Alma Mater Studiorum · Università di Bologna

SCHOOL OF SCIENCE
Master degree in Mathematics
Curriculum in Advanced Mathematics for Applications

LOW-RANK MATRIX FACTORIZATION: FROM LINEAR TO NONLINEAR MODELS

Master thesis in Numerical Analysis

Supervisor:
Chiar.ma Prof.
MARGHERITA PORCELLI

Co-supervisor:
Chiar.mo Prof.
NICOLAS GILLIS

Presented by:
GIOVANNI SERAGHITI

Accademic Year 2022-2023

Contents

Abstract

Il presente elaborato è la rielaborazione del lavoro condotto durante un tirocinio della durata di tre mesi che ha avuto luogo presso la Facultè Polytechnique di Mons in Belgio. In particolare, il tirocinio è stato svolto sotto la supervisione dei Professori Nicolas Gillis e Arnaud Vandaele del Dipartimento di Matematica e Ricerca Operativa dell'Università di Mons e dalla Professoressa Margherita Porcelli dell'Università di Bologna. La tesi tratta in dettaglio e amplia l'articolo

G. Seraghiti, A. Awari, A. Vandaele, M. Porcelli and N. Gillis, *Accelerated Algorithms for Nonlinear Matrix Decomposition with the ReLU function*, MLSP 2023, 17-20 September, 2023, Rome,

contenente i risultati ottenuti durante il tirocinio.

Lo stage si inserisce all'interno di un progetto europeo ERC[1] dell'Università di Mons, coordinato dal Prof. Nicolas Gillis. Il tema principale del progetto riguarda lo studio di modelli per la decomposizione di matrici mediante altre matrici di rango più basso. Tale problema è ricorrente in molte applicazioni tra cui la compressione di dati, la rimozione del rumore da immagini o segnali e nell'ambito del machine learning. Un esempio di tale decomposizione è la ben nota NMF (Nonnegative Matrix Factorization) che approssima la matrice contenente i dati con elementi non negativi.

La NMF sfrutta l'ipotesi di linearità tra le variabili del modello approssimante. Abbandonando questa ipotesi e inserendo una funzione non lineare nelle decomposizione di rango basso, è possibile formulare il cosiddetto problema NMD (Non linear Matrix Decomposition), che costituisce l'oggetto principale di questo elaborato. I modelli di approssimazione non lineare sono diventati ancora più popolari dopo l'esplosione di interesse nei confronti delle reti neurali. In molte di esse, infatti le funzioni di attivazione dei vari livelli della rete sono non lineari.

L'obiettivo principale del tirocinio è stato trovare nuovi algoritmi in grado di affrontare in modo efficiente il problema NMD, accelerando i metodi già esistente, come nel caso dell'*Accelerated NMD* (A-NMD), o proponendo nuove strategie risolutive basate sull'ottimizzazione alternata come per il *Three Blocks NMD* (3B-NMD).

La prima parte dell'elaborato tratta brevemente alcuni concetti che saranno utili nel

[1] `https://sites.google.com/site/nicolasgillis/projects/erc-cons`

seguito dell'eleborato come la minimizzazione alternata e introduce due dei più famosi metodi utilizzati nel contesto dell'approssimazione di matrici di rango basso: la SVD (Singular Value Decomposition) e la PCA (Principal Component Analysis).

Nel secondo capitolo, viene trattato il problema lineare della NMF, fornendo esempi di applicazioni, dando le motivazioni che rendono tale fattorizzazione tanto utilizzata e presentando i principali metodi risolutivi.

In seguito, viene introdotto il problema NMD cercando di fornire alcune chiavi di lettura e possibili interpretazioni intuitive che giustifichino l'utilità di tale decomposizione. Inoltre, vengono presentati i metodi già esistenti per risolvere questo problema e i due nuovi algoritmi: A-NMD e 3B-NMD.

Infine, vengono confrontati i vari metodi per la risoluzione del problema NMD su varie tipologie di dati. Vengono messi in evidenza i vantaggi dell'utilizzo dei nuovi algoritmi rispetto a quelli già esistenti, sia dal punto di vista dell'accuratezza della soluzione che sotto l'aspetto computazionale.

Introduction

One of the central problems in data analysis is extracting the underlying structure within data sets; in other words, we want to isolate meaningful information in a data set that can be used in several applications, such as data compression, noise filtering, reducing the computational effort for further manipulation of the data, or to directly identifying hidden structure in the data. In other words, we want to find a representation of our data in another space which has a lower dimension and that is easier to manipulate or to understand. Low-Rank Matrix Approximation (LRMA) arises in this context and aims to reduce the size of the original data set and to keep the highest possible amount of information. LRMA are essential in many fields of application in mathematics and computer science such as:

- numerical linear algebra,
- signal and image processing;
- graph theory;
- data analysis, and machine learning to perform regression, prediction, clustering, classification, and noise filtering.

What does it mean to compress information from a given data set $X \in \mathbb{R}^{m\times n}$, containing n observations each composed by m measurable variables? In few words, it means to look for r basis vectors of dimension m that give an easier and more meaningful representation of the data, where r is much smaller then the original number of samples n and the ambient dimension m, so $r \ll \min(m, n)$. In many cases, this concept translates in finding some factors $U \in \mathbb{R}^{m\times r}$ and $V \in \mathbb{R}^{r\times n}$ such that $X \approx UV$. In general, data compression is achieved when the number of entries needed to store the data is considerably reduced. In other words $r < \frac{mn}{m+n}$, since X has mn entries, while U and V have only $mr + nr$. When dealing with sparse matrices, we can define as $nnz(X)$ the number of nonzero entries and it is common use to require $r < \frac{nnz(X)}{m+n}$.

One of the possible techniques that may be used in this context is Linear Dimensionality Reduction (LDR). This class of methods requires a strong assumption which is *linearity* and it means that each of the variable of the model is related to the others with

a linear relation. This assumption simplifies consistently the models but it is not always consistent with real-world phenomena. LDR uses a small number of basis elements in order to represent each vector of the data set.

From a mathematical point of view, given a data set of n points $x_j \in \mathbb{R}^m$, LDR finds a small number $r < n$ of basis vectors $u_k \in \mathbb{R}^m$ such that each data point is well-approximated by a linear combination of these basis vectors, that is,

$$x_j \approx \sum_{k=1}^{r} u_k v_{kj} \quad \forall j$$

Introducing a matrix notation, the equivalence between LDR and LRMA model becomes clear

$$[x_1, x_2, \ldots, x_n] \approx [u_1, u_2, \ldots, u_r][v_1, v_2, \ldots, v_n],$$

equivalently

$$X \approx UV,$$

where

- $X \in \mathbb{R}^{m \times n}$, and each column is a data point $X(:, j) = x_j$;
- $U \in \mathbb{R}^{m \times r}$, is the matrix containing basis elements in the columns, that is, $U(:, j) = u_j$;
- $V \in \mathbb{R}^{r \times n}$ is the matrix of the coordinates of data points in the base U, $V(:, j) = v_j$.

Hence LDR provides a rank-r representation of the data set X in the basis U, using the coordinates contained in V, which means that for every column we have

$$x_j \approx U v_j \quad \forall j.$$

An illustrative example can be found in Figure 1.

LDR is a class of methods that comprises many different approaches. We will focus on Nonnegative Matrix Factorization (NMF), that is one of the possible LDR techniques and it requires the factors of the approximation U and V to be nonnegative. This constraint is crucial in order to give an immediate interpretation to the decomposition factors which highlights some important features of the original data.

