = S^{(it)} \cup j^*$:
  Compute the QR decomposition of $D_{S^{(it+1)}} = Q^{(it+1)} R^{(it+1)}$ starting from the decomposition of $D_{S^{(it)}}$: $Q^{(it+1)} = [Q^{(it)}\ \tilde{q}]$
  $r^{(it+1)} = r^{(it)} - \tilde{q}\tilde{q}^T y_i$
  if $\|r^{(it+1)}\|_2 < \epsilon$ stop
end for
Compute the solution estimate: $x = \left(R^{(it+1)}\right)^{-1} \left(Q^{(it+1)}\right)^T y_i$
Output: vector $x \in \mathbb{R}^k$.

computational complexity of OMP, several strategies can be used as described in [74]. In our experiments we considered a variant of Algorithm 18, described in [74, Section 2.3], where the QR decomposition of D_S is used. At each iteration a column of D is added to the support and the QR decomposition can be updated. Then the residual is computed without explicitly determining the sparse solution x. We refer to this variant

as OMP-QR. The main difference between OMP and OMP-QR lies in the computation of x. In the latter x is computed only once when the number of its non-zero elements is determined, while in OMP x is computed every time an atom of D is added to the support. The scheme of OMP-QR is summarized in Algorithm 19. Another popular variant described in [74] is based on the Cholesky decomposition of the matrix $D_S D_S^T$. In a nutshell, when a new atom is added to the support the Cholesky decomposition of $D_S D_S^T$ is updated and the new solution x is computed by considering the normal equation associated to (5.6). A complete description of this *OMP Cholesky* can be found in [26].

5.1.2 Dictionary update

The dictionary update step consists in the (approximate) solution of the problem

$$\min_D \|Y - DX\|_F^2 \quad s.t. \quad D \in \Omega_{n,k}. \tag{5.7}$$

Additional structures can be imposed on the dictionary to enforce specific properties. For example, one can consider *sparse dictionaries* $D = D_1 D_2$ where D_1 is fixed and D_2 is sparse. In [51] the matrix D is composed by orthogonal blocks (*Union of Orthonormal Basis*). In [40] a separable structure on $D = D_1 \otimes D_2$ is imposed and instead of learning the matrix D one can work directly on smaller blocks D_1, D_2. Other structured dictionaries have been explored in the literature, but their analysis is beyond the scope of this thesis. However, some structured dictionaries related to tensor decompositions are reviewed in Section 5.2. A complete review can be found in [26, 23].

Method of Optimal Directions (MOD)

The Method of Optimal Directions (MOD), described in [29], is an alternating method characterized by a dictionary update step based on the exact solution of the least-squares problem (5.7). More precisely, MOD determines the dictionary as

$$D = YX^\dagger, \tag{5.8}$$

where $X^\dagger$ denotes the Moore-Penrose inverse of X, i.e. $X^\dagger = X^T \left(XX^T\right)^{-1}$. This dictionary update step is equivalent to set the partial gradient of the objective function equal to zero. In particular, since $\nabla_D \|Y - DX\|_F^2$ is given by

$$\nabla_D \left(\|Y - DX\|_F^2\right) = -2X^T(Y - DX), \tag{5.9}$$

setting (5.9) equal to zero gives (5.8). The complexity of this dictionary update is mainly due to the computation of the pseudoinverse of X. Note that considering the Cholesky decomposition of XX^T can reduce the computational complexity of the algorithm (see [26]).

K-SVD algorithm

The K-SVD algorithm, discussed in [1], is one of the most cited algorithms to solve Dictionary Learning problems. As MOD, it relies on an alternating optimization strategy. Furthermore, the matrix X is also updated during the minimization in D enhancing performance for practical purposes.

In the dictionary update step, each atom is updated separately using the Singular Value Decomposition (SVD) in the following manner. First of all, we rewrite the objective function in (5.4).

$$\begin{aligned} \|Y - DX\|_F^2 &= \left\| Y - \sum_{j=1}^{k} d_j x_j^T \right\|_F^2 = \left\| \left(Y - \sum_{j=1, j\neq l}^{k} d_j x_j^T \right) - d_l x_l^T \right\|_F^2 \\ &= \left\| E_l - d_l x_l^T \right\|_F^2, \end{aligned} \tag{5.10}$$

where the matrix E_l represents the approximation error when the l column of the dictionary D is removed. To update d_l in order to decrease the overall error, we could try to use the approximation properties of the SVD. More precisely, we could find a rank-1 approximation of E_l, computing its SVD and setting d_l as the first left singular vector and x_l as the first right singular vector multiplied by the corresponding singular value. However this violates the sparsity constraint on X, since usually the right singular vectors are not sparse [1]. To overcome this problem instead of considering (5.10), we consider the following

$$\left\| E_l P_l - d_l x_l^T P_l \right\|_F^2 = \left\| E_l^R - d_l (x_l^R)^T \right\|_F^2, \tag{5.11}$$

where P_l is a matrix that shrinks the vector x_l, by discarding the zero elements. Defining $(x_l^R)^T = x_l^T P_l$ and $E_l^R = E_l P_l$, we can use the SVD to solve (5.11). Considering the SVD of $E_l^R = U_l \Sigma_l V_l^T$, we set d_l equal to the first column of U_l and x_l^R equal to the first column of V_l multiplied by the first singular value. Notice that the constraint on the norm of the atoms of D is satisfied since the singular vectors have unit norm. The overall procedure is summarized in Algorithm 20.

In [1] the K-SVD is applied to image data to recover missing pixel and for compression tasks, while in [27] it is applied to image denoising and in [83] to face recognition. The algorithm shows great practical performance in all these applications. However, from a theoretical point of view, is not possible to determine whether the computed solution is a local minimum of our problem. This is due to the absence of strong convergence guarantees of the iterative algorithm. Convergence is guaranteed only when the sparse coding retrieves the best sparse approximation of the signals Y. This, together with the decrease of the objective function in each dictionary update step, guarantees a monotonic decrease of the objective function and therefore the convergence to a local minimum [1]. However, since this is quite uncommon in practical situations related to image processing, a stopping criterion based on the maximum number of iterations is commonly used.

Algorithm 20 K-SVD [1]

Input: data matrix $Y \in \mathbb{R}^{n \times p}$, initial matrix $D \in \mathbb{R}^{n \times k}$
for $it = 1, \dots,$ **do**
  Sparse Coding: Use any pursuit algorithm to compute $X^{(it)}$
  Dictionary Update:
  for $l = 1, \dots, k$ **do**
    Compute the error $E_l = Y - \sum_{j=1, j \neq l}^{k} d_j x_j^T$
    Restrict E_l by choosing the columns corresponding to the non zero elements of x_l, to obtain E_l^R
    Compute the SVD of $E_l^R = U_l \Sigma_l V_l^T$ and set d_l equal to the first column of U_l and x_l^R equal to the first column of V_l multiplied by the first singular value
  end for
end for
Output: D, X

5.2 Tensor methods

In this section we review some important tensor Dictionary Learning formulations and related algorithms. They can be seen as structured Dictionary Learning problems since they impose a specific structure on the dictionary [26, Chapter 7]. For example, the HO-SuKro algorithm imposes that the dictionary can be written as a sum of Kronecker products of smaller subdictionaries, while K-HOSVD imposes that the corresponding dictionary matrix can be written as a Kronecker product of two smaller subdictionaries.

5.2.1 K-HOSVD

The K-HOSVD is a tensor version of the K-SVD, where the HOSVD is used in place of the matrix SVD. In [66] the DL problem is formulated as follows

$$\min_{D_1, D_2, \mathcal{X}} \|\mathcal{Y} - \mathcal{X} \times_1 D_1 \times_2 D_2\|_F \quad \text{subject to } \|\mathcal{X}\|_0 \leq K_{max} \quad \text{and } D_i \in \Omega_{n_i, k_i} \;\; i = 1, 2, \tag{5.12}$$

where $\mathcal{Y} \in \mathbb{R}^{n_1 \times n_2 \times p}$, $D_1 \in \mathbb{R}^{n_1 \times k_1}$, $D_2 \in \mathbb{R}^{n_2 \times k_2}$ and $\mathcal{X} \in \mathbb{R}^{k_1 \times k_2 \times p}$. This is equivalent to assume that the dictionary D of the related matrix DL problem (5.4) has a separable structure, i.e. $D = D_1 \otimes D_2$. Indeed, by considering the reshape along the second mode of (5.12) we obtain

$$\min_{D_1, D_2, \mathcal{X}} \left\|\mathcal{Y}_{[2]} - (D_1 \otimes D_2)\, \mathcal{X}_{[2]}\right\|_F \quad \text{subject to } \|\mathcal{X}\|_0 \leq K_{max} \quad \text{and } D_i \in \Omega_{n_i, k_i} \;\; i = 1, 2. \tag{5.13}$$

The K-HOSVD algorithm alternates between sparse coding and dictionary update. Each atom of D_1 and D_2 is updated separately as in the matrix setting and the tensor

$\mathcal{X}$ is updated also during the dictionary update step. Following the K-SVD scheme we first rewrite (5.12) as

$$\min_{d_{j_1}^{(1)}, d_{j_2}^{(2)}, x_j} \left\| \mathcal{E}_{j_1,j_2} - d_{j_1}^{(1)} \circ d_{j_2}^{(2)} \circ x_j \right\|_F^2, \tag{5.14}$$

where

$$\mathcal{E}_{j_1,j_2} = \mathcal{Y} - \sum_{\substack{s_1=1 \\ (s_1,s_2)\neq(j_1,j_2)}}^{k_1} \sum_{s_2=1}^{k_2} d_{s_1}^{(1)} \circ d_{s_2}^{(2)} \circ x_s, \qquad s = (s_1 - 1)k_2 + s_2,$$

and $d_{s_i}^{(i)}$ denotes the s_ith column of D_i for $i = 1, 2$, while x_s denotes the sth fiber of tensor $\mathcal{X}$. Then we denote as x_j^R the vector obtained from x_j in (5.14) by omitting its non zero entries. This can be thought as a projection matrix acting on the rank-1 tensor $d_{j_1}^{(1)} \circ d_{j_2}^{(2)} \circ x_j$. The corresponding error tensor is denoted as $\mathcal{E}_{j_1,j_2}^R$. Thus (5.14) becomes

$$\min_{d_{j_1}^{(1)}, d_{j_2}^{(2)}, x_j^R} \left\| \mathcal{E}_{j_1,j_2}^R - d_{j_1}^{(1)} \circ d_{j_2}^{(2)} \circ x_j^R \right\|_F^2.$$

Considering the HOSVD of $E_{j_1,j_2}^R = \mathcal{S} \times_1 U_1 \times_2 U_2 \times_3 U_3$ we set $d_{j_1}^{(1)} = u_{11}$, $d_{j_2}^{(2)} = u_{21}$ and $x_j^R = s_{111} u_{31}$, where u_{11}, u_{21} and u_{31} denote the first columns of U_1, U_2, U_3 respectively and $s_{111} = \mathcal{S}(1, 1, 1)$. For the sparse coding step the equivalent formulation (5.13) is considered and Algorithm 18 is applied. The whole procedure is summarized in Algorithm 21.

It is worth to underline that even though the HOSVD does not have the property of best approximation of the SVD, the K-HOSVD algorithm has good practical behaviour as shown in [66].

5.2.2 HO-SuKro

Another interesting and recent tensor formulation of the DL problem is based on the representation of the dictionary as a sum of Kronecker products of smaller subdictionaries. This formulation was explored both in [21, 22] by Dantas, Cohen and Gibronval and in [34, 69] by Ghassemi, Shakeri, Bajwa and Sarwate. Furthermore in [34, 69] the authors provide sufficient conditions on several parameters of the DL problem, such as the number of samples, i.e. the number of columns of Y, that guarantee the local recovery of the structured dictionary with high probability.

As observed in [21], imposing a Kronecker structure on the dictionary matrix is equivalent to imposing a CP structure on the corresponding dictionary tensor. To determine the factor dictionaries, several algorithms have been developed either using the CP structure or using directly the Kronecker structure. The following discussion refers to [22]. Consider the data tensor $\mathcal{Y} \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times p}$ that can be thought as a collection of p third-order tensors, such as RGB images. Instead of solving the classical DL problem (5.4) for

Algorithm 21 K-HOSVD [66]

Input: data tensor $\mathcal{Y} \in \mathbb{R}^{n_1 \times n_2 \times p}$, initial matrices $D_1 \in \mathbb{R}^{n_1 \times k_1}$, $D_2 \in \mathbb{R}^{n_2 \times k_2}$
for $it = 1, \ldots,$ **do**
  Sparse Coding: Use any pursuit algorithm to compute the sparse solution of (5.13) $X_{[2]}$
  Dictionary Update:
  for $j_1 = 1, \ldots, k_1$ **do**
    for $j_2 = 1, \ldots, k_2$ **do**
      Compute the error $E_{j_1,j_2} = \mathcal{Y} - \sum_{\substack{s_1=1 \\ (s_1,s_2)\neq(j_1,j_2)}}^{k_1} \sum_{s_2=1}^{k_2} d_{s_1}^{(1)} \circ d_{s_2}^{(2)} \circ x_s$
      Restrict $\mathcal{E}_{j_1,j_2}$ by choosing the columns corresponding to the non zero elements of x_j^R, to obtain $\mathcal{E}_{j_1,j_2}^R$
      Compute the HOSVD of $\mathcal{E}_{j_1,j_2}^R = \mathcal{S} \times_1 U_1 \times_2 U_2 \times_3 U_3$
      Set
      $d_{j_1}^{(1)} \leftarrow u_{11}$
      $d_{j_2}^{(2)} \leftarrow u_{21}$
      $x_j^R \leftarrow s_{111} u_{31}$
    end for
  end for
end for
Output:$D_1 \in \mathbb{R}^{n_1 \times k_1}$, $D_2 \in \mathbb{R}^{n_2 \times k_2}$ and $\mathcal{X} \in \mathbb{R}^{k_1 \times k_2 \times p}$.

a general dictionary D, the HO-SuKro restricts to dictionaries that can be written as a High-Order Sum of Kronecker products, i.e.

$$D = \sum_{q=1}^{R} D_{1,q} \otimes D_{2,q} \otimes D_{3,q}, \tag{5.15}$$

where $D_{j,q} \in \mathbb{R}^{n_j \times k_j}$ for $j = 1, 2, 3$ with $k = k_1 k_2 k_3$. Thus the DL problem becomes

$$\min_{D_{j,q}, \mathcal{X}} \left\| \mathcal{Y}_{[3]} - \left(\sum_{q=1}^{R} D_{1,q} \otimes D_{2,q} \otimes D_{3,q} \right) \mathcal{X}_{[3]} \right\|_F^2 + g(\mathcal{X}) \tag{5.16}$$

where $\mathcal{X} \in \mathbb{R}^{k_1 \times k_2 \times k_3 \times p}$ is a sparse fourth-order tensor and g is a sparsity inducing function. For example g can be set equal to the zero norm of $\mathcal{X}$ which is the total number of nonzero elements of $\mathcal{X}$.

For the dictionary update step in the HO-SuKro algorithm we have to write the partial cost function for each block $\Delta_j = [D_{j,1}, \ldots D_{j,R}]$ for $j = 1, 2, 3$. Consider the case

Algorithm 22 HO-SuKro [22, Algorithm 1]

Input: data matrix $\mathcal{Y} \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times p}$, sparsity threshold T, initial matrices $\{D_{j,q}\}_{q=1,\dots,R}^{j=1,2,3}$
for $it = 1, \dots,$ **do**
  Sparse coding: Use any sparse coding algorithm to solve

$$\min_{\mathcal{X}} \left\| \mathcal{Y}_{[3]} - \left(\sum_{q=1}^{R} D_{1,q} \otimes D_{2,q} \otimes D_{3,q} \right) \mathcal{X}_{[3]} \right\|_F^2 + g(\mathcal{X})$$

  Dictionary update:
  while update $D_{j,q}$ is significant
  for $j = 1, 2, 3$ **do**
    Compute $\hat{\Delta}_j = \underset{\Delta_j}{\operatorname{argmin}} \|\mathcal{Y}_{(1)} - \Delta_j U\|_F^2$
  end for
end for
Output: Blocks $\{D_{j,q}\}_{q=1,\dots,R}^{j=1,2,3}$ and the sparse tensor $\mathcal{X}$

$j = 1$

$$f(D_{1,q}) = \left\| \mathcal{Y}_{(1)} - \sum_{q=1}^{R} D_{1,q} \mathcal{X}_{(1)} \left(D_{2,q} \otimes D_{3,q} \otimes I_n \right)^T \right\|_F^2 = \|\mathcal{Y}^{(1)} - \Delta_1 U\|_F^2, \tag{5.17}$$

where $U = [U_1; \dots; U_R]$ with $U_q = \mathcal{X}_{(1)} \left(D_{2,q} \otimes D_{3,q} \otimes I_n \right)^T$ for $q = 1, \dots, R$. The minimum of (5.17) is given in closed form by the following

$$\hat{\Delta}_1 = \underset{\Delta_1}{\operatorname{argmin}} \|\mathcal{Y}_{(1)} - \Delta_1 U\|_F^2 = \mathcal{Y}_{(1)} U^{\dagger}.$$

The same procedure is used to determine $\hat{\Delta}_2$ and $\hat{\Delta}_3$. For the sparse coding step (5.16) is considered and a sparse coding algorithm is used to determine $\mathcal{X}_{[3]}$ depending on the chosen penalty function g. For example, if $g(\mathcal{X}) = \|\mathcal{X}\|_0$, then the OMP algorithm is used. Note that in this step the computation of the whole D is avoided and the Kronecker structure is used to lower the computational cost. The overall procedure is summarized in Algorithm 22.

5.2.3 GRADTENSOR

In [87] the GRADTENSOR algorithm, based on sparse Tucker decomposition, is presented. In this setting, the problem has a slightly different formulation for the sparsity

Algorithm 23 TOMP [87, Algorithm 1]

Input: Dictionaries D_i, input signal $\mathcal{X}$, maximum number of non-zeros coefficients $t_{max} \leq s$, tolerance ϵ.
while $|M_1||M_2||M_3| < t_{max}$ and $\|\mathcal{R}\|_F > \epsilon$ **do**
  for $t = 1, \ldots$ **do**
    Select Atom:
    $[m_1^t, m_2^t, m_3^t] = argmax_{[m_1,m_2,m_3]} |\mathcal{R} \times_1 D_1(:, m_1) \times_2 D_2(:, m_2) \times_3 D_3(:, m_3)|$
    $M_n = M_n \cup m_n^t$ for $n = 1, 2, 3$
    Update the residual:
    $D_n^* = D_n(:, M_n)$ for $n = 1, 2, 3$
    $e = argmin_u \|(D_3^* \otimes D_2^* \otimes D_1^*)u - y\|_2^2$ and reshape e as a third-order tensor $\mathcal{E}$
    $\mathcal{R} = \mathcal{Y} - \mathcal{E} \times_1 D_1^* \times_2 D_2^* \times_3 D_3^*$
  end for
end while
$\mathcal{X}(M_1, M_2, M_3) = \mathcal{E}$
Output: tensor $\mathcal{X}$, (M_1, M_2, M_3).

constraint. More precisely, the DL problem is

$$\min_{\mathcal{X}, D_1, D_2, D_3} F(\mathcal{X}, D_1, D_2, D_3) \quad s.t. \quad x_{m_1,m_2,m_3} = 0 \quad \forall (m_1, m_2, m_3) \notin (M_1 \times M_2 \times M_3) \tag{5.18}$$

where

$$F(\mathcal{X}, D_1, D_2, D_3) = \|\mathcal{Y} - \mathcal{X} \times_1 D_1 \times_2 D_2 \times_3 D_3\|_F^2$$

and $\mathcal{Y} \in \mathbb{R}^{n_1 \times n_2 \times p}$, $\mathcal{X} \in \mathbb{R}^{k_1 \times k_2 \times k_3}$, $D_1 \in \mathbb{R}^{n_1 \times k_1}$, $D_2 \in \mathbb{R}^{n_2 \times k_2}$ and $D_3 \in \mathbb{R}^{p \times k_3}$. The total sparsity (i.e. number of the non zero elements) of $\mathcal{X}$ is denoted by $s = s_1 s_2 s_3$. The GRADTENSOR algorithm computes the numerical solution of (5.18) by alternating between the dictionary update and the sparse coding step. The latter is performed using a tensor version of the OMP, called TOMP (Tensor-OMP) and described in Algorithm 23.

In the dictionary update step each dictionary D_i is computed iteratively by gradient descent in an alternating manner until the difference between two consecutive iterations is smaller than a given threshold. To this end we rewrite (5.18) using the unfoldings along different modes as follows.

$$F_{D_1} = \left\|Y_{(1)} - D_1 X_{(1)} (D_3 \otimes D_2)^T\right\|_F^2, \tag{5.19}$$

$$F_{D_2} = \left\|Y_{(2)} - D_2 X_{(2)} (D_2 \otimes D_1)^T\right\|_F^2, \tag{5.20}$$

$$F_{D_3} = \left\|Y_{(3)} - D_3 X_{(3)} (D_2 \otimes D_1)^T\right\|_F^2, \tag{5.21}$$

Algorithm 24 GRADTENSOR [87, Algorithm 2]

Input: data tensor $\mathcal{Y}$, sparse core tensor $\mathcal{X}$, maximum sparsity value $s = s_1 s_2 s_3$, step size γ, tolerance ϵ, ϵ_1, ϵ_2, ϵ_3.
The matrices D_n are initialized by left singular vectors of $Y_{(n)}$ for $n = 1, 2, 3$
for $i = 1, \dots$ **do**
  Sparse coding: Use TOMP to solve $\min_{\mathcal{X}} F(\mathcal{X}, D_1, D_2, D_3)$ *s.t.* $x_{m_1,m_2,m_3} = 0 \quad \forall (m_1, m_2, m_3) \notin (M_1 \times M_2 \times M_3)$
  Dictionary update:
  for $n = 1, 2, 3$ **do**
    for $it = 0, \dots$ **do**
      $D_n^{(it+1)} = D_n^{(it)} - \gamma \nabla F_{D_n}$
      if $\|D_n^{(it+1)} - D_n^{(it)}\|_F \leq \epsilon_n$ stop
    end for
    if $F(\mathcal{X}, D_1, D_2, D_3) \leq \epsilon$ stop
  end for
end for
Output: $\mathcal{X}$ and D_n for $n = 1, 2, 3$

Thus the updates for the dictionaries are given by

$$D_n^{(it+1)} = D_n^{(it)} - \gamma \nabla F_{D_n}, \qquad n = 1, 2, 3,$$

where it is the current step of the gradient descent algorithm. The whole procedure is summarized in Algorithm 24. Though in [87], the authors do not provide any convergence result, the algorithm shows good experimental behaviour.

5.2.4 K-CPD

The K-CPD algorithm, presented in [25], follows the scheme of K-SVD and K-HOSVD and updates each atom of the dictionary separately using the CPD. Given a tensor $\mathcal{Y}^\Omega \in \mathbb{R}^{n_1 \times \cdots \times n_N \times p}$ the K-CPD aims to solve the following optimization problem.

$$\min_{D^\Omega, X} \left\| \mathcal{Y}^\Omega - \mathcal{D}^\Omega \times_{N+1} X^T \right\|_F^2 \quad \text{s.t.} \, \|x_i\|_0 \leq T \quad \forall i = 1, \dots, K, \tag{5.22}$$

where $\mathcal{D}^\Omega \in \mathbb{R}^{k_1 \times \cdots \times k_N \times K}$ is the dictionary and each atom $\mathcal{D}_k = \mathcal{D}^\Omega(:, \dots, :, k)$ is an Nth-order rank-1 tensor $\mathcal{D}_k = d_k^{(1)} \circ \cdots \circ d_k^{(N)}$ and x_i denotes the i-th column of the matrix X. The solution of (5.22) is computed using an alternating procedure, as for most DL algorithms. The dictionary update step is performed as follows. For each atom k the error is given by

$$\mathcal{E}_k^{(R)} = \mathcal{Y}^\Omega - \sum_{j \neq k} \mathcal{D}_j \times_{N+1} (x_j^{(R)})^T,$$

Algorithm 25 K-CPD [25, Algorithm 2]

Input: data tensor $\mathcal{Y}$, threshold of sparsity $\tau > 0$, initial value for $\mathcal{D}^{\Omega}$.
for $it = 1, \ldots,$ **do**
  for $k = 1, \ldots, K$ **do**
    Compute $\mathcal{E}_k^{(R)} = \mathcal{Y}^{\Omega} - \sum_{j \neq k} \mathcal{D}_j \times_{N+1} (x_j^{(R)})^T$
    Compute the rank-1 CP approximation of the error tensor

$$\mathcal{E}_k^{(R)} = \lambda_k a_k^{(1)} \circ \cdots \circ a_k^{(N+1)}$$

    Set $d_k^{(n)} = a_k^{(n)}$ for $n = 1, \ldots, N$ and $x_k^{(R)} = \lambda_k \left(a_k^{(N+1)} \right)^T$
  end for
end for
Output: X and $\mathcal{D}^{\Omega}$.

where x_j^R is obtained from x_j by taking only the non zero elements as in K-SVD or K-HOSVD. Then a rank-1 approximation of $\mathcal{E}_k^{(R)}$ is determined using the CP decomposition

$$\mathcal{E}_k^{(R)} = \lambda_k a_k^{(1)} \circ \cdots \circ a_k^{(N+1)}. \tag{5.23}$$

Then, assuming that each $\mathcal{D}_k$ has a rank-1 structure $\mathcal{D}_k = d_k^{(1)} \circ \cdots \circ d_k^{(N)}$, each $d_k^{(n)}$ is set equal to $a_k^{(n)}$ for $n = 1, \ldots, N$. As in K-SVD and K-HOSVD also $x_k^{(R)}$ is updated as $x_k^{(R)} = \lambda_k \left(a_k^{(N+1)} \right)^T$. The whole procedure is summarized in Algorithm 25. The experiments performed show that the K-CPD slightly outperforms K-SVD. The sparse coding step is performed using a variant of OMP, named MOMP (Multilinear OMP), that takes into account the structure of the dictionary $\mathcal{D}^{\Omega}$. For further details see [25, Algorithm 1].

Another algorithm based on the CPD is described in [43] where the objective function is formulated as follows.

$$\min_{\mathcal{X}, D_1, D_2} \|\mathcal{Y} - \mathcal{X} \times_1 D_1 \times_2 D_2\|_F^2 + \lambda \|\mathcal{X}\|_0 ,$$

where $\mathcal{X} \in \mathbb{R}^{k_1 \times k_1 \times p}$, $D_1, D_2 \in \mathbb{R}^{\sqrt{n} \times k_1}$. The minimization is performed in an alternating manner and the sparse coding step is performed using an appropriate version of the OMP (2D OMP) described in [30]. Using the outer product and, neglecting the penalty term $\lambda \|\mathcal{X}\|_0$, $\mathcal{Y}$ can be formulated as

$$\begin{aligned} \mathcal{Y} &= \left(d_j^1 \circ d_k^2 \circ x_{jk} \right) + \sum_{a \neq j}^{n} \left(d_j^1 \circ d_a^2 \circ x_{ja} \right) + \sum_{b \neq k}^{n} \left(d_b^1 \circ d_k^2 \circ x_{bk} \right) + \sum_{a \neq j}^{n} \sum_{b \neq k}^{n} \left(d_a^1 \circ d_b^2 \circ x_{ab} \right) \\ &= \mathcal{E}^{(j,k)} + \left(d_j^1 \circ d_k^2 \circ x_{jk} \right) + \sum_{a \neq j}^{n} \left(d_j^1 \circ d_a^2 \circ x_{ja} \right) + \sum_{b \neq k}^{n} \left(d_b^1 \circ d_k^2 \circ x_{bk} \right). \end{aligned}$$

Since d_j^1 and d_k^2 cannot be determined by directly performing CP decomposition on $\mathcal{E}^{(j,k)}$, a Regularized Alternating Least Square (RALS) is used. The procedure is described in the APPENDIX of [43]. The performance of the algorithms are comparable or slightly superior to the K-SVD (see [25, 43]).

Chapter 6

Proximal methods for Dictionary Learning

In this chapter we analyze the application of PALM algorithms presented in Chapters 3 and 4 to the DL problem (5.4) in the matrix setting, as in [2, 86, 52]. Furthermore we propose a new Tensor-Train formulation of the DL problem and we show how PALM-type algorithms (including sPALM) can be naturally applied in order to get convergent schemes. To the best of our knowledge, these are the first tensor-based algorithms in the DL literature with global convergence guarantees.

6.1 Proximal matrix methods

In this section we discuss the convergence of PALM-type algorithms described in Chapter 3 and 4 in the matrix Dictionary Learning framework. PALM, btPALM (i.e. PALM with backtracking strategy for the stepsize) and sPALM when applied to the DL problem will be denoted as PALM-DL , btPALM-DL and sPALM-DL respectively.

In order to use proximal methods, we need to reformulate the matrix DL problem (5.4) by including the constraints in the objective function as follows.

$$\min_{D,X} \|Y - DX\|_F^2 + \delta_{\Omega_{n,k}}(D) + \delta_{\Gamma_{k,p}^{(\tau)}}(X), \tag{6.1}$$

where $\delta_{\Omega_{n,k}}$and $\delta_{\Gamma_{k,p}^{(\tau)}}$ are indicator functions over the sets $\Omega_{n,k}$ and $\Gamma_{k,p}^{(\tau)}$ defined in (5.2) and (5.5), respectively. Given a non-empty and closed set $\Omega \subset \mathbb{R}^{m\times n}$, we recall that the indicator function $\delta_\Omega : \mathbb{R}^{m\times n} \to (-\infty, +\infty]$ is given by

$$\delta_\Omega(A) = \begin{cases} 0 & \text{if } A \in \Omega \\ +\infty & \text{otherwise.} \end{cases} \tag{6.2}$$

The formulation (6.1) is equivalent to (5.4), because when $D \in \Omega_{n,k}$ and $X \in \Gamma_{k,p}^{(\tau)}$ the indicator functions are equal to zero. The formulation (6.1) has clearly the form of the

general problem (3.12) with $H(D,X) = \|Y - DX\|_F^2$, $f_1(D) = \delta_{\Omega_{n,k}}(D)$ and $f_2(X) = \delta_{\Gamma_{k,p}^{(\tau)}}(X)$. The following propositions prove that these functions are semi-algebraic and satisfy Assumption 3.3.1.

Proposition 6.1.1. *The function* $\Psi(D,X) = \|Y - DX\|_F^2 + \delta_{\Omega_{n,k}}(D) + \delta_{\Gamma_{k,p}^{(\tau)}}(X)$ *is semi-algebraic.*

Proof. The function $H(D,X) = \|Y - DX\|_F^2$ is a real polynomial function and thus semi-algebraic. The terms $\delta_{\Omega_{n,k}}$, $\delta_{\Gamma_{k,p}^{(\tau)}}$ are indicator functions of semi-algebraic sets and thus are semi-algebraic. To conclude we just observe that a finite sum of semi-algebraic functions is semi-algebraic as well [9]. □

Proposition 6.1.2. *The partial gradients* $\nabla_X H(D,X)$ *and* $\nabla_D H(D,X)$ *where* $H(D,X) = \|Y - DX\|_F^2$ *are globally Lipschitz continuous with moduli*

$$L_X = 2\|D^T D\|_2 \quad \text{and} \quad L_D = 2\|XX^T\|_2, \tag{6.3}$$

respectively.

Proof. By direct computation the following expressions for the partial gradients of H hold:

$$\nabla_D H(D,X) = -2(Y - DX)X^T,$$
$$\nabla_X H(D,X) = -2D^T(Y - DX).$$

Thus, for any $\hat{D}$,$\tilde{D} \in \mathbb{R}^{n\times k}$ we have

$$\left\|\nabla_D H(\hat{D},X) - \nabla_D H(\tilde{D},X)\right\|_2 = \left\|2\left(\hat{D} - \tilde{D}\right)XX^T\right\|_2 \leq 2\left\|XX^T\right\|_2 \left\|\hat{D} - \tilde{D}\right\|_2,$$

where $L_D = 2\|XX^T\|_2$ is the Lipschitz constant of $\nabla_D H(D,X)$. The same procedure can be repeated to compute L_X. □

Proposition 6.1.3. *The gradient* $\nabla H(D,X)$ *of* $H(D,X) = \|Y - DX\|_F^2$ *is Lipschitz continuous on bounded subsets of* $\mathbb{R}^{n\times k} \times \mathbb{R}^{k\times p}$.

Proof. The result follows directly from the smoothness of $H(D,X)$, as stated in Remark 3.3.2. □

To conclude the convergence of PALM-type algorithms for the Dictionary Learning problem we recall that in the convergence analysis the sequence generated by the proximal algorithms described in Chapter 3 and 4 is assumed to be bounded (Theorems 3.3.1 and 4.2.1). Assuming that the sequence $\{D_k, X_k\}$ is bounded ensures that L_D and L_X are bounded from above. Furthermore as suggested in [9] instead of using L_D and L_X, we can consider the following partial smoothness parameters

$$L'_X = \max\left\{\mu_X,\ 2\|D^T D\|_2\right\} \quad \text{and} \quad L'_D = \max\left\{\mu_D,\ 2\|XX^T\|_2\right\}.$$

where $\mu_X,\ \mu_D > 0$ so that Assumption 3.3.1 is satisfied since L'_D and L'_X are safely bounded away from zero.

We recall that the proximal map of an indicator function δ_Ω over a non-empty and closed set $\Omega \subset \mathbb{R}^{m\times n}$ is the multi-valued projection $P_\Omega : \mathbb{R}^{m\times n} \rightrightarrows \Omega$ such that

$$P_\Omega(A) = \operatorname*{argmin}_{B\in\Omega} \|A - B\|_F. \tag{6.4}$$

The following proposition gives a closed form expression for the corresponding projection operators over the sets $\Omega_{m,n}$ and $\Gamma^{(\tau)}_{m,n}$.

Proposition 6.1.4. *Let $A \in \mathbb{R}^{m\times n}$.*

i. Let $\Omega_{m,n} \subset \mathbb{R}^{m\times n}$ be defined in (5.2). Then $P_{\Omega_{m,n}}(A) = AS^{-1}$, where $S \in \mathbb{R}^{n\times n}$ is a diagonal matrix whose diagonal elements are the norm of the columns of A.

ii. Let $\Gamma_{m,n} \subset \mathbb{R}^{m\times n}$ be defined in (5.5). Then $P_{\Gamma_{m,n}}(A) = \texttt{hard}_\tau(A)$, where $\texttt{hard}_\tau$ is the hard-thresholding function that selects the τ largest elements (in absolute value) of each column of A and zeroes all the others.

Proof. For $B \in \Omega \subset \mathbb{R}^{m\times n}$ having unit norm columns, we have that $\|A - B\|_F^2 = \operatorname{trace}(A^TA) - 2\operatorname{trace}(B^TA) + n$ and thus solving (6.4) is equivalent to finding $B = \operatorname{argmax}_\Omega \operatorname{trace}(A^TB)$.

i) For $\Omega = \Omega_{m,n}$ and $A = [a_1, \dots, a_n]$, $B = [b_1, \dots, b_n]$ with $\|b_j\| = 1$, $j = 1, \dots, n$, it holds that

$$\operatorname{trace}(B^TA) = \sum_{j=1}^n b_j^T a_j = \sum_{j=1}^n \cos(\theta_j)\|a_j\|_2 \le \sum_{j=1}^n \|a_j\|_2,$$

where $\theta_j \in \left[0, \frac{\pi}{2}\right]$, and the upper bound is reached for $b_j = a_j/\|a_j\|_2$.

ii) From [9, Section 4], we have that the projection $P_{\Gamma^{(\tau)}_{m,n}}(A)$ of A onto the set of sparse matrices $\Gamma^{(\tau)}_{m,n}$ is equivalent to select the τ largest elements in absolute value from each column and set to zero all the others. □

By applying Theorem 3.3.1 and Theorem 4.2.1 we can conclude that PALM-type algorithms, including sPALM, are guaranteed to converge to a critical point of problem (6.1).

We report in Algorithm 26 the PALM scheme for the DL problem (6.1) which is similar to the one in [2]. At each iteration, a gradient step is performed using the smoothness parameters L'_D, L'_X and then the projection of the variables D, X onto the proper feasible sets is performed.

In several applications the dimensions of D and X can be quite large and the computation of the Euclidean norm can be quite demanding. Furthermore the Lipschitz constants sometimes provide small stepsizes and slow down the convergence of PALM. Thus, instead of considering constant stepsizes, $\alpha_{j,D}$ and $\alpha_{j,X}$ can be estimated using

Algorithm 26 PALM-DL

1: **Input**: $(D_0, X_0) \in \mathbb{R}^{n\times k} \times \mathbb{R}^{k\times p}$, $\eta_1, \eta_2 > 1$, $\mu_D, \mu_X > 0$
2: **for** $j = 0, 1, \ldots,$ **do**
3: *Update* D: Set $L'_D(X_j) = \max\{\mu_D,\ L_D(X_j)\}$, $\bar{\alpha}_{j,D} = 1/(\eta_D L'_D(X_j))$ and compute

$$D_{j+1} = P_{\Omega_{n,k}}\left(D_j - \bar{\alpha}_{j,1}\nabla_D H(D_j, X_j)\right) \tag{6.5}$$

4: *Update* X: Set $L'_X(D_{j+1}) = \max\{\mu_X,\ L_X(D_{j+1})\}$, $\bar{\alpha}_{j,X} = 1/(\eta_X L'_X(D_{j+1}))$ and compute

$$X_{j+1} = P_{\Gamma^{(\tau)}_{k,p}}\left(X_k - \bar{\alpha}_{j,X}\nabla_X H(D_{j+1}, X_j)\right) \tag{6.6}$$

5: **end for**

a backtracking rule. Indeed, setting $L_{0,D} = 1$, at each iteration $j > 1$ starting from $L_{j,D} = L_{j-1,D}$, $L_{j,D}$ is increased by a constant factor until the following condition is met

$$\Psi_1(D_{j+1}, X_j) < \Psi_1(D_j, X_j) + \langle \nabla_D H(D_j, X_j), D_{j+1} - D_j\rangle + \frac{L_{j,D}}{2}\|D_{j+1} - D_j\|_F^2,$$

where $\Psi_1(D, X) = H(D, X) + \delta_{\Omega_{n,k}}(D)$. Then, the stepsize $\bar{\alpha}_{j,D}$ is taken as the reciprocal of the sought value. The same holds also for the variable X. We start from $L_{0,X} = 1$ and at each iteration the value is increased until

$$\Psi_2(D_{j+1}, X_{j+1}) < \Psi_2(D_{j+1}, X_j) + \langle \nabla_X H(D_{j+1}, X_j), X_{j+1} - X_j\rangle + \frac{L_{j,X}}{2}\|X_{j+1} - X_j\|_F^2,$$

where $\Psi_2(D, X) = H(D, X) + \delta_{\Gamma^{(\tau)}_{k,p}}(X)$.