What if the linearity assumption does not allow to catch the underlying structure of the data? In many real-world phenomena, the relation between the different variables is not linear. Think about phenomena involving rotations, or the evolution of prices in the market; in these cases, one is forced to introduce some nonlinear function in the approximation model and it happens also in data compression. Having this in mind, we will introduce Nonlinear Matrix Decomposition (NMD), which is a new field in matrix approximation that uses elementwise nonlinear functions to detect the hidden structure

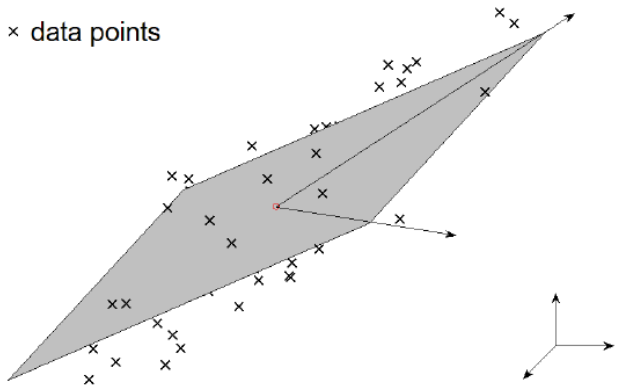


Figure 1: Approximation of three-dimensional data points with a two-dimensional subspace [1].

of the data which is recurrent both in the high-dimensional and low-dimensional representation of the data set. NMD was firstly introduced by Saul in 2022 [2, 3] and it is an open field of research. Furthermore, it is closely related to neural networks which use non-linear functions as activation functions of the different layers. In particular, the ReLU function defined as $f_{ReLU}(\cdot) = \max(0, \cdot)$ is the first choice for hidden layers in several neural networks. This connection motivates the matrix approximation model we will discuss in this work which is denoted ReLU-NMD and that wants to find a low-rank matrix Θ that approximates the original matrix X as $X = f_{ReLU}(\Theta) = \max(0, \Theta)$. In the following chapters we will introduce the problem, show some of the motivations behind it and present some of the algorithms that can be used to tackle the problem. In particular, we will focus on two new algorithms that improves the state-of-the-art methods to solve NMD: A-NMD and 3B-NMD. Those new methods were firstly presented in the paper

G. Seraghiti, A. Awari, A. Vandaele, M. Porcelli and N. Gillis, *Accelerated Algorithms for Nonlinear Matrix Decomposition with the ReLU function*, MLSP 2023, 17-20 September, 2023, Rome.

and their MATLAB implementation is publicly available at `https://gitlab.com/ngillis/ReLU-NMD`.

We will explain in detail the new algorithms for the NMD solution and they will be compared with the state-of-the-art methods. Several tests will be reported both on synthetically built and on real-world data sets.

Chapter 1

Background material

This chapter is devoted to the description of some preliminary optimization tools which will be useful in the next chapters of this thesis and will lay the groundwork for the solution of several low-rank matrix approximation models. In particular, we present the alternating optimization framework, which comprises optimization strategies that rely on exploiting block-structured problems, i.e, problems that can be described with two or more blocks of variables. The main advantage of a block decomposition method is that, when some variables are fixed, it is possible to obtain one or more subproblems of a special structure in the remaining variables. This can be useful in the solution of many optimization problems, especially when the structure of the subproblems can be conveniently exploited. Alternating minimization is used in many applications including compressed sensing, sparse dictionary learning, Nonnegative Matrix Factorization (NMF).

1.1 Alternating minimization

Let us introduce the block-structured optimization problem

$$\begin{aligned} &\min_x f(x) \\ &\text{subject to } x \in A = A_1 \times A_2 \times \cdots \times A_m \subset \mathbb{R}^n, \end{aligned} \tag{1.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a continuously differentiable function and the feasible set A is the Cartesian product of closed, nonempty and convex subsets $A_i \subset \mathbb{R}^{n_i}$, for $i = 1, \ldots, m$, with $\sum_{i=1}^m n_i = n$. Following the same structure, the vector $x \in \mathbb{R}^n$ is divided into m blocks $x = (x_1, x_2, \ldots, x_m)$. The family of algorithms called block-coordinate descent algorithms, is among the most known alternating minimization in the literature. They generate at each iteration a sequence of iterates $\{x^k\}$ such that

$$x^k = (x_1^k, x_2^k, \ldots, x_m^k).$$

Each iteration contains several inner steps, each of those update one individual component of x^k. Following the same notation, the function value $f(x)$ is also denoted as $f(x_1, x_2, \ldots, x_m)$ and for $i = 1, \ldots, m$ the partial gradient of f with respect to x_i, evaluated at x is indicated by $\nabla_i f(x) = \nabla_i f(x_1, x_2, \ldots, x_m) \in \mathbb{R}^{n_i}$.

We now introduce an algorithm that exploits the block structure of problem (1.1) and that allows to minimize each block of variables independently. The block coordinate descent (BCD) method for the solution of (1.1) is defined by the iteration:

$$x_i^{k+1} = \underset{y_i \in A_i}{\operatorname{argmin}} f(x_1^{k+1}, \ldots, x_{i-1}^{k+1}, y_i, x_{i+1}^k, \ldots, x_m^k). \tag{1.2}$$

In general the BCD method may not be convergent, meaning that it can produce a sequence $\{x^k\}$ with limit points that are not critical points of the problem [4]. Indeed

Definition 1.1. *A critical point of problem (1.1) is a point $\bar{x} \in A$ such that $\nabla f(\bar{x})^T(y - \bar{x}) \geq 0$, for every $y \in A$.*

Further hypothesis on the function f or on the constraint set A are needed in order to guarantee the convergence of the method. We also fix the notation for the partial updates of the the BCD method by defining the following vectors in A :

$$\begin{aligned} &w(k,0) = x^k, \\ &w(k,i) = (x_1^{k+1}, \ldots, x_{i-1}^{k+1}, x_i^{k+1}, x_{i+1}^k, \ldots, x_m^k), \quad i = 1, \ldots, m-1, \\ &w(k,m) = x^{k+1}. \end{aligned} \tag{1.3}$$

For convenience we also set $w(k, m+1) = w(k+1, 1)$.

1.1.1 Convergence analysis of the BCD method

We now present some properties of BCD that allow us to prove the convergence of the BCD when only two blocks are taken into account. Note that we do not suppose any further assumption on the objective function, which in particular, does not need to be convex.

A line search algorithm: we recall some properties of an Armijo-type line search algorithm along a feasible direction, which will be used in the sequel in the convergence proofs presented in [5]. Line search is a standard optimization techniques that allows to choose a proper step for several descent algorithms in order to prove the global convergence.

Let $\{z^k\}$ be a given sequence in A and suppose $z^k = (z_1^k, z_2^k, \ldots, z_m^k)$, with $z_i^k \in A_i$, for $i = 1, \ldots, m$. Assume that for all k, we can compute a search direction

$$d_i^k = w_i^k - z_i^k \quad \text{with } w_i^k \in A_i, \tag{1.4}$$

such that the following assumption holds:

Assumption 1 Let $\{d^k\}$ be the sequence of search directions from (1.4), then:

1. there exists a number $M > 0$ such that $\|d_i^k\| \leq M$ for all k;
2. we have $\nabla_i f(z^k)^T d_i^k < 0$ for all k.

Line search algorithm (LS algorithm) Set $\gamma_i \in (0,1), \delta_i \in (0,1)$. Compute

$$\alpha_i^k = \max_{j=0,1,\ldots} \{(\delta_i)^j : f(z_1^k, \ldots, z_i^k + (\delta_i)^j d_i^k, \ldots, z_m^k) \leq f(z^k) + \gamma_i (\delta_i)^j \nabla_i f(z^k)^T d_i^k\}. \quad (1.5)$$

Then the following proposition holds

Proposition 1.1.1. *Let $\{z^k\}$ be a sequence of points in A and let $\{d_i^k\}$ be a sequence of directions such that Assumption 1 is satisfied. Let α_i^k be computed by Algorithm LS, then:*

1. *there exists a finite integer j such that $\alpha_i^k = (\delta_i)^j$ satisfies the condition in (1.5);*
2. *if $\{z^k\}$ converges to $\bar{z}$ and*

$$\lim_{k\to\infty} f(z^k) - f(z_1^k, \ldots, z_i^k + \alpha_i^k d_i^k, \ldots, z_m^k) = 0, \quad (1.6)$$

then we have

$$\lim_{k\to\infty} \nabla_i f(z^k)^T d_i^k = 0. \quad (1.7)$$

Let us observe that from the update in (1.2), $w(k,i)$ defined in (1.3) is by construction the global minimizer of f in the $i-$th component subspace, and therefore it satisfies the necessary optimality condition:

$$\nabla_i f(w(k,i))^T (y_i - x_i^{k+1}) \geq 0 \quad \forall y_i \in A_i. \quad (1.8)$$

The following proposition describes the properties of the sequence generated by BCD and will allow us to guarantee the convergence of the algorithm in the 2-block case.

Proposition 1.1.2. *Suppose that for some $i \in \{0, \ldots, m\}$ the sequence $\{w(k,i)\}$ admits a limit point $\bar{w}$. Then, for every $j \in \{0, \ldots, m\}$ we have*

$$\lim_{k\to\infty} f(w(k,i)) = f(\bar{w}).$$

Proof. Let us consider an infinite subset $K \subset \{0, \dots, m\}$ and an index $i \in \{0, \dots, m\}$ such that the subsequence $\{w(k,i)\}_K$ converges to a point $\bar{w}$. By the update in (1.2) we have

$$f(w(k+1,i)) \leq f(w(k,i)). \tag{1.9}$$

Then the continuity of f and the convergence of $\{w(k,i)\}_K$ imply that the sequence $\{f(w(k,i))\}$ has a subsequence converging to $\{f(\bar{w})\}$. Since $\{f(w(k,i))\}$ is not increasing by (1.9), it is bounded from below and converges to $f(\bar{w})$. Then the assertion follows immediately from the fact that

$$f(w(k+1,i) \leq f(w(k+1,j) \leq f(w(k,i) \quad \text{for } 0 \leq j \leq 1,$$

and

$$f(w(k+2,i) \leq f(w(k+1,j) \leq f(w(k+1,i) \quad \text{for } i \leq j \leq m.$$

□

Proposition 1.1.3. *Suppose that for some $i \in \{1, \dots, m\}$ the sequence $\{w(k,i)\}$ admits a limit point $\bar{w}$. Then we have*

$$\nabla_i f(\bar{w})^T (y_i - \bar{w}_i) \geq 0 \quad \forall y_i \in A_i \tag{1.10}$$

and moreover

$$\nabla_{i^*} f(\bar{w})^T (y_{i^*} - \bar{w}_{i^*}) \geq 0 \quad \forall y_{i^*} \in A_{i^*}, \tag{1.11}$$

where $i^* = i (mod\ m) + 1$.

Proof. Let $\{w(k,i)\}_K$ be a sequence converging to $\bar{w}$. From the continuity of $\nabla_i f$ and from (1.8) we immediately get (1.10).

Let us prove (1.11), suppose at first $i \in \{1, \dots, m\}$, so that $i^* = i + 1$. Reasoning by contradiction, let us assume that there exists a vector $\tilde{y}_{i+1} \in A_{i+1}$ such that

$$\nabla_{i+1} f(\bar{w})^T (\tilde{y}_{i+1} - \bar{w}_{i+1}) < 0. \tag{1.12}$$

Then, letting

$$d^k_{i+1} = \tilde{y}_{i+1} - w(k,i)_{i+1} = \tilde{y}_{i+1} - x^k_{i+1}.$$

From the fact that $\{w(k,i)\}_K$ is convergent, we have that the sequence $\{d^k_{i+1}\}_K$ is bounded. Recalling (1.12) and taking into account the continuity assumption on $\nabla_i f$ it follows that there exists a subset $K_1 \subset K$ such that

$$\nabla_{i+1} f(w(k,i))^T d^k_{i+1} < 0 \quad \forall k \in K_1,$$

and therefore the sequence $\{d^k_{i+1}\}_{K_1}$ and $\{w(k,i)\}_{K_1}$ are such that the Assumption 1 holds, provided that we identify $\{z^k\}$ with $\{w(k,i)\}_{K_1}$.

Now, for all $k \in K_1$ suppose we compute α_{i+1}^k by means of Algorithm LS; then we have

$$f(x_1^{k+1}, \ldots, x_i^{k+1}, x_{i+1}^k + \alpha_{i+1}^k d_{i+1}^k, \ldots, x_m^k) \leq f(w(k,i)).$$

Moreover, as $x_{i+1}^k \in A_{i+1}$, $x_{i+1}^k + d_{i+1}^k \in A_{i+1}$, $\alpha_{i+1}^k \in (0,1]$, and A_{i+1} is convex, it follows that

$$x_{i+1}^k + \alpha_{i+1}^k d_{i+1}^k \in A_{i+1}.$$

In addition, recalling that

$$f(w(k,i+1)) = \min_{y_{i+1} \in X_{i+1}} f(x_1^{k+1}, \ldots, x_i^{k+1}, y_{i+1}, \ldots, x_m^k),$$

we can write

$$f(w(k,i+1)) \leq f(x_1^{k+1}, \ldots, x_i^{k+1}, x_{i+1}^k + \alpha_{i+1}^k d_{i+1}^k, \ldots, x_m^k) \leq f(w(k,i)). \tag{1.13}$$

By Proposition 1.1.3 we have that the sequences $\{f(w(k,j))\}$ are convergent to a unique limit for all $j \in \{0, \ldots, m\}$, amd hence we obtain

$$\lim_{k \to \infty, k \in K_1} f(w(k,i)) - f(x_1^{k+1}, \ldots, x_i^{k+1}, x_{i+1}^k + \alpha_{i+1}^k d_{i+1}^k, \ldots, x_m^k) = 0.$$

Then, using Proposition 1.1.1, where we identify $\{z^k\}$ with $\{w(k,i)\}_{K_1}$, it follows that

$$\nabla_{i+1} f(\bar{w})^T (\tilde{y}_{i+1} - \bar{w}_{i+1}) = 0,$$

which contradicts (1.12), so that we have proved that (1.10) holds when $i \in \{1, \ldots, m-1\}$. When $i = m$, so that $i^* = 1$, we can repeat the same reasoning, noting that we assumed $w(k, m+1) = w(k+1, 1)$ □

The above result implies, in particular, that every limit point of the sequence $\{x^k\}$ generated by BCD method is a critical point with respect to the components x_1 and x_m. This means that it holds the following corollary.

Corollary 1.1.1. *Let $\{x^k\}$ be the sequence generated by the BCD method and suppose there exists a limit point $\bar{x}$. Then we have*

$$\nabla_1 f(\bar{x})^T (y_1 - \bar{x}_1) \quad \forall y_1 \in A_1, \tag{1.14}$$

and

$$\nabla_m f(\bar{x})^T (y_m - \bar{x}_m) \quad \forall y_m \in A_m, \tag{1.15}$$

Let us now consider the 2-blocks problem:

$$\begin{aligned} &\min_x f(x) = f(x_1, x_2), \\ &\text{s.t. } x \in X_1 \times X_2. \end{aligned} \tag{1.16}$$

This type of problem is recurrent in many field of application and it may be useful to use a two-block decomposition since it may allow to employ parallel techniques for solving the subproblems. Furthermore in general the subproblems are way easier to solve than the original problem. In addition for the two block BCD a proof of convergence in the unconstrained case is given in [6]. Then the extension to the constrained 2Block BCD is an immediate consequence of Corollary 1.1.1.

Corollary 1.1.2. *Suppose that the sequence $\{x^k\}$ generated by the 2Block BCD method has limit points. Then every limit point $\bar{x}$ of $\{x^k\}$ is a critical point of (1.16)*

In addition, we also mention that adding some convexity hypothesis on the objective function, allows to prove the convergence of the general $m-$blocks BCD method. We will not include those results in this thesis but they can be found in [5].

1.1.2 A proximal point modification of BCD method (PBCD)

We will now present a more general BCD method including a proximal step and we will prove the convergence of the method without any assumption on the objective function [5], also in the general $m-$block case. This type of algorithm has some close connections with low-rank models for matrices, in particular with nonnegative matrix factorization [7].

Proximal algorithms are tools for solving convex, nonsmooth, constrained, optimization problems. In proximal algorithms, the base operation is evaluating the proximal operator of a function, which involves solving a convex optimization problem. This problem can be solved with standard methods, but it often admits closed form solutions or can be solved very quickly with simple specialized methods.