The above sufficient decrease conditions are motivated by the *descent lemma* reported in Lemma 3.3.1. The btPALM-DL procedure is summarized in Algorithm 27.

Since gradient methods based on Lipschitz moduli can yield slow convergence, we propose to use sPALM-DL, the PALM based algorithm with spectral steplength presented in Chapter 4.

As previously discussed, in the sPALM framework, the idea is to approximate at each step the second order derivatives $\nabla_{DD}H(D, X)$ and $\nabla_{XX}H(D, X)$ by a multiple of the identity matrix I, i.e.

$$\nabla_{DD}H(D, X) \approx \alpha_D^{-1} I \qquad \nabla_{XX}H(D, X) \approx \alpha_X^{-1} I,$$

for some α_D, $\alpha_X > 0$. Consider the case of the partial Hessian $\nabla_{DD}H(D, X)$. For each iteration j, let $S_{j,D} = D_{j+1} - D_j$ and $Z_{j,D} = \nabla_D H(D_{j+1}, X_j) - \nabla_D H(D_j, X_j)$, then

$$\alpha^{BB1}_{j+1,D} = \operatorname*{argmin}_{\alpha} \|\alpha^{-1} S_{j,D} - Z_{j,D}\|_F \quad \text{and} \quad \alpha^{BB2}_{j+1,D} = \operatorname*{argmin}_{\alpha} \|S_{j,D} - \alpha Z_{j,D}\|_F$$

Algorithm 27 btPALM-DL

1: **Input**: $(D_0, X_0) \in \mathbb{R}^{n\times k} \times \mathbb{R}^{k\times p}$, ρ_1, $\rho_2 > 1$.
2: Set $L_{0,D} = 1$, $L_{0,X} = 1$
3: **for** $j = 0, 1, \ldots,$ **do**
4: *Update* D: Compute

$$D_{j+1} = P_{\Omega_{n,k}}\left(D_j - \frac{1}{L_{j,D}}\nabla_D H(D_j, X_j)\right) \tag{6.7}$$

where $L_{j,D} = \rho_1^{i_j} L_{j-1,D}$ and i_j is the smallest nonnegative integer for which the condition

$$\Psi_1(D_{j+1}, X_j) < \Psi_1(D_j, X_j) + \langle \nabla_D H(D_j, X_j), D_{j+1} - D_j\rangle + \frac{L_{j,D}}{2}\|D_{j+1} - D_j\|_F^2$$

is satisfied.
5: *Update* X: Compute

$$X_{j+1} = P_{\Gamma_{k,p}^{(\tau)}}\left(X_j - \frac{1}{L_{j,X}}\nabla_X H(D_{j+1}, X_j)\right) \tag{6.8}$$

where $L_{j,X} = \rho_2^{i_j} L_{j-1,X}$ and i_j is the smallest nonnegative integer for which the condition

$$\Psi_2(D_{j+1}, X_{j+1}) < \Psi_2(D_{j+1}, X_j) + \langle \nabla_X H(D_{j+1}, X_j), X_{j+1} - X_j\rangle + \frac{L_{j,X}}{2}\|X_{j+1} - X_j\|_F^2$$

is satisfied.
6: **end for**

that is

$$\alpha_{j+1,D}^{BB1} = \frac{\langle S_{j,D}, S_{j,D}\rangle}{\langle S_{j,D}, Z_{j,D}\rangle} \quad \text{and} \quad \alpha_{j+1,D}^{BB2} = \frac{\langle S_{j,D}, Z_{j,D}\rangle}{\langle Z_{j,D}, Z_{j,D}\rangle}. \tag{6.9}$$

The same holds for the variable X. Among the several rules that can be used to combine $\alpha_{j,D}^{BB1}$ and $\alpha_{j,D}^{BB2}$ it was observed that one of the best choices in this DL framework is to alternate the two stepsizes. We report in Algorithm 28 an alternating rule described in Chapter 4 that seems to give the best numerical performance in this DL setting.

The sPALM algorithm is reported in Algorithm 29. The spectral stepsizes $\alpha_{j,D}$ and $\alpha_{j,X}$ are then decreased by a constant factors ρ_1, ρ_2 until the conditions (6.11) and (6.13) are satisfied, respectively. The following proposition shows that using second order information in the stepsizes may accelerate convergence since it results in longer steps than with the reciprocal of the Lipschitz constants.

Proposition 6.1.5. *Let $S_j = X_{j+1} - X_j$ and $T_j = D_{j+1} - D_j$ be computed at the jth*

Algorithm 28 Computation of the spectral stepsize $\alpha_{j+1,M}$, $M = D$ or $M = X$

Input: $S_{j,M}, Z_{j,M}$, $0 < \alpha_{\min} < \alpha_{\max}$.
if $\langle S_{j,M}, Z_{j,M}\rangle > 0$ **then**
  if k is odd **then**
    $\alpha_{j+1,M} = \max\left\{\alpha_{\min}, \min\left\{\alpha^{BB1}_{j+1,M}, \alpha_{\max}\right\}\right\}$ with $\alpha^{BB1}_{j+1,i} = \frac{\langle S_{j,M}, S_{j,M}\rangle}{\langle S_{j,M}, Z_{j,M}\rangle}$
  else
    $\alpha_{j+1,M} = \max\left\{\alpha_{\min}, \min\left\{\alpha^{BB2}_{j+1,M}, \alpha_{\max}\right\}\right\}$ with $\alpha^{BB2}_{j+1,M} = \frac{\langle S_{j,M}, Z_{j,M}\rangle}{\langle Z_{j,M}, Z_{j,M}\rangle}$
  end if
else
  $\alpha_{j+1,M} = 1$
end if

iteration of sPALM applied to problem (6.1). Assume that $\langle S_j, (D_j^T D_j)S_j\rangle \neq 0$ *and* $\langle T_j, T_j(X_{j+1}X_{j+1}^T)\rangle \neq 0$. *Then the BB stepsizes take the form*

$$\alpha^{BB1}_{X_{j+1}} = \frac{1}{2}\frac{\langle S_j, S_j\rangle}{\langle S_j, (D_j^T D_j)S_j\rangle} \quad \text{and} \quad \alpha^{BB2}_{X_{j+1}} = \frac{1}{2}\frac{\langle S_j, (D_j^T D_j)S_j\rangle}{\langle (D_j^T D_j)S_j, (D_j^T D_j)S_j\rangle},$$

$$\alpha^{BB1}_{D_{j+1}} = \frac{1}{2}\frac{\langle T_j, T_j\rangle}{\langle T_j, T_j(X_{j+1}X_{j+1}^T)\rangle} \quad \text{and} \quad \alpha^{BB2}_{D_{j+1}} = \frac{1}{2}\frac{\langle T_j, T_j(X_{j+1}X_{j+1}^T)\rangle}{\langle T_j(X_{j+1}X_{j+1}^T), T_j(X_{j+1}X_{j+1}^T)\rangle},$$

and the following bounds hold

$$\frac{1}{L_{X_j}} < \alpha^{BB2}_{X_{j+1}} < \min\left\{\alpha^{BB1}_{X_{j+1}}, \frac{1}{2\sigma^2_{\min}(D_j)}\right\}$$

and

$$\frac{1}{L_{D_{j+1}}} < \alpha^{BB2}_{D_{j+1}} < \min\left\{\alpha^{BB1}_{D_{j+1}}, \frac{1}{2\sigma^2_{\min}(X_{j+1})}\right\} \tag{6.14}$$

where $\sigma_{\min}(D_j)$ *and* $\sigma_{\min}(X_{j+1})$ *are the smallest nonzero singular values of* D_j *and* X_{j+1}, *respectively, and* L_{D_j} *and* $L_{X_{j+1}}$ *are given in (6.3).*

Proof. The stepsizes form derives from their definitions in (6.9) and from observing that $\nabla_X H = -2D^T(Y - DX)$ and $\nabla_D H = -2(Y - DX)X^T$. Moreover, the partial Hessians of H have the form $\nabla_{XX}H = I \otimes 2(D^T D)$, $\nabla_{DD}H = 2(XX^T) \otimes I$ and are positive semidefinite. Therefore the 2-norm Lipschitz constants are $L_X = \lambda_{\max}(\nabla_{XX}H)$ and $L_D = \lambda_{\max}(\nabla_{DD}H)$.

Let us consider the stepsizes $\alpha^{BB1}_{X_{j+1}}$ and $\alpha^{BB2}_{X_{j+1}}$. We observe that they are the reciprocal of Rayleigh quotients for $2(D_j^T D_j)$ and $2(D_j D_j^T)$ and S_j and $D_j S_j$, respectively, as

Algorithm 29 sPALM-DL

1: **Input**: $(D_0, X_0) \in \mathbb{R}^{n\times k} \times \mathbb{R}^{k\times p}$, $\rho_1, \delta_1, \rho_2, \delta_2 \in (0,1)$, $0 < \alpha_{min} < \alpha_{max}$, $\alpha_{0,1}, \alpha_{0,2} \in [\alpha_{min}, \alpha_{max}]$.
2: **for** $j = 0, 1, \ldots,$ **do**
3: *Update* x: Set

$$D_{j+1} = prox^{f_1}_{1/\bar{\alpha}_{j,D}}\left(D_j - \bar{\alpha}_{j,D}\nabla_D H(D_j, X_j)\right) \tag{6.10}$$

where $\bar{\alpha}_{j,D} = \rho_1^{i_j}\alpha_{j,D}$ and i_j is the smallest nonnegative integer for which the condition

$$\Psi_1(D_{j+1}, X_j) < \Psi_1(D_j, X_j) - \frac{\delta_1}{2\bar{\alpha}_{j,D}}\|D_{j+1} - D_j\|_F^2 \tag{6.11}$$

is satisfied.
4: Compute $\alpha_{j+1,D} \in [\alpha_{min}, \alpha_{max}]$ using Algorithm 28 with $S_{j,D} = D_{j+1} - D_j$ and $Z_{j,D} = \nabla_D H(D_{j+1}, X_j) - \nabla_D H(D_j, X_j)$.
5: *Update* y: Set

$$X_{j+1} = prox^{f_2}_{1/\bar{\alpha}_{j,X}}\left(X_j - \bar{\alpha}_{j,X}\nabla_X H(D_{j+1}, X_j)\right) \tag{6.12}$$

where $\bar{\alpha}_{j,X} = \rho_2^{l_j}\alpha_{j,X}$ and l_j is the smallest nonnegative integer for which the condition

$$\Psi_2(D_{j+1}, X_{j+1}) < \Psi_2(D_{j+1}, X_j) - \frac{\delta_2}{2\bar{\alpha}_{j,X}}\|X_{j+1} - X_j\|_2^2 \tag{6.13}$$

is satisfied.
6: Compute $\alpha_{j+1,X} \in [\alpha_{min}, \alpha_{max}]$ using Algorithm 28 with $S_{j,X} = X_{j+1} - X_j$ and $Z_{j,X} = \nabla_X H(D_{j+1}, X_{j+1}) - \nabla_X H(D_{j+1}, X_j)$.
7: **end for**

$\alpha^{BB2}_{X_{j+1}} = \frac{1}{2}\|D_jS_j\|_F^2/\langle D_jS_j, (D_jD_j^T)(D_jS_j)\rangle$. Moreover,

$$\begin{aligned}\alpha^{BB2}_{X_{j+1}} &= \frac{1}{2}\frac{\langle S_j, S_j\rangle}{\langle S_j, (D_j^TD_j)S_j\rangle}\frac{\langle S_j, (D_j^TD_j)S_j\rangle^2}{\langle (D_j^TD_j)S_j, (D_j^TD_j)S_j\rangle\langle S_j, S_j\rangle} \\ &= \alpha^{BB1}_{X_{j+1}}\frac{\|S_j\|_F^2\|(D_j^TD_j)S_j\|_F^2\cos^2\phi_j}{\|S_j\|_F^2\|(D_j^TD_j)S_j\|_F^2} = \alpha^{BB1}_{X_{j+1}}\cos^2\phi_j\end{aligned}$$

where ϕ_j is the angle between the vectorization $\hat{s}_j$ of S_j and $(I \otimes (D_j^TD_j))\hat{s}_j$. Therefore,

$$\alpha^{BB1}_{X_{j+1}} > \alpha^{BB2}_{X_{j+1}} > \frac{1}{\lambda_{\max}(2(D_j^TD_j))} = \frac{1}{L_{X_j}}.$$

Finally, let $\mathtt{D}_j = D_j^TD_j > 0$. Then $\alpha^{BB2}_{X_{j+1}} = \frac{1}{2}\|\mathtt{D}_j^{\frac{1}{2}}S_j\|_F^2/\|(\mathtt{D}_j^{\frac{1}{2}})\mathtt{D}_j^{\frac{1}{2}}S_j\|_F^2 < 1/(2\sigma^2_{\min}(D_j))$, where the last inequality follows from the fact that $\mathtt{D}_j^{\frac{1}{2}}S_j$ belongs to the range of $\mathtt{D}_j$.

The inequalities for $\alpha_{D_{j+1}}^{BB1}$ and $\alpha_{D_{j+1}}^{BB2}$ in (6.14) can be derived analogously. □

6.2 Tensor proximal methods for a new Tensor-Train formulation of the DL problem

To preserve the multidimensional structure of the data we propose a more general tensor form of the DL problem. More precisely, consider for instance a fourth-dimensional array $\mathcal{Y} \in \mathbb{R}^{n_1 \times n_2 \times n_e \times n_p}$ consisting of $n_1 \times n_2$ images of n_p persons in n_e expressions. Using the $\binom{m}{n}$-mode product and the constraint sets defined in (5.2) and (5.5), the tensor DL problem can be formulated as

$$\min_{\mathcal{D},\mathcal{X}} \|\mathcal{Y} - \mathcal{D} \times_3^1 \mathcal{X}\|_F^2 \quad s.t. \quad \mathcal{D}_{[2]} \in \Omega_{n_1 n_2, k}, \quad \mathcal{X}_{[1]} \in \Gamma_{k, n_e n_p}^{(\tau)}, \tag{6.15}$$

where $\mathcal{D} \in \mathbb{R}^{n_1 \times n_2 \times k}$ is a third-order tensor with unit norm frontal slices, and $\mathcal{X} \in \mathbb{R}^{k \times n_e \times n_p}$ is a sparse tensor with at most τ nonzero elements per column fiber, and $k > n_1 n_2$. Notice that the constraint in (5.2) on the columns of D results here in a constraint on the slices of the tensor $\mathcal{D}$ or on the columns of its matricization $\mathcal{D}_{[2]}$. Analogously, the sparsity constraint in the tensor formulation reduces to a constraint either on the maximum number of nonzero elements per column fiber of $\mathcal{X}$ or, equivalently, on the maximum number of nonzero elements per column of the unfolding of $\mathcal{X}$, $\mathcal{X}_{[1]}$. Hence, all the constraints proper of the matrix setting can be reformulated on the tensors themselves or on their matricizations. The formulation (6.15) is equivalent to (5.4) since, using Proposition 1.1.2, $\|\mathcal{Y} - \mathcal{D} \times_3^1 \mathcal{X}\|_F = \| \left(\mathcal{Y} - \mathcal{D} \times_3^1 \mathcal{X}\right)_{[2]} \|_F = \|\mathcal{Y}_{[2]} - \mathcal{D}_{[2]} \mathcal{X}_{[1]}\|_F$. To reduce memory requirements, instead of considering the whole tensor $\mathcal{D}$, we can consider its Tensor-Train Decomposition. We recall that such decomposition requires that for an Nth-order tensor, the reshapes along the second mode of the first $N-1$ TT-cores have orthonormal columns. For example, for a third-order tensor the columns of G_1 and of $(\mathcal{G}_2)_{[2]}$ are orthonormal. Therefore, to properly define the Tensor-Train formulation of the DL problem the additional constraint set of matrices with orthonormal columns is employed,

$$\Theta_{m,n} = \{G \in \mathbb{R}^{m \times n} : G^T G = I_n\}. \tag{6.16}$$

Then, the TT formulation of the DL problem takes the form

$$\min_{G_1, \mathcal{G}_2, G_3, \mathcal{X}} \|\mathcal{Y} - (G_1 \times_2^1 \mathcal{G}_2 \times_3^1 G_3) \times_3^1 \mathcal{X}\|_F^2 \quad s.t. \quad \begin{array}{l} \mathcal{X}_{[1]} \in \Gamma_{k, n_e n_p}^{(\tau)} \;\; G_1 \in \Theta_{n_1, r_1} \\ (\mathcal{G}_2)_{[2]} \in \Theta_{r_1 n_2, r_2} \;\; G_3 \in \Omega_{r_2, k} \end{array} \tag{6.17}$$

By tensor manipulations described in Chapter 1

$$\|\mathcal{Y} - (G_1 \times_2^1 \mathcal{G}_2 \times_3^1 G_3) \times_3^1 \mathcal{X}\|_F = \|\mathcal{Y}_{[2]} - (I_{n_2} \otimes G_1) (\mathcal{G}_2)_{[2]} G_3 \mathcal{X}_{[1]}\|_F.$$

The constraints on G_1 and $(\mathcal{G}_2)_{[2]}$ are due to the particular tensor decomposition used, while the constraints on G_3 and $\mathcal{X}_{[1]}$ are inherited from the DL formulation. In particular, using the TT formulation, the constraint on the columns of $\mathcal{D}_{[2]}$ in (6.15) becomes a constraint on the columns of G_3. This can be easily proved using the orthogonality of G_1 and $(\mathcal{G}_2)_{[2]}$ as follows

$$\left\|\mathcal{D}_{[2]}(:,j)\right\|_2 = \left\|\left(I_{n_2} \otimes G_1\right)(\mathcal{G}_2)_{[2]} G_3(:,j)\right\|_2 = \left\|G_3(:,j)\right\|_2 .$$

The formulation described above can be extended to multiway tensors. Let $\mathcal{Y} \in \mathbb{R}^{n_1 \times \cdots \times n_q \times \cdots \times n_s}$ be the tensor obtained by storing $n_{q+1} \times \cdots \times n_s$ tensors of dimension $n_1 \times \cdots \times n_q$. Then, the TT formulation can be written as

$$\begin{aligned} &\min_{G_1,\ldots,G_{q+1},\mathcal{X}} \left\|\mathcal{Y} - \left(G_1 \times_2^1 \mathcal{G}_2 \times_3^1 \cdots \times_3^1 G_{q+1}\right) \times_{q+1}^1 \mathcal{X}\right\|_F^2 s.t. \\ &\mathcal{X}_{[1]} \in \Gamma^{(\tau)}_{k,n_{q+1}\ldots n_s},\ \ G_1 \in \Theta_{n_1,r_1}\ \ G_{q+1} \in \Omega_{r_q,k}, \\ &(\mathcal{G}_j)_{[2]} \in \Theta_{r_{j-1}n_j,r_{j+1}} \ \text{ for } \ \ j = 2,\ldots,q. \end{aligned} \tag{6.18}$$

This formulation can be used when either the dimensionality of the database or the dimensionality of the single data is higher than 2. For example, consider a database of $n_1 \times n_2 \times n_3$ RGB images of n_p subjects in n_e expressions, n_{ill} illumination and n_v view angles. This can be represented as a 7th-order tensor $\mathcal{Y} \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times n_e \times n_{ill} \times n_v \times n_p}$. In this context $\mathcal{X} \in \mathbb{R}^{k \times n_e \times n_{ill} \times n_v \times n_p}$ and $\mathcal{D} \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times k}$ can be represented using four TT-cores. Thus, we can use (6.18) with $q = 3$ and $s = 7$.