Definition 1.2 (Proximal operator)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a closed proper convex function, The proximal operator $prox_f : \mathbb{R}^n \to \mathbb{R}^n$ of f is defined by*

$$prox_f(y) = \operatorname*{argmin}_x \left(f(x) + \frac{1}{2}\|x - y\|_2^2 \right). \tag{1.17}$$

In the Proximal BCD (PBCD), we substitute the iteration in (1.1) with a proximal step, see Algorithm 1.

The following proposition shows that the limit points obtained by the PBCD methods are critical points of the problem (1.2).

Algorithm 1 PBCD method

1: Set $k = 0$, $x^0 \in A$, $\tau_i > 0$ for $i = 1, \dots, m$
2: **for** $i = 1, 2, \dots, m$ **do**
3: $\quad x_i^{k+1} = \arg\min_{y_i \in A_i} \{f(x_1^{k+1}, \dots, y_i, \dots, x_m^k) + \frac{1}{2}\tau_i \|y_i - x_i^k\|^2\}$.
4: **end for**
5: Set $x^{k+1} = (x_1^{k+1}, \dots, x_m^{k+1})$,
6: $k = k + 1$

Proposition 1.1.4. *Suppose that the PBCD method is well defined and that the sequence $\{x^k\}$ has limit points. Then every limit point $\bar{x}$ of $\{x^k\}$ is a critical point of problem (1.2).*

Proof. Let us assume that exists a subsequence $\{x^k\}_K$ converging to a point $\bar{x} \in A$. Define the vectors

$$\tilde{w}(k, 0) = x^k,$$
$$\tilde{w}(k, i) = (x_1^{k+1}, \dots, x_i^{k+1}, x_{i+1}^k, \dots, x_m^k) \quad \text{for } i = 1, \dots, m.$$

Then we have

$$f(\tilde{w}(k, i)) \leq f(\tilde{w}(k, i-1)) - \frac{1}{2}\tau_i \|\tilde{w}(k, i) - \tilde{w}(k, i-1)\|^2, \tag{1.18}$$

from which it follows

$$f(x^{k+1}) \leq f(\tilde{w}(k, i)) \leq f(\tilde{w}(k, i-1)) \leq f(x^k) \quad \text{for } i = 1, \dots, m. \tag{1.19}$$

Reasoning as in Proposition 1.1.2, we obtain

$$\lim_{k \to \infty} f(x^{k+1}) - f(x^k) = 0,$$

and hence, taking the limit in (1.18) for $k \to \infty$ we have

$$\lim_{k \to \infty} \|\tilde{w}(k, i) - \tilde{w}(k, i-1)\| = 0, \quad \text{for } i = 1, \dots, m, \tag{1.20}$$

which implies

$$\lim_{k \to \infty, k \in K} \tilde{w}(k, i) = \bar{x}, \quad i = 1, \dots, m. \tag{1.21}$$

Now, for every $j \in \{1, \dots, m\}$, as x_j^{k+1} is generated according to the rule in PBCD algorithm (Algorithm 1), the point

$$\tilde{w}(k, j) = (x_1^{k+1}, \dots, x_j^{k+1}, \dots, x_m^k),$$

satisfies the optimality condition

$$[\nabla_j f(\tilde{w}(k, j)) + \tau_j(\tilde{w}_j(k, j) - \tilde{w}_j(k, j-1))]^T (y_j - \tilde{w}_j(k, j)) \geq 0 \quad \forall y_j \in A_j. \tag{1.22}$$

Then, taking the limit for $k \to \infty, k \in K$, recalling (1.20) and (1.21) plus the continuity assumption on ∇f, for every $j \in \{1, \dots, m\}$ we obtain

$$\nabla_j f(\bar{x})^T (y_j - \bar{x}_j) \geq 0 \quad \forall y_j \in A_j,$$

which proves our assertion. □

1.1.3 Inertial Block Proximal method (IBP)

We now present a new efficient algorithm which includes a momentum step or equivalently an extrapolation step in the BCD scheme, see [8]. Indeed, momentum is a techniques which allows to use the information from previous iterations in a more clever way and it helps reaching faster convergence rate. It was introduced at first by Polyak in [9], where the descent direction of the algorithm is modified adding a momentum term, equal to the difference of the two previous iterates, the deriving point is called extrapolated point. Nesterov then, introduced the accelerated gradient method which evaluates the gradient at the extrapolated point. Both the approaches share the idea that, if the information from previous iterates is aggregated properly, it can produce a richer overall picture of the function than the standard iteration, and thus has the potential to yield better convergence. In the convex setting it is possible to prove that those techniques increase considerably the convergence rate of several algorithms, keeping the computational cost almost unchanged. In this section we apply extrapolation techniques to PBCD Algorithm 1.

Consider a particular instance of the problem in (1.1) which is a non-smooth, non convex optimization problem of the form

$$\min_{x \in E} F(x), \quad \text{where } F(x) := f(x) + r(x) \tag{1.23}$$

- $E = E_1 \times, \dots, E_s$ with E_i, being finite dimensional real linear spaces, equipped with a norm $\|\|_i$.
- $f : E \to \mathbb{R}$ is a continuous but possibly non-smooth, non-convex function.
- $r(x) = \sum_{i=1}^{s} r_i(x_i)$ with $r_i : E_i \to \mathbb{R} \cup \{\infty\}$, for $i = 1, \dots, s$, being proper and lower semi-continuous functions.

One of the key features of IBP algorithm is the fact that it allows to choose randomly or deterministically the block of variable to update; it was empirically observed that randomization leads to better solution and faster convergence [10]. For this reason, in IBP algorithm, Algorithm 2, includes an outer loop indexed by k and an inner loop, indexed by j. At each iteration j of the inner loop only one block $i \in \{1, \dots, s\}$ is updated. So the notation we are going to use for the j-th iteration within the k-th

outer loop of the algorithm is $x^{(k,j)}$, while the output, so the main generated sequenxe is denoted $\tilde{x}^{(k)}$. We also suppose that every block has to be updated before starting a new iteration of the outer loop. We will not go into the details on the choice of the parameters of the model. Note only that the momentum parameter $\alpha_i^{(k,j)}$ can either be fixed or it can be selected using more aggressive strategies, for example some adaptive techniques. Furthermore, the proximal parameter $\beta_i^{(k,j)}$ has to be tuned as well.

Algorithm 2 IBP method

Choose $\tilde{x}^(0)$ and initialize the needed parameters
for $k = 1, 2, \dots$ **do**
  $x^{(k,0)} = \tilde{x}^{(k-1)}$.
  for $j = 1, 2, \dots, T_k$ **do**
    Choose $i \in \{1, \dots, s\}$ deterministically or randomly. Let y_i be the value of the i-th block before it wes updated to $x_i^{(k,j-1)}$.
    $\hat{x}_i = x_i^{(k,j-1)} + \alpha_i^{(k,j)} \left(x_i^{(k,j-1)} - y_i \right)$,
    $x_i^{(k,j)} = \underset{x_i}{\operatorname{argmin}} F_i^{(k,j)}(x_i) + \frac{1}{2\beta_i^{(k,j)}} \|x_i - \hat{x}_i\|^2$.
  end for
  $\tilde{x}^{(k)} = x^{(k,T_k)}$.
end for

1.2 Standard matrix approximation models

In this section, we describe two of the methods which are commonly used in order to approximate a given matrix by another one of smaller rank. We start giving some general notions about Singular Value Decomposition (SVD) theory and we will see how it can be used to find the most meaningful information contained on a general data set. This section will be particularly useful since many of the algorithms that will be presented in the next chapters use SVD. We then introduce Principal Component Analysis (PCA), that computes the variance and covariance of a given set of measurements in order to detect the optimal set of variables to express the data in a more efficient and compact way. This method is in general, particularly useful for dimensionality reduction and denoising.