Now we reformulate the Tensor-Train Dictionary Learning problem (6.17) for third-order tensors by including the constraints in the objective function. The extension to Nth-order tensors is straightforward. As for the matrix DL problem, (6.19) becomes

$$\min_{G_1,\mathcal{G}_2,G_3,\mathcal{X}} H(G_1,\mathcal{G}_2,G_3,\mathcal{X}) + \delta_{\Gamma^{(\tau)}_{k,n_e n_p}}(\mathcal{X}_{[1]}) + \delta_{\Theta_{n_1,r_1}}(G_1) + \delta_{\Theta_{r_1 n_2,r_2}}((\mathcal{G}_2)_{[2]}) + \delta_{\Omega_{r_2,k}}(G_3), \tag{6.19}$$

where

$$H(G_1,\mathcal{G}_2,G_3,\mathcal{X}) = \|\mathcal{Y} - (G_1 \times_2^1 \mathcal{G}_2 \times_3^1 G_3) \times_3^1 \mathcal{X}\|_F^2. \tag{6.20}$$

Furthermore in this setting we use the following notation

- $\Psi_1(G_1,G_2,G_3) = \|Y - (I_{n_2} \otimes G_1) G_2 G_3 X\|_F^2 + \delta_{\Theta_{n_1,r_1}}(G_1)$
- $\Psi_2(G_1,G_2,G_3) = \|Y - (I_{n_2} \otimes G_1) G_2 G_3 X\|_F^2 + \delta_{\Theta_{r_1 n_2,r_1}}(G_2)$ with $G_2 = (\mathcal{G}_2)_{[2]}$
- $\Psi_3(G_1,G_2,G_3) = \|Y - (I_{n_2} \otimes G_1) G_2 G_3 X\|_F^2 + \delta_{\Omega_{r_2,k}}(G_3)$
- $\Psi_4(G_1,G_2,G_3) = \|Y - (I_{n_2} \otimes G_1) G_2 G_3 X\|_F^2 + \delta_{\Gamma^{(\tau)}_{k,n_e n_p}}(X)$ with $X = \mathcal{X}_{[1]}$

In order to apply PALM framework to (6.19) we provide the Lipschitz constants of the partial gradient of H in (6.20), where L_S is the Lipschitz constant of H corresponding to the variable S in the $\|\cdot\|_2$ norm.

Proposition 6.2.1. *Set* $G_2 = (\mathcal{G}_2)_{[2]}$, $X = \mathcal{X}_{[1]}$, $Y = \mathcal{Y}_{[2]}$ *and consider*

$$H(G_1, G_2, G_3, X) = \|Y - (I_{n_2} \otimes G_1)\, G_2 G_3 X\|_F^2 .$$

Then the partial gradients of H *are globally Lipschitz. For* G_1, G_2 *having orthonormal columns, the Lipschitz constants satisfy*

$$L_X = 2\|G_3\|_2^2, \quad L_{G_2} = 2\|G_3 X\|_2^2, \quad L_{G_3} = 2\|X\|_2^2$$

and $L_{G_1} = 2\|\sum_{i=1}^{p} \left(A_i A_i^T\right)\|_2$, *where* $A_i \in \mathbb{R}^{r_1 \times n_2}$ *is the matricization of the* i*th column of* $A = G_2 G_3 X$.

Proof. By direct computation the following expressions for the partial gradients of H hold:

$$\begin{aligned}
\nabla_X H &= 2\left((I_{n_2} \otimes G_1)\, G_2 G_3\right)^T \left(-Y + (I_{n_2} \otimes G_1)\, G_2 G_3 X\right) \\
\nabla_{G_1} H &= 2\sum_{i=1}^{p} (Y_i + G_1 A_i) A_i^T \\
\nabla_{G_2} H &= 2\,(I_{n_2} \otimes G_1)^T \left(-Y + (I_{n_2} \otimes G_1)\, G_2 G_3 X\right) (G_3 X)^T \\
\nabla_{G_3} H &= 2 G_2^T\, (I_{n_2} \otimes G_1)^T \left(-Y + (I_{n_2} \otimes G_1)\, G_2 G_3 X\right) X^T
\end{aligned}$$

where $A_i \in \mathbb{R}^{r_1 \times n_2}$ is the matricization of the ith column of $A = G_2 G_3 X$ and Y_i is the matricization of the ith column of Y.

Using these expressions we show that each partial gradient is globally Lipschitz. In doing so, we derive Lipschitz constants by exploiting the orthonormality of the columns of G_1 and G_2. For any $\hat{X}$ and $\tilde{X}$ we get

$$\begin{aligned}
&\|\nabla_X H(G_1, G_2, G_3, \hat{X}) - \nabla_X H(G_1, G_2, G_3, \tilde{X})\|_2 \\
&= 2\|G_3^T G_2^T (I_{n_2} \otimes G_1)^T (I_{n_2} \otimes G_1) G_2 G_3 (\hat{X} - \tilde{X})\|_2 \\
&= 2\|G_3^T G_3 (\hat{X} - \tilde{X})\|_2 < L_X \|\hat{X} - \tilde{X}\|_2,
\end{aligned} \tag{6.21}$$

with $L_X = 2\left\|G_3^T G_3\right\|_2 = 2\|G_3\|_2^2$. In a similar manner we obtain the following inequalities for $\nabla_{G_i} H$, $i = 1, 2, 3$.

In particular, for any $\hat{G}_1$ and $\tilde{G}_1$ we get

$$\begin{aligned}
&\|\nabla_{G_1} H(\hat{G}_1, G_2, G_3, X) - \nabla_{G_1} H(\tilde{G}_1, G_2, G_3, X)\|_2 \\
&= 2\|\sum_{i=1}^{p} (\hat{G}_1 - \tilde{G}_1) A_i A_i^T\| < L_{G_1} \|(\hat{G}_1 - \tilde{G}_1)\|,
\end{aligned}$$

with $L_{G_1} = 2\|\sum_{i=1}^{p} A_i A_i^T\|_2$. Moreover, for any $\hat{G}_2$ and $\tilde{G}_2$, we have

$$\begin{aligned}
&\|\nabla_{G_2} H(G_1, \hat{G}_2, G_3, X) - \nabla_{G_1} H(G_1, \tilde{G}_2, G_3, X)\|_2 \\
&= 2\|(I_{n_2} \otimes G_1)^T (I_{n_2} \otimes G_1)(\hat{G}_2 - \tilde{G}_2) G_3 X (G_3 X)^T\|_2 \\
&= 2\|(\hat{G}_2 - \tilde{G}_2) G_3 X (G_3 X)^T\|_2 < L_{G_2} \|(\hat{G}_2 - \tilde{G}_2)\|_2,
\end{aligned} \tag{6.22}$$

where $L_{G_2} = 2\|G_3 X X^T G_3^T\|_2 = 2\|G_3 X\|_2^2$. Finally, for any $\hat{G}_3$ and $\tilde{G}_3$ we have

$$\begin{aligned}
&\|\nabla_{G_3} H(G_1, G_2, \hat{G}_3, X) - \nabla_{G_3} H(G_1, G_2, \tilde{G}_3, X)\|_2 \\
&= 2 \left\| G_2^T \left(I_{n_2} \otimes G_1\right)^T \left(I_{n_2} \otimes G_1\right) G_2 (\hat{G}_3 - \tilde{G}_3) X X^T \right\|_2 \\
&= 2\|(\hat{G}_3 - \tilde{G}_3) X X^T\|_2 \leq L_{G_3} \|\hat{G}_3 - \tilde{G}_3\|_2,
\end{aligned}$$

where $L_{G_3} = 2\|X X^T\|_2 = 2\|X\|_2^2$. □

It is worth to notice that the explicit calculation of $I \otimes G_1$ can be avoided within the partial gradients computations using the following remark.

Remark 6.2.1. *Using the orthogonality of $I \otimes G_1$ and the properties of the Kronecker product, the partial gradient $\nabla_X H$ in Proposition 6.2.1 can be computed as*

$$\nabla_X H = G_3^T G_2^T (-\tilde{G}^T + G_2 G_3 X), \tag{6.23}$$

where $\tilde{G} = (G_1^T \mathcal{Y}_{[1]})_{[2]}$.

As already discussed, PALM-type algorithms at each iteration require the computation of a proximal step in each variable block, we now give the formal expression of proximal operators for the indicator function on the set $\Theta_{m,n}$ of $m \times n$ matrices with orthonormal columns defined in (6.16).

Proposition 6.2.2. *Let $A \in \mathbb{R}^{m \times n}$ and $\Theta_{m,n} \subset \mathbb{R}^{m \times n}$ be defined in (6.16). Then*

$$P_{\Theta_{m,n}}(A) = U V^T,$$

where U and V are respectively the left and right singular matrices of A.

Proof. For $B \in \Omega \subset \mathbb{R}^{m \times n}$ having unit norm columns, we have that $\|A - B\|_F^2 = \operatorname{trace}(A^T A) - 2\operatorname{trace}(B^T A) + n$ and thus solving (6.4) is equivalent to finding $B = \operatorname{argmax}_\Omega \operatorname{trace}(A^T B)$. Now let $A = U \Sigma V^T$ be the SVD of A, and let $\bar{n} = \min\{m, n\}$. Then

$$\operatorname{trace}(B^T A) = \operatorname{trace}(B^T U \Sigma V^T) = \operatorname{trace}(V^T B^T U \Sigma) = \sum_{i=1}^{\bar{n}} z_{ii} \sigma_i \leq \sum_{i=1}^{\bar{n}} \sigma_i,$$

where z_{ii} is the ith diagonal element of $Z = V^T B^T U$ and σ_i are the singular values of A. In particular, note that $z_{ii} < 1$ as $e_i^T V^T B^T U e_i < \|V e_i\| \|B\| \|U e_i\| = 1$. The upper bound is reached for Z equal to the identity matrix, which is obtained for $B = U V^T$. □

Algorithm 30 describes the general PALM scheme with constant stepsize applied to the tensor formulation (6.19), that is therefore named PALM-DL-TT. The Lipschitz moduli L_X, L_{G_i} $i = 1, 2, 3$ are as introduced in Proposition 6.2.1. The TT algorithmic version of sPALM can be easily obtained following Algorithms 16 and 15 and replacing the stepsizes $\bar{\alpha}_1, \bar{\alpha}_2, \bar{\alpha}_3$ and $\bar{\alpha}_X$ with the spectral stepsizes based of Algorithm 15 and imposing the Armijo sufficient decrease condition. The whole procedure is reported in Algorithm 31.

Algorithm 30 PALM-DL-TT

1: **Input**: Data matrix $\mathcal{Y} \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times n_e \times n_p}$, initial values for $G_1 \in \mathbb{R}^{n_1 \times r_1}$, $\mathcal{G}_2 \in \mathbb{R}^{r_1 \times n_2 \times r_2}$, $G_3 \in \mathbb{R}^{r_2 \times k}$ and $\mathcal{X} \in \mathbb{R}^{k \times n_e \times n_p}$, maximum number τ of non-zero elements of each fiber of $\mathcal{X}$, $\eta_1, \eta_2, \eta_3, \eta_X > 1$, $\mu_1, \mu_2, \mu_3, \mu_X > 0$.
2: Set $G_2 \leftarrow (\mathcal{G}_2)_{[2]}$ and $X \leftarrow (\mathcal{X})_{[1]}$
3: Compute $G_1 = P_{\Theta_{n_1,r_1}}(G_1)$, $G_2 = P_{\Theta_{r_1 n_2, r_2}}(G_2)$, $G_3 = P_{\Omega_{r_2,k}}(G_3)$, $X = P_{\Gamma^{(\tau)}_{k,n_e n_p}}(X)$.
4: **repeat**
5: *Update* G_1: Set $L'_{G_1} = \max\{\eta_1 L_{G_1}, \mu_1\}$ and $\bar{\alpha}_1 = 1/L'_{G_1}$ and compute $G_1 = P_{\Theta_{n_1,r_1}}(G_1 - \bar{\alpha}_1 \nabla_{G_1} H)$;
6: *Update* G_2: Set $L'_{G_2} = \max\{\eta_2 L_{G_2}, \mu_2\}$ and $\bar{\alpha}_2 = 1/L'_{G_2}$ and compute $G_2 = P_{\Theta_{r_1 n_2, r_2}}(G_2 - \bar{\alpha}_2 \nabla_{G_2} H)$;
7: *Update* G_3: Set $L'_{G_3} = \max\{\eta_3 L_{G_3}, \mu_3\}$ and $\bar{\alpha}_3 = 1/L'_{G_3}$ and compute $G_3 = P_{\Omega_{r_2,k}}(G_3 - \bar{\alpha}_3 \nabla_{G_3} H)$;
8: *Update* X: Set $L'_X = \max\{\eta_X L_X, \mu_X\}$ and $\bar{\alpha}_X = 1/L'_X$ and compute $X = P_{\Gamma^{(\tau)}_{k,n_e n_p}}(X - \bar{\alpha}_X \nabla_X H)$;
9: **until** convergence

Remark 6.2.2. *The same considerations of Proposition 6.1.5 carry over to the BB stepsizes and the Lipschitz constants for the TT-DL problem (6.19).*

The next theorem contains our main convergence result. While it is stated for PALM-DL-TT described in Algorithm 30, it can be generalized to *any* PALM-type algorithm applied to the TT DL formulation (6.19).

Theorem 6.2.1. *If the sequence generated by PALM-DL-TT in Algorithm 30 is bounded, then it converges to a critical point of the TT formulation of the DL problem (6.19) and it has finite length property.*

Proof. The result is proved by observing that the function H in (6.19) is twice continuously differentiable, the functions in the objective of problem (6.19) are semi-algebraic and satisfy Assumption 3.3.1. Indeed, all indicator functions are proper, lower semi-continuous and semi-algebraic: the sets $\Omega_{n,m}$ and $\Gamma_{k,n_e n_p}$ are semi-algebraic (see [2, 52]) and $\Theta_{n,m}$ is a closed semi-algebraic set for any n and m. Also, Proposition 6.2.1 ensures that Assumption 3.3.1 A3-A5 hold (see [9, Remark 3]). □

Algorithm 31 sPALM-DL-TT

1: **Input**: Data matrix $\mathcal{Y} \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times n_e \times n_p}$, initial values for $G_1 \in \mathbb{R}^{n_1 \times r_1}$, $\mathcal{G}_2 \in \mathbb{R}^{r_1 \times n_2 \times r_2}$, $G_3 \in \mathbb{R}^{r_2 \times k}$ and $\mathcal{X} \in \mathbb{R}^{k \times n_e \times n_p}$, maximum number τ of non-zero elements of each fiber of $\mathcal{X}$, $\alpha_{min}, \alpha_{max}$, $\delta_1, \delta_2, \delta_3, \delta_X > 0$, $\rho_1, \rho_2, \rho_3, \rho_X > 0$.
2: Set $G_2 \leftarrow (\mathcal{G}_2)_{[2]}$ and $X \leftarrow \mathcal{X}_{[1]}$
3: **for** $j = 0, 1, \ldots,$ **do**
4: *Update* G_1: Set $(G_1)_{j+1} = P_{\Theta_{n_1,r_1}}\left((G_1)_j - \bar{\alpha}_{j,1}\nabla_{G_1} H\left((G_1)_j, (G_2)_j, (G_3)_j, X_j\right)\right)$ where $\bar{\alpha}_{j,1} = \rho_1^{i_j}\alpha_{j,1}$ and i_j is the smallest nonnegative integer for which

$$\Psi_1\left((G_1)_{j+1}, (G_2)_j, (G_3)_j, X_j\right) < \Psi_1\left((G_1)_j, (G_2)_j, (G_3)_j, X_j\right) - \frac{\delta_1}{2\bar{\alpha}_{j,1}} \| (G_1)_{j+1} - (G_1)_j \|_F^2.$$

5: Compute $\alpha_{j+1,1} \in [\alpha_{min}, \alpha_{max}]$ using Algorithm 28 with $S_{j,G_1} = (G_1)_{j+1} - (G_1)_j$ and $Z_{j,G_1} = \nabla_{G_1} H\left((G_1)_{j+1}, (G_2)_j, (G_3)_j, X_j\right) - \nabla_{G_1} H\left((G_1)_j, (G_2)_j, (G_3)_j, X_j\right)$.
6: *Update* G_2: Set $(G_2)_{j+1} = P_{\Theta_{r_1 n_2, r_2}}\left((G_2)_{j+1} - \bar{\alpha}_{j,2}\nabla_{G_2} H\left((G_1)_{j+1}, (G_2)_j, (G_3)_j, X_j\right)\right)$ where $\bar{\alpha}_{j,2} = \rho_2^{i_j}\alpha_{j,2}$ and i_j is the smallest nonnegative integer for which

$$\Psi_2\left((G_1)_{j+1}, (G_2)_{j+1}, (G_3)_j, X_j\right) < \Psi_2\left((G_1)_{j+1}, (G_2)_j, (G_3)_j, X_j\right) - \frac{\delta_2}{2\bar{\alpha}_{j,2}} \| (G_2)_{j+1} - (G_2)_j \|_F^2.$$