1.2.1 Singular Value Decomposition (SVD)

We now introduce the Singular Value Decomposition (SVD) of a given matrix $X \in \mathbb{R}^{m \times n}$. SVD has some advantages that makes it one of the best options in many low rank approximation models. At first, this decomposition always exists, also for rectangular

matrices and moreover, fixing the rank of the approximation to r, it easily provides the optimal rank-r approximation of X, meaning the rank-r matrix that has the smallest error with respect to the original one. The drawback of this approach is that computing the SVD is usually a difficult task and it requires a considerably high computational cost, compared to others low-rank models, so if the dimension of the problem is large it is not always possible to easily compute the SVD decomposition. Let us now state and prove the theorem that defines the SVD and ensures the existence of this decomposition. We start recalling a useful Lemma

Lemma 1.2.1. *Given a matrix $Q_1 \in \mathbb{R}^{m\times k}$, with orthonormal columns, there exists a matrix $Q_2 \in \mathbb{R}^{m\times(m-k)}$ such that $Q = (Q_1, Q_2)$ is an orthogonal matrix.*

Theorem 1.2.1. *Any matrix $A \in \mathbb{R}^{m\times n}$, with $m \geq n$, can be factorized as*

$$A = USV^T, \quad S = \begin{pmatrix} \Sigma \\ 0 \end{pmatrix}, \tag{1.24}$$

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

$$\Sigma = diag(\sigma_1, \sigma_2, \ldots, \sigma_n),$$

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

Proof. The assumption $m \geq n$ is not a restriction: in the other case just apply the theorem to A^T. Consider the maximization problem

$$\sup_{\|x\|_2=1} \|Ax\|_2.$$

Note that we are seeking for the supremum of a continuous function over a closed set, the supremum is attained for some vector x. Set $Ax = \sigma_1 y$, where $\|y\|_2 = 1$ and $\sigma_1 = \|A\|_2$.

Using Lemma 1.2.1 we can construct the orthogonal matrices

$$Z_1 = (y, \bar{Z}_2) \in \mathbb{R}^{m\times m}, \quad W_1 = (x, \bar{W}_2) \in \mathbb{R}^{n\times n}.$$

Then

$$Z_1^T A W_1 = \begin{pmatrix} \sigma_1 & y^T A \bar{W}_2 \\ 0 & \bar{Z}_2^T A \bar{W}_2 \end{pmatrix},$$

since $y^T Ax = \sigma_1$, and $Z_2^T Ax = \sigma_1 \bar{Z}_2^T y = 0$. Set

$$A_1 = Z_1^T A W_1 = \begin{pmatrix} \sigma_1 & w^T \\ 0 & B \end{pmatrix}.$$

Then

$$\frac{1}{\sigma_1^2 + w^T w} \|A_1 \begin{pmatrix} \sigma_1 \\ w \end{pmatrix}\|_2^2 = \frac{1}{\sigma_1^2 + w^T w} \|\begin{pmatrix} \sigma_1 + w^T w \\ Bw \end{pmatrix}\|_2^2 \geq \sigma_1^2 + w^T w.$$

But $\|A_1\|_2^2 = \|Z_1^T A W_1\|_2^2 = \sigma_1^2$; therefore $w = 0$ must hold. Thus we have taken one step toward a diagonalization of A. Proceed now by induction.

$$B = Z_2 \begin{pmatrix} \Sigma_2 \\ 0 \end{pmatrix} W_2, \quad \Sigma_2 = \mathrm{diag}(\sigma_2, \ldots, \sigma_n).$$

Then we have

$$A = Z_1 \begin{pmatrix} \sigma_1 & 0 \\ 0 & B \end{pmatrix} W_1^T = Z_1 \begin{pmatrix} \sigma_1 & 0 \\ 0 & Z_2 \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 \\ 0 & \Sigma_2 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & W_2^T \end{pmatrix} W_1^T.$$

Thus, by defining

$$U = Z_1 \begin{pmatrix} 1 & 0 \\ 0 & Z_2 \end{pmatrix}, \quad \Sigma = \begin{pmatrix} \sigma_1 & 0 \\ 0 & \Sigma_2 \end{pmatrix}, \quad V = W_1 \begin{pmatrix} 1 & 0 \\ 0 & W_2 \end{pmatrix},$$

The theorem is proved. □

The columns u_i, $i = 1, \ldots, n$ of U are called *left singular vectors*, while the ones, v_i, $i = 1, \ldots, m$ of V are denoted *right singular vectors*, and the diagonal elements of Σ, that are σ_i are the *singular values*.

Since $S = \begin{pmatrix} \Sigma \\ 0 \end{pmatrix}$, it is possible to obtain a more compact formulation of the SVD, exploiting the splitting $U = (U_1, U_2)$, where $U_1 \in \mathbb{R}^{m \times n}$ and the product with the lower part of S which contains only zero entries. Hence we get

$$A = U_1 \Sigma V^T \tag{1.25}$$

furthermore, from the expression in (1.25) can be derived the following matrix equations:

$$A v_i = \sigma_i u_i, \quad A^T u_i = \sigma_i v_i, \quad i = 1, 2, \ldots, n. \tag{1.26}$$

The equations in (1.26) suggest the so called *outer product formulation* of the SVD, that is an equivalent formulation obtained by a sum of matrices.

$$A = \sum_{i=1}^{n} \sigma_i u_i v_i^T. \tag{1.27}$$

Moreover, the following proposition holds

Proposition 1.2.1. *Let $A \in \mathbb{R}^{m \times n}$, then the 2-norm of A is given by*

$$\|A\|_2 = \sigma_1.$$

Proof. Without lost of generality, assume that $A \in \mathbb{R}^{m\times n}$ and $m \geq n$, and let the SVD of A be $A = U\Sigma V^T$. The norm is invariant under orthogonal transformations, therefore

$$\|A\|_2 = \|\Sigma\|_2.$$

The result now follows, since the 2-norm of a diagonal matrix is equal to the absolute value of the largest diagonal element. Recall that the diagonal elements of Σ are ordered from the largest to the lowest. Then, it holds:

$$\|\Sigma\|_2^2 = \sup_{\|y\|_2=1} \|\Sigma y\|_2^2 = \sup_{\|y\|_2=1} \sum_{i=1}^{n} \sigma_i^2 y_i^2 \leq \sigma_1^2 \sum_{i=1}^{n} y_i^2 = \sigma_1^2.$$

The equality is reached at $y = e_1$. □

Let us now give some useful definitions.

Definition 1.3 (Range)**.** *The range of a matrix A is the linear subspace,*

$$\mathcal{R}(A) = \{y \ / \ y = Ax, \text{ for arbitrary } x \ \}.$$

Definition 1.4 (Null-space)**.** *The null-space of a matrix A is the linear subspace*

$$\mathcal{N}(A) = \{x \ / \ Ax = 0 \ \}.$$

Assume that A has rank r, this implies that

$$\sigma_1 \geq \sigma_2, \ldots, \sigma_r > \sigma_{r+1} = \cdots = \sigma_n = 0.$$

Then, using the outer product form, we have

$$y = Ax = \sum_{i=1}^{r} \sigma_i u_i v_i^T x = \sum_{i=1}^{r} \sigma_i v_i^T x u_i = \sum_{i=1}^{r} \alpha_i u_i.$$

This means that the left singular values generate the Range of the matrix A.

Furthermore, since $Ax = \sum_{i=1}^{r} \sigma_i u_i v_i^T x$, we see that any vector $z = \sum_{i=r+1}^{n} \beta_i v_i$ is in the null-space of A. Having this observations in mind, we can state the following theorem:

Theorem 1.2.2. *(fundamental subspaces)*

1. *The singular vectors $u_1, u_2, \ldots, u_r$ are an orthonormal basis in $\mathcal{R}(A)$ and*

$$\text{rank}(A) = \dim(\mathcal{R}(A)) = r.$$

2. *The singular vectors $v_{r+1}, v_{r+1}, \ldots, v_n$ are an orthonormal basis in $\mathcal{N}(A)$ and*

$$\dim(\mathcal{N}(A))=n-r.$$

3. *The singular vectors* $v_1, v_2, \ldots, v_r$ *are an orthonormal basis in* $\mathcal{R}(A^T)$.

4. *The singular vectors* $u_{r+1}, u_{r+1}, \ldots, u_m$ *are an orthonormal basis in* $\mathcal{N}(A^T)$.

We now introduce the concept of *numerical rank* and its relation with the Truncated SVD (TSVD). Assume that A is a low-rank matrix plus noise: $A = A_0 + N$, with the noise N which is small compared to the original matrix A_0. In this situation, we call numerical rank the number of large singular values. In other words, those singular values which are above a fixed threshold and that with high probability contain the majority of the information about the matrix A_0, while the singular values small in absolute value are related to the noise N. If we know the rank of A_0, we can remove the noise finding a matrix of the correct rank, which approximates A. The obvious way to do this is simply to truncate the singular value expansion. Assume that the numerical rank is equal to k, then we approximate

$$A = \sum_{i=1}^{n} \sigma_i u_i v_i^T \approx \sum_{i=1}^{k} \sigma_i u_i v_i^T =: A_k. \tag{1.28}$$

The TSVD has several applications, for example it is widely used to stabilize the solution of problems which are extremely ill-conditioned or to remove noise. Furthermore, it can be applied in data compression, providing a lower rank approximation of a given matrix A. The popularity of the TSVD is due to the fact that it is the solution of approximation problems where one wants to approximate a given matrix by one of lower rank. The following theorems indeed hold.

Theorem 1.2.3. *Assume that the matrix* $A \in \mathbb{R}^{m\times n}$ *has rank* $r > k$. *The matrix approximation problem*

$$\min_{\text{rank}(Z)=k} \|A - Z\|_2$$

has the solution

$$Z = A_k := U_k \Sigma_k V_k^T,$$

where $U_k = (u_1, \ldots, u_k)$, $V_k = (v_1, \ldots, v_k)$, *and* $\Sigma_k = (\sigma_1, \ldots, \sigma_k)$. *The minimum is*

$$\|A - A_k\|_2 = \sigma_{k+1}.$$

A proof of this theorem can be found in [11].

Theorem 1.2.4. *Assume that the matrix* $A \in \mathbb{R}^{m\times n}$ *has rank* $r > k$. *The matrix approximation problem*

$$\min_{\text{rank}(Z)=k} \|A - Z\|_F$$

has the solution

$$Z = A_k := U_k \Sigma_k V_k^T,$$

where $U_k = (u_1, \ldots, u_k)$, $V_k = (v_1, \ldots, v_k)$, *and* $\Sigma_k = (\sigma_1, \ldots, \sigma_k)$. *The minimum is*

$$\|A - A_k\|_F = \left(\sum_{i=k+1}^{p} \sigma_i^2. \right)^{\frac{1}{2}},$$

where $p = \min(m, n)$.

In order to prove Theorem 1.2.4 we state at first the following lemma

Lemma 1.2.2. *Consider the* $mn-$*dimensional vector space* $\mathbb{R}^{m\times n}$ *with inner product*

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

and norm

$$\|A\|_F = \langle A, A \rangle^{\frac{1}{2}}.$$

Let $A \in \mathbb{R}^{m\times n}$, *with SVD* $A = U\Sigma V^T$. *Then the matrices*

$$u_i v_j^T, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n, \tag{1.29}$$

are an orthonormal basis in $\mathbb{R}^{m\times n}$.

We are now ready to prove Theorem 1.2.4

Proof. Write the matrix $Z \in \mathbb{R}^{m\times n}$ in terms of the basis in (1.29)

$$Z = \sum_{i,j} \psi_{ij} u_i v_j^T,$$

where the coefficients are to be chosen. For the purpose of this proof the coefficients of Σ are called σ_{ij}. Due to the orthonormality of the basis, and recalling that the matrix A can be written as in (1.28), we have

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

We can choose the second term to be zero. We then obtain the following expression for Z:

$$Z = \sum_{i} \psi_{ii} u_i v_i^T.$$

Since the rank of Z is equal to the number of terms in this sum, we see that the constraint rank(Z)=k, implies that we should have exactly k nonzero terms in the sum. To minimize the objective function we then choose

$$\psi_{ii} = \sigma_{ii} \quad i = 1, 2, \ldots, k,$$

which gives the desired result. □

1.3 Principal Component Analysis (PCA)

Assume to have a data set $X \in \mathbb{R}^{m\times n}$, which can be interpretated as a set of observations consisting of multiple measurements. The number of measurements is the dimension of each sample. In general, we suppose to have n observations, that is n vectors in an m-dimensional subspace, where m is the number of measurement types. The main goal of PCA is to find a better representation of the data, meaning that it looks for the most meaningful basis to re-express the data. This process can be very useful in order to remove noise from the data or to reduce redundant variables, finding a lower dimensional representation of the data.

How can we find this new base that better express the data set? The first assumption that PCA makes is *linearity*, which means that the new base is a linear combination of the previous base. This is a strong assumption that helps to simplify consistently the problem.

In particular, if X is the given data set, we look for a new representation Y such that they are related by a linear transformation P, that is

$$PX = Y. \tag{1.30}$$

Let us fix the notation: p_i are the rows of P, x_i are the columns of X and y_i are the columns of Y. Note that in equation (1.30) P is the matrix that transform X into Y and geometrically, it represents a rotation. The rows of P, $\{p_1, p_2, \dots, p_m\}$, are a set of new basis vectors that express the columns of X.

Writing the explicit product and defining the scalar product between two vectors a and b as $a \cdot b$ we have

$$Y = \begin{pmatrix} p_1 \cdot x_1 & \cdots & p_1 \cdot x_n \\ \vdots & \ddots & \vdots \\ p_m \cdot x_1 & \cdots & p_m \cdot x_n \end{pmatrix}.$$

This means that each column is of the form

$$y_i = \begin{pmatrix} p_1 \cdot x_i \\ \vdots \\ p_m \cdot x_i \end{pmatrix}.$$

In other words, the j-th coefficient of y_i is a projection on the j-th row of P. Therefore the rows of P are indeed a new set of basis vectors for representing the columns of X.

How do we choose the matrix P? The choice of the basis depends on the features that we want Y to express. Further assumptions beyond linearity are needed. Keep in mind that the main goal is to identify and reduce redundancies between individual variables, which are closely related to the concepts of variance and covariance. Consider two sets of simultaneous measurements with zero mean in vector form

$$a = (a_1, a_2, \dots, a_n), \quad b = (b_1, b_2, \dots, b_n).$$

The covariance can be expressed as a dot product matrix computation

$$\sigma_{ab}^2 = \frac{1}{n-1} ab^T. \tag{1.31}$$

Take now into account m vectors $x_1, x_1, \ldots, x_m$ and build the matrix

$$X = \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix},$$

then we can define the *covariance matrix*

$$S_X = \frac{1}{n-1} XX^T. \tag{1.32}$$

Note that S_X is a square $m \times m$ matrix, that contains the variance of the measurements in the diagonal and the off diagonal terms represent the covariance between measurement types. So computing S_X quantifies the correlations between all possible pairs of measurements. The idea is to obtain a transformation Y such that the covariance matrix S_Y satisfies certain properties. In particular, if we want to reduce redundancies, we want each variable to co-vary as little as possible with the others. In other words we want the off diagonal term to be zero, meaning that removing redundancies is equivalent to diagonalize S_Y.

1.3.1 Solving PCA

Let us make another assumption on the matrix P which is typical of PCA: assume P is an orthonormal matrix, that is $\{p_1, p_2, \ldots, p_m\}$ are an orthonormal basis. Furthermore we consider the directions p_i with larger variance as the most important and we call them *principal.*

We can formulate PCA problem as follows: find some orthonormal matrix P, where $Y = PX$ such that $S_Y = \frac{1}{n-1} YY^T$ is diagonalized. The rows of P are the principal components of X.

Let us write S_Y in terms of the variable P;

$$S_Y = \frac{1}{n-1} YY^T = \frac{1}{n-1} (PX)(PX)^T = \frac{1}{n-1} (PX)X^T P^T = \frac{1}{n-1} PAP^T. \tag{1.33}$$

Recall that $A = XX^T$ is a symmetric matrix and using the Spectral Theorem, it can be diagonalized by the orthogonal matrix of its eigenvectors. So we select the matrix P such that each row p_i is an eigenvector of XX^T. Following this reasoning, we find $A = P^T DP$, then, having in mind that $P^T = P^{-1}$, we have

$$S_Y = \frac{1}{n-1} PAP^T = \frac{1}{n-1} (PP^T) D (PP^T) = \frac{1}{n-1} D \tag{1.34}$$

S_Y is now diagonalized.