7: Compute $\alpha_{j+1,2} \in [\alpha_{min}, \alpha_{max}]$ using Algorithm 28 with $S_{j,G_2} = (G_2)_{j+1} - (G_2)_j$ and $Z_{j,G_2} = \nabla_{G_2} H\left((G_1)_{j+1}, (G_2)_{j+1}, (G_3)_j, X_j\right) - \nabla_{G_2} H\left((G_1)_{j+1}, (G_2)_j, (G_3)_j, X_j\right)$.
8: *Update* G_3: Set $(G_3)_{j+1} = P_{\Omega_{r_2,k}}\left((G_3)_{j+1} - \bar{\alpha}_{j,3}\nabla_{G_3} H\left((G_1)_{j+1}, (G_2)_{j+1}, (G_3)_j, X_j\right)\right)$ where $\bar{\alpha}_{j,3} = \rho_3^{i_j}\alpha_{j,3}$ and i_j is the smallest nonnegative integer for which

$$\Psi_3\left((G_1)_{j+1}, (G_2)_{j+1}, (G_3)_{j+1}, X_j\right) < \Psi_3\left((G_1)_{j+1}, (G_2)_{j+1}, (G_3)_j, X_j\right) - \frac{\delta_3}{2\bar{\alpha}_{j,3}} \| (G_3)_{j+1} - (G_3)_j \|_F^2.$$

9: Compute $\alpha_{j+1,3} \in [\alpha_{min}, \alpha_{max}]$ using Algorithm 28 with $S_{j,G_3} = (G_3)_{j+1} - (G_3)_j$ and $Z_{j,G_3} = \nabla_{G_3} H\left((G_1)_{j+1}, (G_2)_{j+1}, (G_3)_{j+1}, X_j\right) - \nabla_{G_3} H\left((G_1)_{j+1}, (G_2)_{j+1}, (G_3)_j, X_j\right)$.
10: *Update* X: Set $X_{j+1} = P_{\Gamma^{(\tau)}_{k,n_e n_p}}\left(X_{j+1} - \bar{\alpha}_{j,X}\nabla_X H\left((G_1)_{j+1}, (G_2)_{j+1}, (G_3)_{j+1}, X_j\right)\right)$ where $\bar{\alpha}_{j,X} = \rho_X^{i_j}\alpha_{j,X}$ and i_j is the smallest nonnegative integer for which

$$\Psi_4\left((G_1)_{j+1}, (G_2)_{j+1}, (G_3)_{j+1}, X_{j+1}\right) < \Psi_4\left((G_1)_{j+1}, (G_2)_{j+1}, (G_3)_{j+1}, X_j\right) - \frac{\delta_X}{2\bar{\alpha}_{j,X}} \|X_{j+1} - X_j\|_F^2.$$

11: Compute $\alpha_{j+1,X} \in [\alpha_{min}, \alpha_{max}]$ using Algorithm 28 with $S_{j,X} = X_{j+1} - X_j$ and $Z_{j,X} = \nabla_X H\left((G_1)_{j+1}, (G_2)_{j+1}, (G_3)_{j+1}, X_{j+1}\right) - \nabla_X H\left((G_1)_{j+1}, (G_2)_j, (G_3)_{j+1}, X_j\right)$.
12: **end for**

Chapter 7

Dictionary Learning for image classification

In this chapter we apply the PALM-based algorithms to the Dictionary Learning problem in image classification. More precisely, in Section 7.1 we explain how to deal with a classification problem within the DL framework both in the matrix and in the tensor setting by presenting two different classification models. Furthermore we provide some useful considerations to apply PALM-based algorithms in this context. In Section 7.2 we show some numerical experiments on benchmark datasets to highlight the advantages in using a spectral step both in the matrix and in the tensor formulation. Finally, we underline the advantages in using a tensor approach for memory savings in the 3D setting and for classification performance in the 4D setting.

7.1 The classification problem

Among several image processing tasks the DL formulation has also been applied in classification contexts showing a great potential in addressing this task (see e.g. [2, 83]). In this chapter we focus on two main kinds of classification algorithms of the DL literature while we refer the reader to [26, Chapter 8] for an overview of different DL classification algorithms. In the first, the dictionary D and the sparse matrix X are learnt from data by solving (5.4) and a classifier matrix $W \in \mathbb{R}^{n_p \times k}$ is computed a posteriori as the solution of the following problem:

$$\min_W \|C - WX\|_F^2 + \beta \|W\|_F^2, \tag{7.1}$$

where β is a positive small constant, and the matrix $C \in \mathbb{R}^{n_p \times n_e n_p}$ contains the labels of the images stored in Y. In particular, if the column i of Y contains the person ℓ, then the ith column of C is equal to e_ℓ, the ℓth canonical basis vector.

In the second category of classification algorithms the classifier W is learnt from the

data together with X and D:

$$\min_{D,X,W} \|Y - DX\|_F^2 + \gamma\|C - WX\|_F^2 \quad s.t. \quad D \in \Omega_{n,k}, \quad X \in \Gamma_{k,p}^{(\tau)}, \tag{7.2}$$

where γ is a positive constant. This formulation aims to enforce the model to have a representative strength together with a discriminative action, thanks to the presence of W in the minimization procedure. For this reason this classification algorithm has been named "discriminative DL"; see, e.g., [26, sec.8.5.2] and references therein. To apply PALM framework to (7.2), we rewrite the problem by including the constraints in the objective function as

$$\min_{D,X,W} \|Y - DX\|_F^2 + \gamma\|C - WX\|_F^2 + \delta_{\Omega_{n,k}}(D) + \delta_{\Gamma_{k,p}^{(\tau)}}(X). \tag{7.3}$$

Then, a convergence result can be proved for the problem (7.3) by generalizing Propositions 6.1.1 and 6.1.3. More precisely, it can be proved that (7.3) is semi-algebraic by observing that the term $\gamma\|C - WX\|_F^2$ is a real polynomial function. Then using the smoothness of $H_C(D, X, W) := \|Y - DX\|_F^2 + \gamma\|C - WX\|_F^2$ and the fact that its partial gradient are Lipschitz continuous with moduli

$$L_X = 2\|D^T D\|_2 + 2\gamma\|W^T W\|_2, \quad L_D = 2\|XX^T\|_2, \quad \text{and} \quad L_W = 2\gamma\|XX^T\|_2.$$

we can conclude the convergence of PALM-based algorithms for (7.3).

Once the classifier has been computed, the classification task proceeds as follows: Given a new image $y \in \mathbb{R}^n$ to be classified, its sparse representation $x \in \mathbb{R}^k$ is computed, e.g., by OMP-type algorithms, see Chapter 5, Algorithms 18 and 19. Then, y is classified as person $\hat{\ell} = \text{argmax}_i |Wx|_i$.

Preserving the multidimensional structure of the data can be extremely useful also in classification contexts. For example a database of black and white images of n_p persons in n_e expressions can be represented by a fourth-order tensor $\mathcal{Y} \in \mathbb{R}^{n_1 \times n_2 \times n_e \times n_p}$, where n_1 and n_2 are the number of rows and columns of a single image respectively. Thus we rewrite (7.2) in a tensor setting as

$$\begin{aligned} &\min_{\mathcal{D},\mathcal{X},W} \left\|\mathcal{Y} - \mathcal{D} \times_3^1 \mathcal{X}\right\|_F^2 + \gamma\|\mathcal{C} - \mathcal{X} \times_1 W\|_F^2 \\ &s.t. \quad \mathcal{D}_{[2]} \in \Omega_{n_1 n_2, k}, \quad \mathcal{X}_{[1]} \in \Gamma_{k, n_e n_p}^{(\tau)}, \; W \in \mathbb{R}^{n_p \times k}. \end{aligned} \tag{7.4}$$

In particular $\mathcal{C}$ is such that its mode-1 unfolding is the matrix C defined in (7.2), i.e. $\mathcal{C}_{[1]} = C$. Using the TT Decomposition the problem is given as

$$\begin{aligned} &\min_{G_1,\mathcal{G}_2,G_3,\mathcal{X}} \left\|\mathcal{Y} - (G_1 \times_2^1 \mathcal{G}_2 \times_3^1 G_3) \times_3^1 \mathcal{X}\right\|_F^2 + \gamma\|\mathcal{C} - \mathcal{X} \times_1 W\|_F^2 \\ &s.t. \qquad \mathcal{X}_{[1]} \in \Gamma_{k, n_e n_p}^{(\tau)} \; G_1 \in \Theta_{n_1, r_1}, \quad (\mathcal{G}_2)_{[2]} \in \Theta_{r_1 n_2, r_2} \; G_3 \in \Omega_{r_2, k}. \end{aligned} \tag{7.5}$$

As discussed in the previous chapter, (7.5) can be generalized to the multi-order setting. In order to apply PALM type algorithms to the tensor classification problem, we rewrite (7.5) as

$$\begin{aligned}\min_{G_1,\mathcal{G}_2,G_3,\mathcal{X},\mathcal{W}} H_C(G_1,\mathcal{G}_2,G_3,\mathcal{X},\mathcal{W}) + \delta_{\Gamma^{(\tau)}_{k,n_e n_p}}(\mathcal{X}_{[1]}) \\ +\delta_{\Theta_{n_1,r_1}}(G_1) + \delta_{\Theta_{r_1 n_2,r_2}}((\mathcal{G}_2)_{[2]}) + \delta_{\Omega_{r_2,k}}(G_3),\end{aligned} \tag{7.6}$$

where $H_C(G_1,\mathcal{G}_2,G_3,\mathcal{X},\mathcal{W}) = \|\mathcal{Y}-(G_1\times_2^1\mathcal{G}_2\times_3^1 G_3)\times_3^1\mathcal{X}\|_F^2+\gamma\|\mathcal{C}-\mathcal{X}\times_1 W\|_F^2$. Theorem 6.2.1 can be extended to this tensor formulation of the classification problem (7.6) by generalizing Proposition 6.2.1 to the function H_C in (7.6) as shown below. All the other considerations still hold.

Proposition 7.1.1. *Consider the function H_C defined below (7.6) and let $G_2 = (\mathcal{G}_2)_{[2]}$ and $X = \mathcal{X}_{[1]}$. Then the partial gradients of H_C are globally Lipschitz and, for G_1, G_2 having orthonormal columns, the Lipschitz constants L_W and L_X, satisfy:*

$$L_W = 2\,\gamma\,\|X\|_2^2, \quad L_X = 2\,\|G_3\|_2^2 + 2\,\gamma\,\|W\|_2^2,$$

while L_{G_1}, L_{G_2} and L_{G_3} are given in Proposition 6.2.1.

In order to compute the sparse representation $x \in \mathbb{R}^k$ of an image $y \in \mathbb{R}^{n_i}$ in this Tensor-Train setting, we derived a variant of OMP named OMP-TT and reported in Algorithm 32. As can be noticed, OMP-TT is a variant of OMP-QR (Algorithm 19) where the orthogonality of the first two TT-cores is used. This algorithm can be extended to higher-order tensor observing that all the TT-cores, except the last one, are orthogonal.

Algorithm 32 OMP-TT

1: **Input**: data vector $y \in \mathbb{R}^n$, cores of the TT Decomposition G_1, $\mathcal{G}_2$, G_3
2: Initialize the vector $x = 0$, $S^{(0)} = support(x) = \emptyset$
3: Compute $\tilde{y} = (\mathcal{G}_2)^T_{[2]}(I_{n_1} \otimes G_1^T)y$
4: Compute $\alpha = G_3^T\tilde{y}$
5: **for** $t = 1, \cdots, \tau$ **do**
6: **Select Atom**:
7: $j^* = argmax_j \alpha$
8: $S^{(it+1)} = S^{(it)} \cup j^*$:
9: Compute the QR decomposition of $(G_3)_{S^{(it+1)}} = Q^{(it+1)}R^{(it+1)}$ starting from the decomposition of $(G_3)_{S^{(it)}}$: $Q^{(it+1)} = [Q^{(it)}\ \tilde{q}]$
10: $r^{(it+1)} = r^{(it)} - \tilde{q}\tilde{q}^T\tilde{y}$
11: **if** $\|r^{(it+1)}\|_2 < \epsilon$ stop
12: **end for**
13: Compute the solution estimate: $x = \left(R^{(it+1)}\right)^{-1}\left(Q^{(it+1)}\right)^T\tilde{y}$
14: **Output**: vector $x \in \mathbb{R}^k$.

PALM-DL	PALM applied to the matrix DL problem (5.4)	Algorithm 26
sPALM-DL	sPALM for the matrix DL problem (5.4)	Algorithm 29
btPALM-DL	btPALM for the matrix DL problem (5.4)	Algorithm 27
PALM-DL-TT	PALM for the TT-DL problem (6.17)	Algorithm 30
sPALM-TT	sPALM for the TT-DL problem (6.17)	Algorithm 31
PALM-CDL	PALM for the matrix classification DL problem (7.2)	
sPALM-CDL	sPALM for the matrix classification DL problem (7.2)	
PALM-CDL-TT	PALM for the classification TT-DL problem (7.5)	
sPALM-CDL-TT	sPALM for the classification TT-DL problem (7.5)	

Table 7.1: All methods employed in our experiments.

7.2 Numerical experiments

In this section we focus our experimental analysis to the convergent PALM type algorithms described in the previous chapters. More precisely, we numerically explore the advantages of using the spectral variant of PALM and the possible benefits of using a TT based algorithm in the solution of the image classification problem. The considered algorithms are compared in terms of efficiency and classification performance both in the matrix and tensor settings. A truncated approach for the TT formulation is also tested. Section 7.2.5 is devoted to the treatment of 5th-order tensors, leading to a 4D implementation of our algorithms; numerical experiments on a suitable database are reported, illustrating the effectiveness of the TT approach.

7.2.1 Implementation details

We consider the PALM-type algorithms described in Table 7.1, implemented in Matlab. All the numerical experiments were conducted on one node HPE ProLiant DL560 Gen10 with 4 Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz and 100 of the 512 Gb of RAM using Matlab R2019a. We consider the *classification rate* as performance measure, that is, the percentage of correctly classified persons or objects over the total number of test images.

As starting approximations for the tested algorithms, a random dictionary $\mathcal{D} \in \mathbb{R}^{n_1 \times n_2 \times k}$ with unit norm frontal slices and a sparse random tensor $\mathcal{X} \in \mathbb{R}^{k \times n_e \times n_p}$ with at most τ non-zero elements per column fiber were used. Then, for the matrix formulation we set $Y = \mathcal{Y}_{[2]}$, $D = \mathcal{D}_{[2]}$ and $X = \mathcal{X}_{[1]}$. A similar choice is made also for the Tensor-Train setting where G_1, $\mathcal{G}_2$ and G_3 are initialized as the TT-cores of $\mathcal{D}$. The initial guess for the classification matrix W, when needed, is computed by solving (7.1) with $\beta = 0$. The parameter k is set such that $n < k < \tilde{n}_e n_p$ as is common in Dictionary Learning. More precisely, we set $k = 441$ for the face databases, $k = 1225$ for the Fashion MNIST and $k = 1600$ for the MNIST. The sparsity parameter τ depends on the number

Database	k	τ	β	γ
MIT	441	$4n_p$	0	1
Extended Yale	441	$4n_p$	0	1
MNIST	1600	$2n_p$	0	1
Fashion MNIST	1225	$4n_p$	0	1

Table 7.2: Parameters of the DL classification problem for different datasets.

of classes, n_p, of the database and is set to $4n_p$ for all the databases except MNIST for which $\tau = 2n_p$. Other choices of these parameters have been explored. However, we just report the results for these values of k and τ which seem to exhibit higher classification rate. Several values for the β and γ in problems (7.1)-(7.5) have been tested and then fixed to the values 0 and 1, respectively. A summary of the parameters used for the problem formulation can be found in Table 7.2.

For the PALM based algorithms, i.e. PALM-DL, PALM-DL-TT and PALM-CDL, PALM-CDL-TT, the μ's are set equal to 10^{-10} and the η's are set equal to 1 as greater values gave much worse practical behavior. Regarding implementations based on the spectral variant we set: $\alpha_{\min} = 10^{-10}$, $\alpha_{\max} = 10^{10}$, the initial stepsizes are set equal to 1, ρ's are set equal to 0.5 and δ's equal to 10^{-4}. Finally, the approximated Lipschitz constants $L_{j,D}$ and $L_{j,X}$ in btPALM-DL are doubled during the backtracking strategy, i.e. $\rho_1 = \rho_2 = 2$ in Algorithm 27.

Deriving a reliable criterion for terminating the iteration is a crucial step towards the development of a robust method. Most DL implementations in the literature seem to merely rely on the number of iterations as stopping criterion. In addition to a safeguard strategy on a maximum (loose) number of iterations, in our implementation the following stopping criterion based on the TT iterates variation was used, that is

$$\|\bar{G}_1 - G_1\|_F + \|\bar{G}_2 - G_2\|_F + \|\bar{G}_3 - G_3\|_F + \|\bar{X} - X\|_F \leq \texttt{tol},$$

where the upper bar denotes the approximate solution from the previous iteration.

7.2.2 Preliminary tests on the matrix DL formulation

In this section we want to explore the potential of the spectral gradient step compared to that based on the Lipschitz constants in the solution of the matrix DL problem (5.4). To this end we compare sPALM-DL with PALM-DL and btPALM-DL on the four datasets described in Table A.1. For this experiment we set the maximum number of iterations equal to 50 and the tolerance `tol` in the stopping criterion $\|\bar{D} - D\|_F + \|\bar{X} - X\|_F \leq$ `tol` equal to $\texttt{tol} = 10^{-3}\sqrt{n_1 n_2 k + k n_e n_p}$, where once again, the upper bar denotes the approximate solution from the previous iteration.

The plots in Figure 7.1 show the values of $H(D, X) = \|Y - DX\|_F$ as the iterations proceed, for the considered methods and all datasets. The use of a backtracking

rule (btPALM-DL) to estimate the Lipschitz constants yields a much faster decrease in the residual value than using the constant stepsize (PALM-DL). Moreover, for all the databases, sPALM-DL converges in far fewer iterations than btPALM-DL. Though each iteration of sPALM-DL is more expensive, (see the larger slope in Figure 7.2) the CPU time of the spectral method is significantly lower than for the other methods, illustrating the advantage of using higher order information in the gradient step.