1.3.2 Relation with SVD

Let X be an arbitrary $m \times n$ matrix and $X^T X$ be a rank r, square, symmetric, $m \times m$ matrix. Let us now fix the notation:

- $\{\hat{v}_1, \hat{v}_2, \ldots, \hat{v}_r\}$ is the set of orthonormal $n \times 1$ eigenvectors with associated eigenvalues $\{\lambda_1, \lambda_2, \ldots, \lambda_r\}$ of the symmetric matrix $X^T X$

$$(X^T X)\hat{v}_i = \lambda_i \hat{v}_i,$$

- $\sigma_i = \sqrt{\lambda_i}$ are the singular values.
- $\{\hat{u}_1, \hat{u}_2, \ldots, \hat{u}_r\}$ is the set of orthonormal $m \times 1$ vectors defined by $\hat{u}_i = \frac{1}{\sigma_i} X\hat{v}_i$.

Note at first, that if λ_i are the eigenvalues of $X^T X$, it holds

$$(X\hat{v}_i)^T (X\hat{v}_j) = \hat{v}_i^T X^T X \hat{v}_j = \hat{v}_i^T (\lambda_j \hat{v}_j) = \lambda_j \delta_{ij}.$$

This means that

$$\|X\hat{v}_i\|^2 = (X\hat{v}_i)(X\hat{v}_i) = \lambda_i = \sigma_i^2.$$

From the definition of the vectors $\hat{u}_i$, we can restate the singular values decomposition as follows

$$X\hat{v}_i = \sigma_i \hat{u}_i. \tag{1.35}$$

We now construct the matrices which compose the singular value decomposition. Note that since the rank of the matrix is r we have to fill the set of $\{\hat{v}_1, \hat{v}_2, \ldots, \hat{v}_r\}$ and $\{\hat{u}_1, \hat{u}_2, \ldots, \hat{u}_r\}$ with $(n-r)$ and $(m-r)$ orthonormal vectors respectively. Hence we can define

$$V = [\hat{v}_1 \hat{v}_2 \ldots \hat{v}_n],$$
$$U = [\hat{v}_1 \hat{v}_2 \ldots \hat{v}_m],$$
$$\Sigma = [\sigma_1 \sigma_2 \ldots \sigma_r 0 \ldots 0].$$

Then we can write the expression in (1.35) in matrix form

$$XV = U\Sigma. \tag{1.36}$$

Manipulating equation (1.36) we get

$$X = U\Sigma V^T,$$
$$U^T X = \Sigma V^T,$$
$$U^T X = Z.$$

Note that the previous columns $\{\hat{u}_1, \hat{u}_2, \ldots, \hat{u}_r\}$ are now rows in U^T. Hence U^T is a change of basis from X to $Z = \Sigma V^T$ and it plays the same role of the matrix P in the

PCA. This means that U^T is a basis that spans the columns of X. Following the same reasoning we can obtain the following equation

$$V^T X^T = Z, \quad \text{where } Z = U^T \Sigma.$$

Again the columns of V (or the rows of V^T) are an orthonormal basis transforming X^T into Z, meaning that the basis spans the rows of X. It starts now to become clear that SVD and PCA are indeed closely related. Suppose to have an original matrix $m \times n$ X and define a new $n \times m$ matrix Y as follows

$$Y = \frac{1}{\sqrt{n-1}} X^T.$$

This definition becomes clear analyzing $Y^T Y$.

$$Y^T Y = \left(\frac{1}{\sqrt{n-1}} X^T \right)^T \left(\frac{1}{\sqrt{n-1}} X^T \right) = \frac{1}{n-1} X X^T = S_X.$$

By construction $Y^T Y$ is the covariance matrix of X and we have already seen that the principal components of X are the eigenvectors of S_X. So if we compute the SVD of Y, the columns of V contain the eigenvectors of $Y^T Y = S_X$. Therefore the columns of V are the principal components of X. This means that V spans the row space of $Y = \frac{1}{\sqrt{n-1}} X^T$, therefore V spans the column space of $\frac{1}{\sqrt{n-1}} X$.

In conclusion, finding the principal components of X is equivalent to find an orthonormal basis that spans the column space of X and the first singular values are the most meaningful ones.

Chapter 2

Nonnegative Matrix Factorization (NMF)

In this chapter we introduce Nonnegative Matrix Factorization (NMF) following the presentation in [1]. NMF became popular after the seminal paper of Lee and Seung in 1998 [12]. It is one of the most used techniques in Linear Dimensionality Reduction (LDR). It consists in solving a constrained optimization problem that allows to extract important features from a given data set.

In other words, given a matrix X, NMF finds two factors U and V which approximate $X \approx UV$, and in many cases, they can be easily interpretated since they are constrained to be elementwise nonnegative. From a mathematical point of view, given $X \in {\mathbb{R}_+}^{m \times n}$ and r, approximation rank, we want to solve the problem

$$\min_{U \geq 0, V \geq 0} \quad D(X, UV) \tag{2.1}$$

where $D(X, UV)$ measures the error between X and the approximation UV. The function D can be chosen in several ways and we will provide a brief overview based on maximum likelihood estimation. In many practical applications, the presence of noise does not always allow to obtain an exact decomposition $X = UV$, for this reason, the error measure $D(X, UV)$ is often needed.

We will at first introduce the problem and some of the applications where it can be found, providing some practical examples, such as features extraction in images and documents classification. We then present some of the algorithms that can be applied to solve the standard NMF problem. Most of the algorithms that will be shown use alternating minimization, which was introduced in Chapter 1.

2.1 Applications of NMF

Let us first introduce some of the possible applications of NMF. Given a matrix X, we want to understand when it is useful to decompose the matrix as $X \approx UV$, with $U, V \geq 0$. The main strength of NMF is the fact that its factors U, V are easily an intuitively interpretable.

2.1.1 Features extraction in a set of images

The first application and maybe the most recurrent one is features extraction. Suppose to have a set of vectorized grey-scale images. Vectorized means that the two-dimensional images are transformed into a long one-dimensional vector, for example, by stacking the columns of the image on top of each other. Collect all the vectors, columnwise in a matrix X. This means that the element X_{ij} contains the intensity of the i-th pixel of the j-th image. Note that each of the elements of X is nonnegative. Finding a solution of the general NMF problem in (2.1), means that we have two nonnegative matrices U and V such that $X \approx UV$. As mentioned in the previous section, the columns of U form a basis for the column of X. Since U is nonnegative, its columns can be interpretated as basis images, in other words, the columns of U are vectors of pixel intensities, whose linear combinations allow us to approximate each input image. Furthermore, also V is nonnegative and this means that all the coefficients in the linear combination are either positive or zero, so no cancellations are allowed. Hence, each of the basis image must correspond to a local feature that the original images share between each other. For example, if the columns of X contains facial images, the columns of U would be linked to eyes, noses, mustaches, and lips. The weighted sum of all the basis elements gives the approximation of each of the original images (see Figure 2.1 for a visual example). Another possible example is the features extraction of geometrical shapes. NMF can distinguish between the edges that compose a given picture as Figure 2.2 displays with the swimmer data set.

2.1.2 Text mining: topic modeling and document classification

Topic modeling is a type of statistical model which allows to detect the common "topics" that a collection of documents share. Depending on the topic that a certain document is about, one might expect some words to be more or less recurrent than others. The "topics" produced by topic modeling techniques are clusters of similar words. In other terms, topic modeling allows to detect the topic of each document from a collection based on the words that it contains. The connection with document clustering is straightforward, each document is contained in the cluster related to a certain topic. See [13] for additional information.