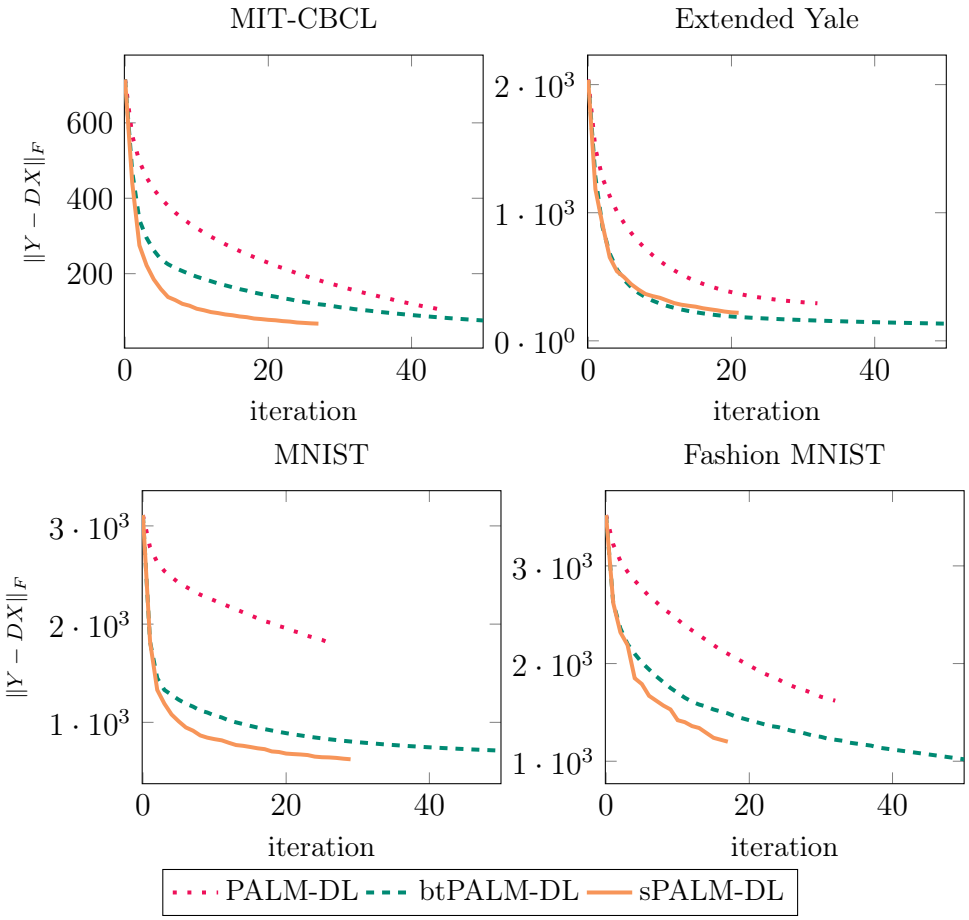


Figure 7.1: Residual norm history for PALM-DL, btPALM-DL, sPALM-DL.

Table 7.3 reports the number of iterations and the rate of classification success for all methods: sPALM-DL satisfies the stopping criterion in the smallest number of iterations, whereas btPALM-DL reaches 50 iterations on all datasets.

Overall, the advantages of using a spectral step in sPALM-DL is a faster decrease of the objective function than with PALM-DL and btPALM-DL (see Figure 7.1), obtained in less CPU time, with similar classification rates.

For the sake of completeness we also report in Table 7.4 a comparison between the K-SVD and the sPALM algorithm in terms of CPU time needed to obtain a comparable value of the objective function. For this experiment, the sPALM algorithm is run until the

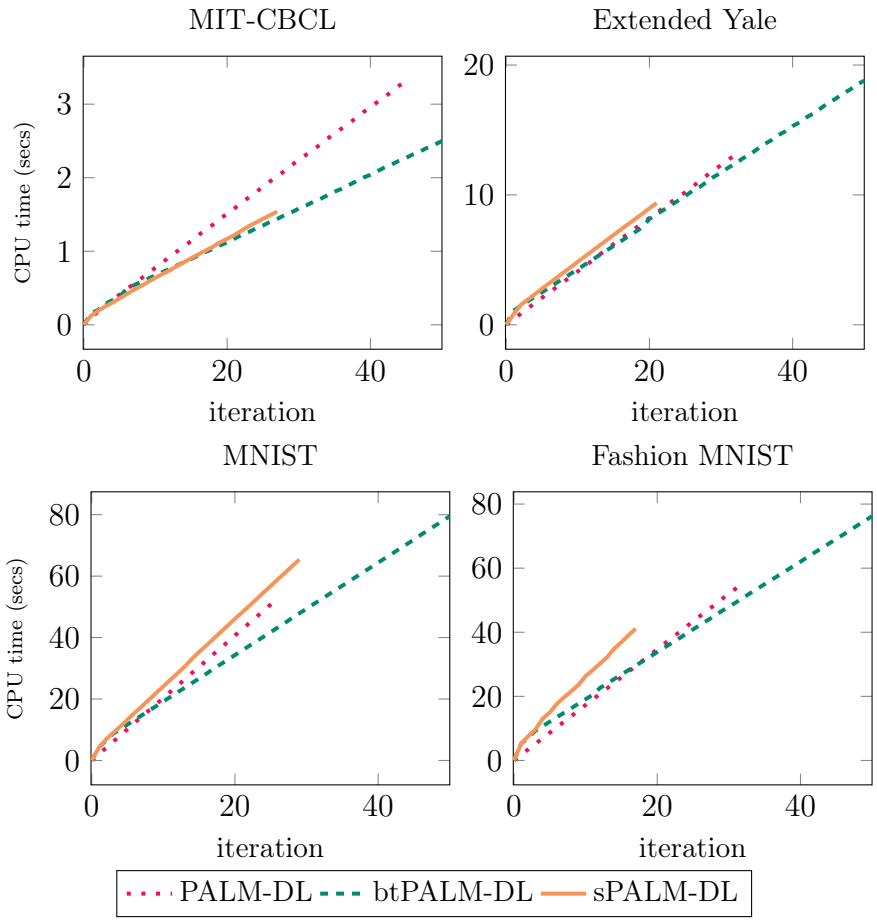


Figure 7.2: CPU time as iterations increase, for PALM-DL, btPALM-DL, sPALM-DL.

objective function is smaller than the value of the objective function obtained with two iterations of K-SVD. Notice that for all the databases, except MNIST, sPALM-DL gives almost the same value of the objective function in less CPU time than K-SVD, although the most time consuming parts of the latter algorithm are precompiled C routines.

7.2.3 Numerical experiments on the tensor DL formulation

Given the training set $\mathcal{Y} \in \mathbb{R}^{n_1 \times n_2 \times \tilde{n}_e \times n_p}$, we solve the DL classification problems using the two strategies described in this chapter.

The first consists in using PALM-DL, sPALM-DL to solve (6.1), and PALM-DL-TT and sPALM-TT to solve (6.19) and then in determining the classification matrix W by solving (7.1). Figure 7.3 displays the classification success rate of all algorithms as the iterations proceed. The use of a spectral step results in higher classification performance for all examined data. For the MIT-CBCL, sPALM based algorithms achieve the maximum classification rate after 20 iterations while PALM-DL and PALM-DL-TT need

	PALM-DL		btPALM-DL		sPALM-DL	
Database	%	#it	%	#it	%	#it
MIT-CBCL	100%	46	99.38%	50	100%	27
Ext'd Yale shrunk	90.62%	32	93.12%	50	90.94%	21
MNIST	66.89%	26	84.14%	50	84.76%	29
Fashion MNIST	68.56%	32	65.60%	50	67.38%	17

Table 7.3: Successful classification rates and number of iteration of PALM-DL, btPALM-DL, sPALM-DL on four different databases.

	sPALM-DL		K-SVD	
Database	$\|Y - DX\|_F$	CPU time	$\|Y - DX\|_F$	CPU time
MIT-CBCL	61.98	2.06	62.77	8.7
Ext'd Yale shrunk	109.39	80.31	109.80	148.32
MNIST	761.14	122.93	763.06	91.34
Fashion MNIST	827.76	132.57	829.51	140.98

Table 7.4: Value of the objective function and CPU time for sPALM-DL and K-SVD.

more iterations to reach the same rate. When processing MNIST the classification performance of PALM-DL and PALM-DL-TT decreases as iterations progress, suggesting overfitting, whereas a slight improvement occurs with sPALM-DL and sPALM-TT. On these datasets, the Tensor-Train formulation does not seem to be beneficial for classification purposes. Computer memory limitations however may favour the tensor approach, as we will discuss in the next section.

The second strategy consists in applying the various PALM variants to the penalized problems (7.3) and (7.6). Also in this setting, the sPALM implementations outperform the PALM based ones. It is also interesting to observe that considering the formulation (7.3) does not result in better classification rates compared with (6.1). Once again, the tensor approach behaves similarly to the matrix formulation.

7.2.4 Truncated approach for memory saving

One of the most challenging aspects in dealing with huge databases is to reduce memory requirements. For a database $\mathcal{Y} \in \mathbb{R}^{n_1 \times n_2 \times n_e \times n_p}$, PALM-DL and sPALM-DL store $\mathcal{D} \in \mathbb{R}^{n_1 \times n_2 \times k}$ and (sparse) $\mathcal{X} \in \mathbb{R}^{k \times n_e \times n_p}$, requiring $m_P := n_1 n_2 k + \tau n_e n_p$ memory allocations. This quantity can be quite large in real image applications. We next investigate the possibility of truncating the tensor decomposition, possibly without interfering with the classification performance. In the Tensor-Train based algorithms PALM-DL-TT and sPALM-TT storage for the arrays $G_1 \in \mathbb{R}^{n_1 \times r_1}$, $\mathcal{G}_2 \in \mathbb{R}^{r_1 \times n_2 \times r_2}$, $G_3 \in \mathbb{R}^{r_2 \times k}$ and

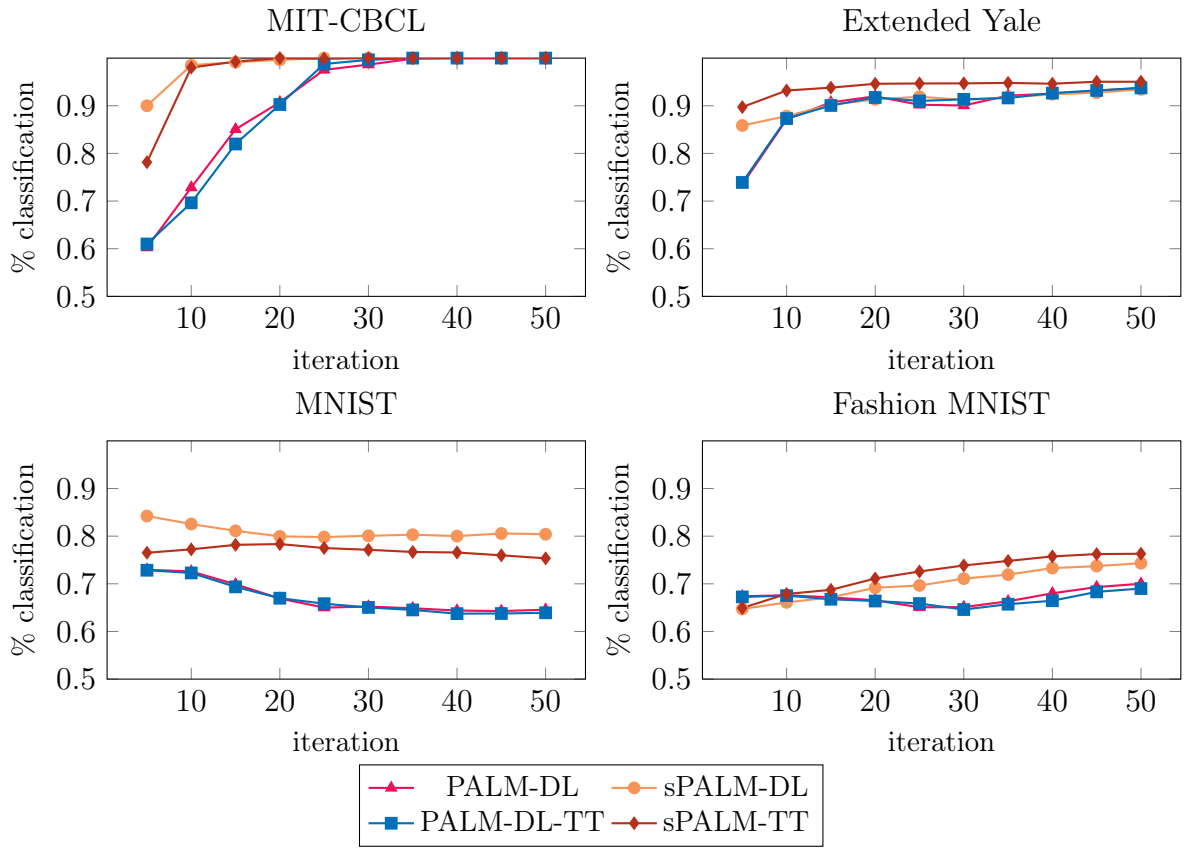


Figure 7.3: Classification performance of PALM-DL, sPALM-DL, PALM-DL-TT and sPALM-TT with respect to the number of iterations for four different database, using the formulations (6.1) and (6.19).

$\mathcal{X} \in \mathbb{R}^{k \times n_e \times n_p}$ is required, yielding $m_{TT} := n_1 r_1 + r_1 n_2 r_2 + r_2 k + \tau n_e n_p$ allocations. As discussed in Section 2.3, in the Tensor-Train decomposition the truncation can be performed either by using a threshold on the error of the unfoldings or by choosing a proper value of the TT-ranks. In this context we set $r_1 = n_1$, i.e. no truncation is performed along the first mode, and we explore the behaviour of the algorithms with respect to r_2. The value of r_2 determines whether the (truncated) TT approach is more memory efficient than the full scheme by comparing m_{TT} and m_P. In Figure 7.5 we show the classification rates for all TT based methods on two of the datasets after 50 iterations, as r_2 varies up to the maximum value obtainable for that dataset ($r_2 \leq 225$ and $r \leq 300$ for MIT-CBCL and Extended Yale, resp.). We note that the TT variants are able to achieve good classification performance also with small values of r_2. In particular, for the MIT-CBCL and sPALM-TT choosing a value of r_2 greater than 40 has no benefit on the classification performance, suggesting the use of $r_2 = 40$, thus reducing the overall memory costs with respect to PALM ($m_P = 108,945$ vs $m_{TT} = 36,585$). Similarly, for Extended Yale the value $r_2 = 150$ can be chosen without dramatically spoiling the

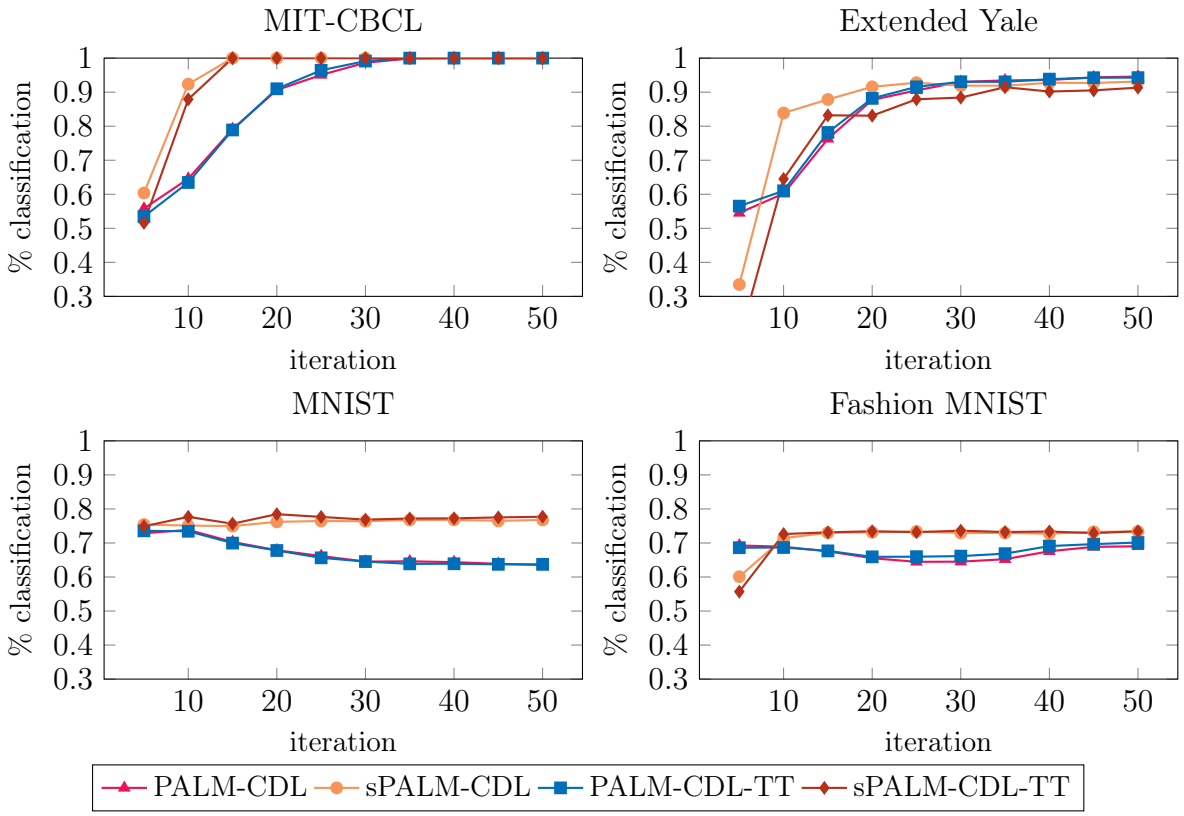


Figure 7.4: Classification performance of PALM-CDL, sPALM-CDL, PALM-CDL-TT and sPALM-CDL-TT with respect to the number of iterations for the four databases, using the formulations (7.3) and (7.6).

classification performance. In other words the Tensor-Train Decomposition enables us to store the information useful for classification purposes in a more compact manner. Depending on the specific application, several strategies to determine a proper value of r_2 can be explored. For example, one may be interested in the minimum value of r_2 that gives a certain accuracy in the classification. In another setting, instead, one may want to limit the memory requirements selecting the maximum value of r_2 that keeps the memory requirements below that threshold. Finally, if one is interested in preserving a certain "expressiveness" of the dictionary, other strategies based on the percentage of singular values of $(\mathcal{G}_2)_{[2]}$ can be choosen, similarly to the one described in Chapter 2.

7.2.5 A classification example in 4D setting

The Tensor-Train decomposition allows us to readily extend the 3D formulation, explored in the previous sections, to higher-order tensors. In the following we analyze the classification performance of PALM-DL-TT and sPALM-TT for a 5th-order tensor $\mathcal{Y} \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times \tilde{n}_e \times n_p}$ and we compare them with their matrix versions PALM-DL and

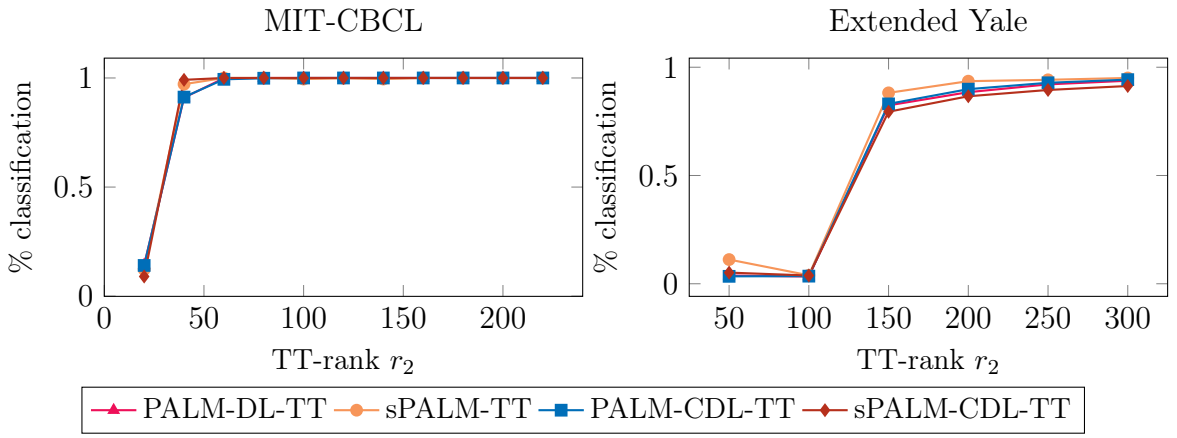


Figure 7.5: Classification performance of the Tensor-Train based algorithms for different values of the TT-rank r_2.

sPALM-DL. First of all, we write the TT Dictionary Learning problem as in (6.18) with $q = 3$ and $s = 5$. Furthermore, we notice that

$$\|\mathcal{Y} - (G_1 \times_2^1 \mathcal{G}_2 \times_3^1 \mathcal{G}_3 \times_3^1 G_4) \times_4^1 \mathcal{X}\|_F = \|\mathcal{Y}_{[3]} - (I_{n_2 n_3} \otimes G_1) \left(I_{n_3} \otimes (\mathcal{G}_2)_{[2]}\right) (\mathcal{G}_3)_{[2]} G_4 \mathcal{X}_{[1]}\|_F \tag{7.7}$$

where G_1,$(\mathcal{G}_2)_{[2]}$, $(\mathcal{G}_3)_{[2]}$, are matrices with orthonormal columns, and more precisely, $G_1 \in \Theta_{n_1,r_1}$, $(\mathcal{G}_2)_{[2]} \in \Theta_{r_1 n_2, r_2}$, $(\mathcal{G}_3)_{[2]} \in \Theta_{r_2 n_3, r_3}$ while $G_4 \in \Omega_{r_3,k}$ has unit norm columns. The following proposition provides an expression for the gradient of H and corresponding Lipschitz constants using the orthogonality of the first three TT-cores.

Proposition 7.2.1. *Let* $G_2 = (\mathcal{G}_2)_{[2]}$, $G_3 = (\mathcal{G}_3)_{[2]}$, $X = \mathcal{X}_{[1]}$, $Y = \mathcal{Y}_{[3]}$ *and*

$$H(G_1, G_2, G_3, G_4, X) = \|Y - (I_{n_2 n_3} \otimes G_1)(I_{n_3} \otimes G_2) G_3 G_4 X\|_F^2 .$$

Then the partial gradients of H *satisfy Assumption A3. Moreover, the following upper bounds for the Lipschitz constants hold:* $L_X = 2\|G_4\|_2^2$, $L_{G_3} = 2\|G_4 X\|_2^2$, $L_{G_4} = 2\|X\|_2^2$, $L_{G_1} = 2\|\sum_{i=1}^p A_i A_i^T\|_2$, $L_{G_2} = 2\|\sum_{i=1}^p B_i B_i^T\|_2$, *where* $A_i \in \mathbb{R}^{r_1 \times n_2}$ *is the matricization of the* i*th column of* $A = (I_{n_3} \otimes G_2) G_3 G_4 X$ *and* $B_i \in \mathbb{R}^{r_1 \times n_2}$ *is the matricization of the* i*th column of* $B = G_3 G_4 X$.

Proof. By direct computation we obtain the following expressions for the partial gradients of H:

$$\nabla_X H = -2 (I_{n_2 n_3} \otimes G_1)(I_{n_3} \otimes G_2) G_3 G_4)^T Y + 2(G_3 G_4)^T G_3 G_4 X$$

and $\nabla_{G_1} H = 2\sum_{i=1}^p (-Y_i + G_1 A_i) A_i^T$, where Y_i denotes the matricization of the ith column of Y, $\nabla_{G_2} H = 2\sum_{i=1}^p (-Y_i + G_1 B_i) B_i^T$,

$$\nabla_{G_3} H = 2((I_{n_2 n_3} \otimes G_1)(I_{n_3} \otimes G_2))^T \left(-Y + (I_{n_2 n_3} \otimes G_1)(I_{n_3} \otimes G_2)) G_3 G_4 X\right) (G_4 X)^T .$$

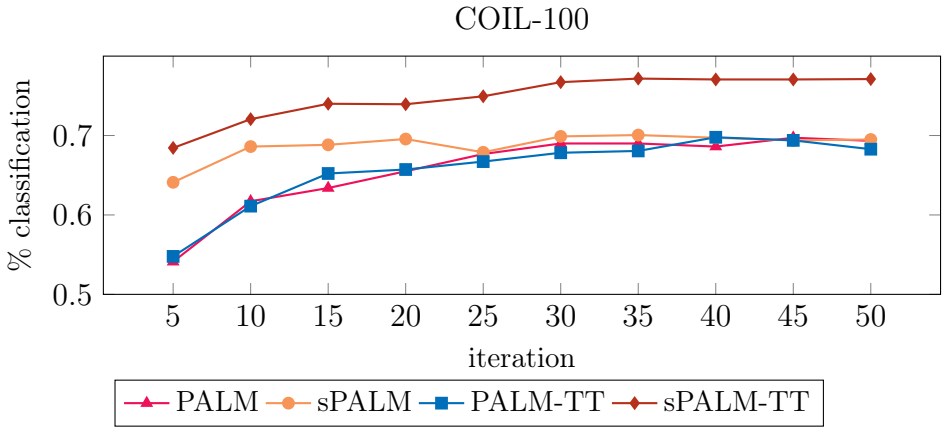


Figure 7.6: Classification performance of PALM-DL, sPALM-DL, PALM-DL-TT and sPALM-TT with respect to the number of iterations for the COIL-100 database.

For ∇_{G_4}, we have $\nabla_{G_4} H = 2C^T(-Y + CG_4X)X^T$, where $C = (I_{n_2n_3} \otimes G_1)(I_{n_3} \otimes G_2)G_3$. Using these expressions we follow the proof of Proposition 6.2.1 to obtain the required Lipschitz moduli. □

To test this formulation we consider the COIL-100 database [56], containing RGB images of $n_p = 100$ different objects in $n_e = 72$ view angles. For this experiment the size of each image is reduced to $16 \times 16 \times 3$ to preserve the relation among the number of pixels $n = n_1n_2n_3$, the atoms of the dictionary k and the total number of images n_en_p (i.e., $n < k < n_en_p$). The training set $\mathcal{Y}$ is composed by all the objects in $\tilde{n}_e = 54$ different view angles, corresponding to 75% of the total number of images. We set the number of atoms to $k = 1400$ and the sparsity parameter to $\tau = 4n_p$. The reported results correspond to the classification success along the iterations for a maximum of 50 iterations. From Figure 7.6 we can notice that the spectral step enhances the classification rate significantly for the TT formulation. In particular at the $50th$ iteration the classification rate of sPALM-TT is equal to 77.11% while the other methods do not reach 70%. Furthermore we observe that using a number of iterations greater than 30 has almost no impact on the classification rate.

7.3 Closing considerations

In this Chapter we presented two different classification models within the DL framework. In the first the classifier matrix is computed at the end of the minimization process while in the other it is learnt from data together with the dictionary and the sparse matrix. The numerical experiments on four different databases show that there is no advantage in using the latter model in terms of classification performance.

The numerical results in Section 7.2.2 highlight that the new sPALM algorithm yields faster convergence with respect to the other PALM-based algorithms due to the employment of second order information in the gradient stepsizes. Moreover, since the convergence of sPALM is guaranteed for a wide class of nonconvex and non-smooth problems, further constraints both on the dictionary and on the sparse matrix, such as non-negativity, can be set. Finally, we have experimentally shown that the TT formulation may give advantages in terms of memory requirements and rate of successful classification, especially when applied to 4D databases.

Conclusions

In this thesis we addressed the problem of image classification through the Tensor-Train decomposition with the aim of highlighting the advantages of the Tensor-Train approach with respect to other tensor decompositions when dealing with image recognition problems. Using a multidimensional approach is crucial in all image classification algorithms, since it allows to preserve the inner structure of the processed data such as the neighbouring relations among pixels that would be lost with vectorization processes.

In Chapter 2 we proposed a TT-based classification algorithm in which the database is decomposed using the Tensor-Train decomposition and the resulting TT-cores are used to classify a new image. In Chapter 7 we proposed a new Tensor-Train formulation of the Dictionary Learning problem and a new spectral Proximal Alternating Linearized Minimization algorithm (sPALM) that can be used for a large class of nonconvex and non-smooth problems. The numerical experiments reported in Section 7.2 show the advantage in using a Barzilai-Borwein stepsize in the Dictionary Learning context. In particular, employing second order information in the proximal gradient step results in better overall performance, i.e. less CPU time, faster convergence and higher success rate with respect to Lipschitz based stepsizes. Furthermore the combination with the Tensor-Train approach increases the classification performance especially in 4D examples and gives several advantages in terms of memory savings.

Thus, in both the presented models the Tensor-Train decomposition is able to reduce memory requirements while preserving the classification performance. It also allows to easily handle extensions to higher order settings as shown in Section 2.4.3 and Section 7.2.5,avoiding the problem of the curse of dimensionality affecting other tensor decompositions such as the HOSVD. This can be crucial when several features, such as illumination, view angle or backdrop image sets, need to be considered.

The properties of the TT decomposition explored in this thesis together with its low computational cost make it an extremely powerful tool whenever the number of parameters of a model needs to be reduced due to time or computational constraints. Thus, the use of the TT decomposition can be extended to many novel research fields known to be computationally demanding, such as deep learning.

Appendix A

Description of the databases

In this work we consider 10 different databases of images of n_p persons or objects in n_e expressions, where by expression we mean different illuminations, view angles, facial expression etc. The first 9 databases are composed by greyscale images each of which can be both considered either as a $n_1 n_2$ vector or as an $n_1 \times n_2$ matrix depending on the algorithm. The last database is composed by RGB images that can be represented by third-order tensors. These databases can provide a good benchmark for classification algorithms since they present significant differences in terms of persons or objects and "expressions". Furthermore, for the first four databases it holds that $n_i > n_e$ whereas for the others, $n_i < n_e$. In Chapter 2 this difference is crucial to explore the behaviour of different tensor methods. In Chapter 7 we only use the databases for which $n_i < n_e n_p$, since we consider an overcomplete DL problem. The characteristics of all databases are summarized in Table A.1.

1. **Orl** ([17]) contains 400 greyscale images in PGM format of 40 persons. Each subject is photographed in 10 different expressions. In Figure A.1 all the expressions of subject 1 are shown.

Figure A.1: Subject 1 of the Orl database.

2. **COIL-20** ([57]) is composed by greyscale images of 20 objects, each of which is photographed in 72 different view angles. In Figure A.2 object 1 in 15 different view angles is shown.

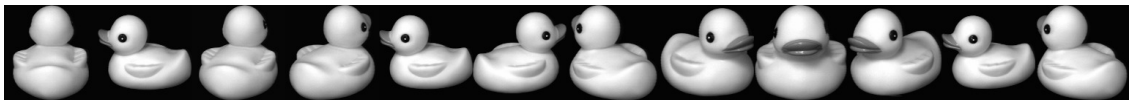


Figure A.2: Object 1 of the COIL-20 database.

3. **Faces95** ([72]) consists of 1440 RGB images in JPG format of 72 persons in slightly different positions with respect to the camera for a total of 20 expressions. RGB images were transformed in gray images within Matlab. The same was done for the other images with colors. In Figure A.3 the first subject is reported in all expressions.

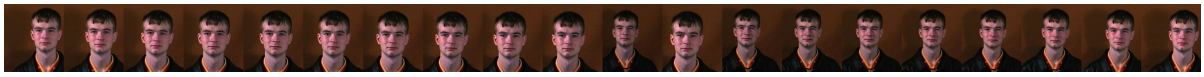


Figure A.3: Subject 1 of the Faces95 database photographed in 20 different expressions.

4. **Faces96** ([73]) is composed by 2261 images in JPG format of 119 persons at different distance with respect to the camera totalling 19 expressions. As for the previous database, the images were transformed in gray images. In Figure A.4 the first subject is reported in all the expressions.

Figure A.4: Subject 1 of the Faces96 database.

5. **MIT-CBCL**[1] ([80]) is composed by 3240 greyscale images of 10 persons in 324 different expressions. Each image is reduced to 15×15 pixels in order to have $n_i < n_e$. In Figure A.5, 10 different expressions of person one are shown.

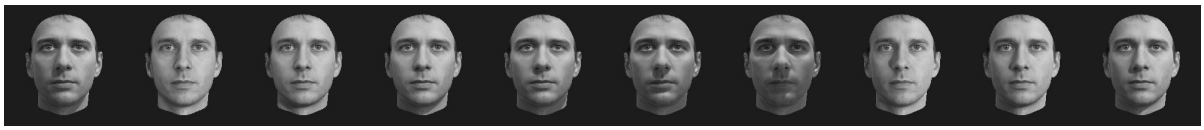


Figure A.5: Subject 1 of the MIT-CBCL database in 10 different expressions.

6. **Extended Yale** ([33]) consists of more than 16,000 images of 28 subjects in 585 expressions. Each image is reduced to 20×15 pixels in order to have $n_i < n_e$. In Figure A.6 subject 1 is reported in 15 different expressions.

[1]Copyright 2003 -2005 Massachusetts Institute of Technology. All Rights Reserved.

Figure A.6: Subject 1 of the Extended Yale database in 15 different expressions.

7. **MNIST** ([50]) contains 28×28 size images of ten handwritten digits (from 0 to 9) split in a training set composed by 60,000 images and a test set composed by 10,000 images. The number of "expressions" for each digit varies. In Table A.1 the minimum and maximum number of expressions is reported. In Figure A.7, 10 different versions of the digit "three" are shown.

Figure A.7: Digit 3 of the MNIST database.

8. **Fashion MNIST** ([82]) contains 70,000 images of 10 different kinds of Zalando's articles; 3,000 images per item for the training set and 1,000 for the test set were used. As for MNIST, the variable "expression" is not well-defined. Each database of n_p persons in n_e expressions is split into 75% training and 25% test sets, so that $\tilde{n}_e = 0.75 n_e$ is the total number of expressions used for training. In Figure A.8 the first object is represented in 10 different conditions.

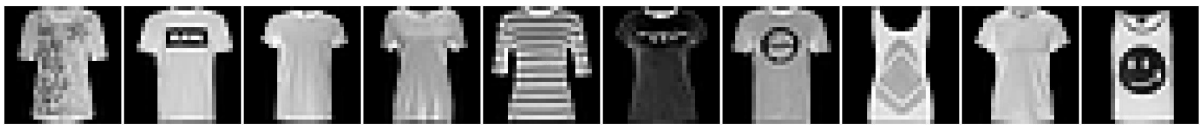


Figure A.8: First object of the Fashion MNIST database.

9. **Weizmann** [2] ([55]) face image database is composed by 28 subjects taken in 5 different view angles, 3 illumination conditions and 2 facial expressions. Each image size has been reduced from 352×512 to 14×20. In Figure A.9 subject 1 in different conditions is represented.

[2] downloaded on June 2018.

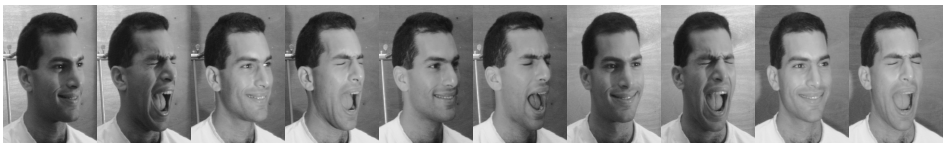


Figure A.9: Subject 1 of the Weizmann database.

10. **COIL-100** ([56]) is composed by RGB images of 100 objects in 72 different view angles. Figure A.10 shows an example of an object of the database in different view angles.

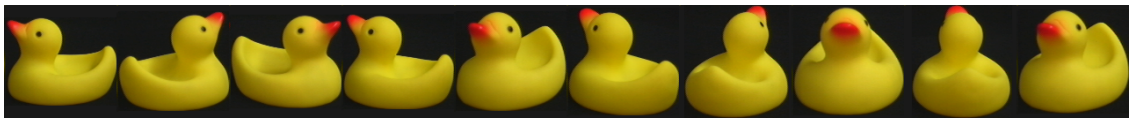


Figure A.10: Object 1 of the COIL-20 database.

Database	pixel size (original)	pixel size (resized)	n_e	n_p
Orl	92×112	92×112	10	40
COIL-20	128×128	128×128	72	20
Faces95	180×200	180×200	20	72
Faces96	196×196	196×196	19	119
MIT-CBCL	115×115	15×15	324	10
Ext'd Yale shrunk	640×480	20×15	585	28
MNIST	28×28	28×28	$892 - 6742$	10
Fashion MNIST	28×28	28×28	7000	10
Weizmann database	352×512	14×20	28	30
COIL-100	$128 \times 128 \times 3$	$16 \times 16 \times 3$	72	100

Table A.1: Pixel size, number of expressions and persons of all databases.

Bibliography

[1] M. Aharon, M. Elad, and A. Bruckstein. "K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation". In: *IEEE Trans. on Signal Processing* 54 (2006), pp. 4311–4322. ISSN: 1053-587X. DOI: 10.1109/tsp.2006.881199.

[2] C. Bao, H. Ji, Y. Quan, and Z. Shen. "L_0 Norm Based Dictionary Learning by Proximal Methods with Global Convergence". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*. IEEE Computer Society, 2014, pp. 3858–3865. DOI: 10.1109/CVPR.2014.493.

[3] J. Barzilai and J. M. Borwein. "Two-Point Step Size Gradient Methods". In: *IMA J. Numer. Anal.* 8.1 (Jan. 1988), pp. 141–148. ISSN: 0272-4979. DOI: 10.1093/imanum/8.1.141. eprint: https://academic.oup.com/imajna/article-pdf/8/1/141/2402762/8-1-141.pdf. URL: https://doi.org/10.1093/imanum/8.1.141.

[4] H. H. Bauschke and P. L. Combettes. *Convex analysis and monotone operator theory in Hilbert spaces*. Vol. 408. New York: Springer, 2011.

[5] A. Beck. *First-Order Methods in Optimization*. Philadelphia, PA, USA: SIAM-Society for Industrial and Applied Mathematics, 2017. ISBN: 1611974984.

[6] A. Beck and M. Teboulle. "A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems". In: *SIAM J. Imaging Sci.* 2.1 (2009), pp. 183–202. DOI: 10.1137/080716542.

[7] E. G. Birgin, J. M. Martínez, and M. Raydan. "Nonmonotone spectral projected gradient methods on convex sets". In: *SIAM J. Optim.* 10.4 (2000), pp. 1196–1211.

[8] E. G. Birgin, J. M. Martínez, and M. Raydan. "Spectral projected gradient methods: review and perspectives". In: *Journal of Statistical Software* 60.3 (2014), pp. 1–21.

[9] J. Bolte, S. Sabach, and M. Teboulle. "Proximal alternating linearized minimization for nonconvex and nonsmooth problems". In: *Math. Program.* 146.1-2 (2014), pp. 459–494. DOI: 10.1007/s10107-013-0701-9.

[10] S. Bonettini. "Inexact block coordinate descent methods with application to non-negative matrix factorization". In: *IMA J. Numer. Anal.* 31.4 (2011), pp. 1431–1452. DOI: 10.1093/imanum/drq024.

[11] S. Bonettini, I. Loris, F. Porta, and M. Prato. "Variable metric inexact line-search-based methods for nonsmooth optimization". In: *SIAM J. Optim.* 26.2 (2016), pp. 891–921.

[12] S. Bonettini, I. Loris, F. Porta, M. Prato, and S. Rebegoldi. "On the convergence of a linesearch based proximal-gradient method for nonconvex optimization". In: *Inverse Problems* 33.5 (2017), p. 055005. ISSN: 1361-6420. DOI: 10.1088/1361-6420/aa5bfd. URL: http://dx.doi.org/10.1088/1361-6420/aa5bfd.

[13] S. Bonettini, M. Prato, and S. Rebegoldi. "A cyclic block coordinate descent method with generalized gradient projections". In: *Applied Mathematics and Computation* 286 (2016), pp. 288–300. ISSN: 0096-3003. DOI: https://doi.org/10.1016/j.amc.2016.04.031. URL: https://www.sciencedirect.com/science/article/pii/S0096300316302818.

[14] M. Boussé, N. Vervliet, O. Debals, and L. De Lathauwer. "Face recognition as a Kronecker product equation". In: *2017 IEEE 7th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*. 2017, pp. 1–5. DOI: 10.1109/CAMSAP.2017.8313140.

[15] D. Brandoni, M. Porcelli, and V. Simoncini. *A Tensor-Train Dictionary Learning algorithm based on Spectral Proximal Alternating Linearized Minimization*. 2021. arXiv: 2107.11644 [math.NA].

[16] D. Brandoni and V. Simoncini. "Tensor-Train decomposition for image recognition". In: *Calcolo* 57 (2020).

[17] AT&T Laboratories Cambridge. *The ORL Database of Faces*. 2002.

[18] A. Cichocki. "Era of Big Data Processing: A New Approach via Tensor Networks and Tensor Decompositions". In: *CoRR* abs/1403.2048 (2014). arXiv: 1403.2048. URL: http://arxiv.org/abs/1403.2048.

[19] A. Cichocki, N. Lee, I. Oseledets, A. Phan, Q. Zhao, and D. P. Mandic. "Tensor Networks for Dimensionality Reduction and Large-scale Optimization: Part 1 Low-Rank Tensor Decompositions". In: 9 (2016), pp. 249–429. ISSN: 1935-8237. DOI: 10.1561/2200000059.

[20] A. Cichocki, D. P. Mandic, A. H. Phan, C. F. Caiafa, G. Zhou, Q. Zhao, and L. De Lathauwer. "Tensor Decompositions for Signal Processing Applications From Two-way to Multiway Component Analysis". In: *CoRR* abs/1403.4462 (2014). arXiv: 1403.4462. URL: http://arxiv.org/abs/1403.4462.

[21] C. F. Dantas, J. E. Cohen, and R. Gribonval. "Learning Fast Dictionaries for Sparse Representations Using Low-Rank Tensor Decompositions". In: *Latent Variable Analysis and Signal Separation - 14th International Conference, LVA/ICA 2018, Guildford, UK, July 2-5, 2018, Proceedings*. Ed. by Yannick Deville, Sharon Gannot, Russell Mason, Mark D. Plumbley, and Dominic Ward. Vol. 10891. Lecture Notes in Computer Science. Springer, 2018, pp. 456–466. DOI: 10.1007/978-3-319-93764-9_42. URL: https://doi.org/10.1007/978-3-319-93764-9_42.

[22] C. F. Dantas, J. E. Cohen, and R. Gribonval. "Learning Tensor-structured Dictionaries with Application to Hyperspectral Image Denoising". In: *27th European Signal Processing Conference, EUSIPCO 2019, A Coruña, Spain, September 2-6, 2019*. IEEE, 2019, pp. 1–5. DOI: 10.23919/EUSIPCO.2019.8902593.

[23] Cassio F. Dantas. "Accelerating sparse inverse problems using structured approximations". PhD Thesis. Université Rennes 1, 2019. URL: https://tel.archives-ouvertes.fr/tel-02494569.

[24] L. De Lathauwer, B. De Moor, and J. Vandewalle. "A multilinear singular value decomposition". In: *SIAM journal on Matrix Analysis and Applications* 21.4 (2000), pp. 1253–1278.

[25] G. Duan, H. Wang, Z. Liu, J. Deng, and Y. Chen. "K-CPD: Learning of overcomplete dictionaries for tensor sparse coding". In: *Proceedings of the 21st International Conference on Pattern Recognition, ICPR 2012, Tsukuba, Japan, November 11-15, 2012*. IEEE Computer Society, 2012, pp. 493–496. URL: http://ieeexplore.ieee.org/document/6460179/.

[26] B. Dumitrescu and P. Irofti. *Dictionary learning algorithms and applications*. Springer, 2018.

[27] M. Elad and M. Aharon. "Image Denoising Via Sparse and Redundant Representations Over Learned Dictionaries". In: *IEEE Trans. Image Process.* 15.12 (2006), pp. 3736–3745. DOI: 10.1109/TIP.2006.881969.

[28] L. Eldén. *Matrix Methods in Data Mining and Pattern Recognition (Fundamentals of Algorithms)*. USA: Society for Industrial and Applied Mathematics, 2007. ISBN: 0898716268.

[29] K. Engan, S.O. Aase, and J. Hakon Husoy. "Method of optimal directions for frame design". In: *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No.99CH36258)*. Vol. 5. 1999, 2443–2446 vol.5. DOI: 10.1109/ICASSP.1999.760624.

[30] Y. Fang, J. Wu, and B. Huang. "2D sparse signal recovery via 2D orthogonal matching pursuit". In: *Sci. China Inf. Sci.* 55.4 (2012), pp. 889–897. DOI: 10.1007/s11432-012-4551-5.

[31] G. Frassoldati, L. Zanni, and G. Zanghirati. "New adaptive stepsize selections in gradient methods". In: *Journal of Industrial & Management Optimization* 4.2 (2008), pp. 299–312.

[32] X. Gao, X. Cai, and D. Han. "A Gauss-Seidel type inertial proximal alternating linearized minimization for a class of nonconvex optimization problems". In: *J. Glob. Optim.* 76.4 (2020), pp. 863–887. DOI: 10.1007/s10898-019-00819-5.

[33] A.S. Georghiades, P.N. Belhumeur, and D.J. Kriegman. "From Few to Many: Illumination Cone Models for Face Recognition under Variable Lighting and Pose". In: *IEEE Trans. Pattern Anal. Mach. Intelligence* 23.6 (2001), pp. 643–660.

[34] M. Ghassemi, Z. Shakeri, W. U. Bajwa, and A. D. Sarwate. "Sample Complexity Bounds for Low-Separation-Rank Dictionary Learning". In: *2019 IEEE International Symposium on Information Theory (ISIT)*. 2019, pp. 2294–2298. DOI: 10.1109/ISIT.2019.8849698.

[35] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 2013. ISBN: 9781421407944. URL: https://books.google.it/books?id=X5YfsuCWpxMC.

[36] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[37] L. Grippo and M. Sciandrone. "Nonmonotone derivative-free methods for nonlinear equations". In: *Comput. Optim. Appl.* 37.3 (2007), pp. 297–328.

[38] L. Grippo and M Sciandrone. "On the convergence of the block nonlinear Gauss-Seidel method under convex constraints". In: *Operations Research Letters* 26 (Apr. 2000), pp. 127–136. DOI: 10.1016/S0167-6377(99)00074-7.

[39] N. Hao, M. E. Kilmer, K. Braman, and R. C. Hoover. "Facial recognition using tensor-tensor decompositions". In: *SIAM Journal on Imaging Sciences* 6.1 (2013), pp. 437–463.

[40] S. Hawe, M. Seibert, and M. Kleinsteuber. "Separable Dictionary Learning". In: Portland, OR, USA. Portland, OR, USA: IEEE, 2013, pp. 438–445. ISBN: 978-1-5386-5672-3. DOI: 10.1109/CVPR.2013.63.

[41] D. Hernandez and T. B. Brown. *Measuring the Algorithmic Efficiency of Neural Networks*. 2020. arXiv: 2005.04305 [cs.LG].

[42] L.T.K. Hien, D.N. Phan, and N. Gillis. "An Inertial Block Majorization Minimization Framework for Nonsmooth Nonconvex Optimization". In: *arXiv preprint arXiv:2010.12133* (2020). arXiv: 2010.12133 [math.OC].

[43] S. Hsieh, C. Lu, and S. Pei. "2D sparse dictionary learning via tensor decomposition". In: *2014 IEEE Global Conference on Signal and Information Processing, GlobalSIP 2014, Atlanta, GA, USA, December 3-5, 2014*. IEEE, 2014, pp. 492–496. DOI: 10.1109/GlobalSIP.2014.7032166.

[44] B. Iannazzo and M. Porcelli. "The Riemannian Barzilai–Borwein method with nonmonotone line search and the matrix geometric mean computation". In: *IMA J. Numer. Anal.* 38.1 (2018), pp. 495–517.

[45] M. Jouni. "Image Analysis Based on Tensor Representations". Theses. Université Grenoble Alpes [2020-....], Jan. 2021. URL: https://hal.archives-ouvertes.fr/tel-03223274.

[46] C. Kanzow and T. Lechner. "Globalized inexact proximal Newton-type methods for nonconvex composite functions". In: *Comput. Optim. Appl.* 78.2 (2021), pp. 377–410.

[47] T. G. Kolda. *Multilinear Operators for Higher-order Decompositions*. Tech. rep. SAND2006-2081. Sandia National Laboratories, 2006. DOI: 10.2172/923081. URL: http://www.osti.gov/scitech/biblio/923081.

[48] T. G. Kolda and B. W. Bader. "Tensor decompositions and applications". In: *SIAM review* 51.3 (2009), pp. 455–500.

[49] S. Krämer. "A Geometric Description of Feasible Singular Values in the Tensor Train Format". In: (Jan. 29, 2017). arXiv: http://arxiv.org/abs/1701.08437v2 [math.NA].

[50] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[51] S. Lesage, R. Gribonval, F. Bimbot, and L. Benaroya. "Learning unions of orthonormal bases with thresholded singular value decomposition". In: Philadelphia, PA, USA. Vol. 5. Philadelphia, PA, USA: IEEE, 2005, v/293–v/296 Vol. 5. ISBN: 0-7803-8874-7. DOI: 10.1109/ICASSP.2005.1416298.

[52] Z. Li, S. Ding, W. Chen, Z. Yang, and S. Xie. "Proximal Alternating Minimization for Analysis Dictionary Learning and Convergence Analysis". In: *IEEE Trans. on Emerging Topics in Computational Intelligence* 2.6 (2018), pp. 439–449. DOI: 10.1109/TETCI.2018.2806890.

[53] J. Mairal, F.R. Bach, and J. Ponce. "Sparse Modeling for Image and Vision Processing". In: *Found. Trends. Comput. Graph. Vis.* 8.2-3 (Dec. 2014), pp. 85–283. ISSN: 1572-2740. DOI: 10.1561/0600000058. URL: https://doi.org/10.1561/0600000058.

[54] E. Meli, B. Morini, M. Porcelli, and C. Sgattoni. "Solving nonlinear systems of equations via spectral residual methods: stepsize selection and applications". In: *arXiv preprint arXiv:2005.05851* (2020).

[55] Y. Moses. *Weizmann institute database*. 1997. URL: ftp.eris.weizmann.ac.il/pub/FaceBase.

[56] S. A. Nene, S. K. Nayar, and H. Murase. *Columbia Object Image Library (COIL-100)*. Tech. rep. CUCS-006-96. Department of Computer Science, Columbia University, 1996.

[57] S. A. Nene, S. K. Nayar, and H. Murase. *Columbia Object Image Library (COIL-20)*. Tech. rep. CUCS-005-96. Department of Computer Science, Columbia University, 1996.

[58] P. Ochs, Y. Chen, T. Brox, and T. Pock. *iPiano: Inertial Proximal Algorithm for Non-Convex Optimization*. 2014. arXiv: 1404.4805 [cs.CV].

[59] B. Olshausen and D. Field. "Emergence of simple-cell receptive field properties by learning a sparse code for natural images". In: *Nature* 381 (July 1996), pp. 607–9. DOI: 10.1038/381607a0.

[60] I. V. Oseledets. "Tensor-train decomposition". In: *SIAM Journal on Scientific Computing* 33.5 (2011), pp. 2295–2317.

[61] A. H. Phan, A. Cichocki, A. Uschmajew, P. Tichavský, G. Luta, and D. P. Mandic. "Tensor Networks for Latent Variable Analysis. Part I: Algorithms for Tensor Train Decomposition". In: *CoRR* abs/1609.09230 (2016). arXiv: 1609.09230. URL: http://arxiv.org/abs/1609.09230.

[62] T. Pock and S. Sabach. "Proximal Alternating Linearized Minimization (iPALM) for Nonconvex and Nonsmooth Problems". In: *SIAM J. Imaging Sci.* 9.4 (2016), pp. 1756–1787. DOI: 10.1137/16M1064064.

[63] M. Raydan. "On the Barzilai and Borwein choice of steplength for the gradient method". In: *IMA Journal of Numerical Analysis* 13.3 (July 1993), pp. 321–326. ISSN: 0272-4979. DOI: 10.1093/imanum/13.3.321. eprint: https://academic.oup.com/imajna/article-pdf/13/3/321/2076305/13-3-321.pdf. URL: https://doi.org/10.1093/imanum/13.3.321.

[64] M. Raydan. "The Barzilai and Borwein Gradient Method for the Large Scale Unconstrained Minimization Problem". In: *SIAM J. Optim.* 7.1 (1997), pp. 26–33. DOI: 10.1137/S1052623494266365.

[65] S. Rebegoldi, S. Bonettini, and M. Prato. "Application of cyclic block generalized gradient projection methods to poisson blind deconvolution". In: Aug. 2015, pp. 225–229. DOI: 10.1109/EUSIPCO.2015.7362378.

[66] F. Roemer, G. Del Galdo, and M. Haardt. "Tensor-based algorithms for learning multidimensional separable dictionaries". In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2014, pp. 3963–3967. DOI: 10.1109/ICASSP.2014.6854345.

[67] B. Savas and L. Eldén. "Handwritten digit classification using higher order singular value decomposition". In: *Pattern Recognition* 40.3 (2007), pp. 993 –1003. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2006.08.004. URL: http://www.sciencedirect.com/science/article/pii/S0031320306003542.

[68] D. di Serafino, V. Ruggiero, G. Toraldo, and L. Zanni. "On the steplength selection in gradient methods for unconstrained optimization". In: *Applied Mathematics and Computation* 318 (2018), pp. 176–195.

[69] Z. Shakeri, A. D. Sarwate, and W. U. Bajwa. "Identifiability of Kronecker-Structured Dictionaries for Tensor Data". In: *IEEE Journal of Selected Topics in Signal Processing* 12.5 (2018), pp. 1047–1062. DOI: 10.1109/JSTSP.2018.2838092.

[70] E. P. Simoncelli, W. T. Freeman, E. H. Adelson, and D. J. Heeger. "Shiftable multiscale transforms". In: *IEEE Transactions on Information Theory* 38 (2 1992), pp. 587–607. ISSN: 1557-9654. DOI: 10.1109/18.119725.

[71] M. Sokolova and G. Lapalme. "A Systematic Analysis of Performance Measures for Classification Tasks". In: *Inf. Process. Manage.* 45.4 (July 2009), 427–437. ISSN: 0306-4573. DOI: 10.1016/j.ipm.2009.03.002. URL: https://doi.org/10.1016/j.ipm.2009.03.002.

[72] L. Spacek. *Facial Images: Faces95*. 1995. URL: https://cmp.felk.cvut.cz/~spacelib/faces/faces95.html.

[73] L. Spacek. *Facial Images: Faces96*. 1996. URL: https://cmp.felk.cvut.cz/~spacelib/faces/faces96.html.

[74] B. L. Sturm and M. G. Christensen. "Comparison of orthogonal matching pursuit implementations". In: *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*. 2012, pp. 220–224.

[75] N. C. Thompson, K. Greenewald, K. Lee, and G. F. Manso. *The Computational Limits of Deep Learning*. 2020. arXiv: 2007.05558 [cs.LG].

[76] D. Tock. "Tensor Decomposition and its Applications". MA thesis. University of Chester, 2010.

[77] L. R. Tucker. "Some mathematical notes on three-mode factor analysis". In: *Psychometrika* 31.3 (1966), pp. 279–311. ISSN: 1860-0980. DOI: 10.1007/BF02289464. URL: https://doi.org/10.1007/BF02289464.

[78] M. A. O. Vasilescu and D. Terzopoulos. "Multilinear analysis of image ensembles: Tensorfaces". In: *European Conference on Computer Vision*. Springer. 2002, pp. 447–460.

[79] M. A. O. Vasilescu and D. Terzopoulos. "Multilinear image analysis for facial recognition". In: *Object recognition supported by user interaction for service robots*. Vol. 2. IEEE. 2002, pp. 511–514.

[80] B. Weyrauch, B. Heisele, J. Huang, and V. Blanz. "Component-Based Face Recognition with 3D Morphable Models". In: *2004 Conference on Computer Vision and Pattern Recognition Workshop*. 2004, pp. 85–85. DOI: 10.1109/CVPR.2004.315.

[81] S. J. Wright, R. D. Nowak, and M. A. T. Figueiredo. "Sparse Reconstruction by Separable Approximation". In: *IEEE Trans. on signal processing* 57.7 (2009), pp. 2479–2493. DOI: 10.1109/tsp.2009.2016892.

[82] H. Xiao, K. Rasul, and R. Vollgraf. "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms". In: *arXiv preprint arXiv:1708.07747* (Aug. 28, 2017). arXiv: cs.LG/1708.07747 [cs.LG].

[83] Q. Zhang and B. Li. "Discriminative K-SVD for dictionary learning in face recognition". In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2010, pp. 2691–2698. DOI: 10.1109/CVPR.2010.5539989.

[84] Z. Zhao, P. Zheng, S. Xu, and X. Wu. *Object Detection with Deep Learning: A Review*. 2019. arXiv: 1807.05511 [cs.CV].

[85] B. Zhou, L. Gao, and Y. Dai. "Gradient Methods with Adaptive Step-Sizes". In: *Computational Optimization and Applications* 35 (Sept. 2006), pp. 69–86. DOI: 10.1007/s10589-006-6446-0.

[86] H. Zhu and M.K. Ng. "Structured Dictionary Learning for Image Denoising Under Mixed Gaussian and Impulse Noise". In: *IEEE Trans. Image Process.* 29 (2020), pp. 6680–6693. DOI: 10.1109/TIP.2020.2992895.

[87] S. Zubair and W. Wang. "Tensor dictionary learning with sparse TUCKER decomposition". In: *2013 18th International Conference on Digital Signal Processing (DSP)*. 2013, pp. 1–6. DOI: 10.1109/ICDSP.2013.6622725.