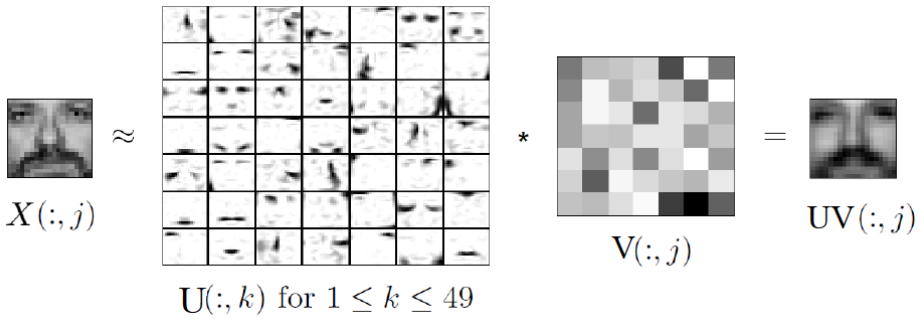


Figure 2.1: NMF applied on the CBCL face data set with $r = 49$ (2429 images with 19×19 pixels each). On the left is a column of X reshaped as an image. In the middle are the 49 columns of the basis U reshaped as images and displayed in a 7×7 grid and V contains the coefficients. UV is the NMF reconstruction [1].

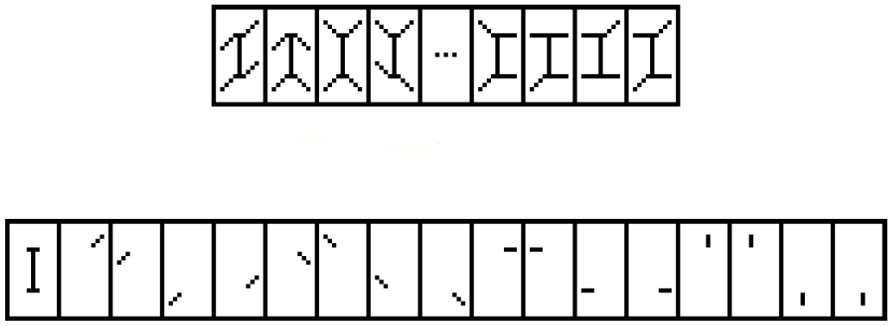


Figure 2.2: The top picture contains some sample images from swimmer data set, while the bottom one are the NMF basis elements[1].

From a mathematical point of view, suppose that each column of the matrix X corresponds to a document. This means that the columns of X are vectors of words count and they are also nonnegative. For example X_{ij} is the number of times the word i appears in the document j. This is the so-called *bag of words model.* Find NMF of the matrix X and obtain

$$X(:, j) \approx \sum_{k=1}^{r} U(:, k)V(k, j).$$

In this model, the columns of U are still nonnegative and they can be interpreted as vectors of words count as well. Note that the number of columns of U is much smaller than the columns of X, this means that the NMF has selected those words that better express the whole set of documents. In fact, no cancellation are allowed because V is nonnegative too, and hence each column of U must contain words that appear simultane-

ously in these documents. The most intuitive interpretation is that the words contained in each columns of U are all related to the same topic. Moreover, the columns of the factor V indicate the importance of the topics discussed in the corresponding documents. An intuitive illustration is given in Figure 2.3.

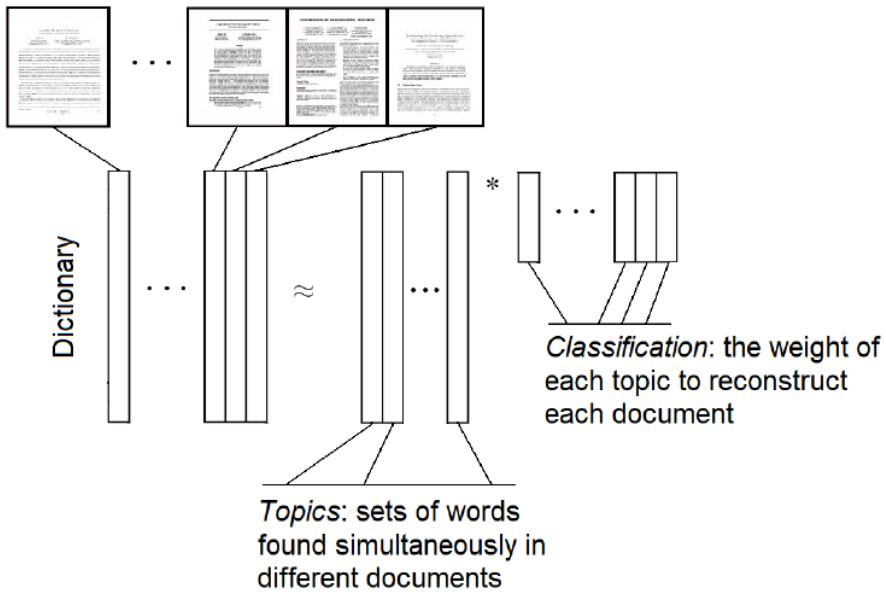


Figure 2.3: Illustration of the application of NMF to topic modelling, in which the basis elements in the decomposition represent the main topics of the set of documents.[1].

NMF can be applied to several other fields such as blind hyperspectral unmixing in which it allows to recover the different materials visible in an image only measuring the intensity of the lights within a scene. Moreover, it can be used to separate audio sources or in the analysis of chemical reactions or in geoscience and remote sensing. See [1] for additional explanations.

2.2 Error measure

When designing NMF model, the choice of the error measure $D(X, UV)$ in (2.1) is essential and it is used to evaluate the quality of the approximation. This quantity depends directly on the statistical properties of the noise added to the low-rank matrix, which, as it happens in the majority of real-life applications, it is unknown. Therefore, one has to choose which error measure to use relying on some strategy.

It is in our purpose only to present some of the possible error measures that can be included in the NMF model and to explain how they can be obtained as maximum

likelihood estimators of statistical distributions of the noise. Suppose the matrix X contains the observations of a random variable, $\tilde{X}_{ij}$, defined by the parameter $(\hat{U}\hat{V})_{ij}$, where the factors $\hat{U} > 0$ and $\hat{V} > 0$ are deterministic and unknown. The random variable $\tilde{X}$ depends on the parameter as $\tilde{X} = N((\hat{U}\hat{V})_{ij})$, where $N(\cdot)$ represents the noise and it can be either additive or multiplicative. In the following, we always suppose that the noise is independent and identically distributed (i.i.d.) and that the distribution of the noise is centered in $(\hat{U}\hat{V})_{ij}$ with variance σ.

Let us define the probability density function of $\tilde{X}_{ij}$ as $p(\tilde{X}_{ij}, (\hat{U}\hat{V})_{ij}, \sigma)$, then, since the noise is i.i.d. the likelihood of the sample X with respect to $(\hat{U}\hat{V})_{ij}$ and σ is

$$\ell(X; \hat{U}\hat{V}, \sigma) = \prod_{i,j} p(\tilde{X}_{ij}, (\hat{U}\hat{V})_{ij}, \sigma). \tag{2.2}$$

Given X, the parameters $\hat{U}$, $\hat{V}$, and σ can be estimated by solving the problem

$$\min_{U \geq 0, V \geq 0, \sigma} \ell(X; UV, \sigma).$$

In order to simplify the expression in (2.2), the logarithm of the likelihood is usually considered, multiplied by -1, and it is also possible to get rid of the parameter σ, which does not play a role in the estimation of $\hat{U}$ and $\hat{V}$. We then get an easier estimator of the form

$$\min_{U \geq 0, V \geq 0,} D(X, UV), \tag{2.3}$$

for some D, specified below.

Let us present some of the possible i.i.d. noise model. We do not explicitly derive the likelihood function which is out of the purposes of this thesis.

- *Gaussian*: let us suppose that the random variable $\hat{X}$ follows a model with additive noise $\hat{X} = \hat{U}\hat{V} + \hat{N}$, with $\hat{N}$ which is i.i.d. Gaussian with 0 mean and standard deviation σ. In this case, $\tilde{X}$ follows a Gaussian distribution,

  $$\tilde{X}_{ij} \sim \mathcal{N}\big((\hat{U}\hat{V})_{ij}, \sigma\big) \quad \forall i, j \text{ and some } \sigma.$$

  Then the logarithmic likelihood is

  $$D(X, UV) = \sum_{i,j} (X - UV)_{ij}^2 = \|X - UV\|_F^2. \tag{2.4}$$

- *Uniform*: each entry of $\tilde{X}$ follows the distribution

  $$\tilde{X}_{ij} \sim \mathcal{U}((\hat{U}\hat{V})_{ij}) - c, (\hat{U}\hat{V})_{ij} + c) \quad \forall i, j,$$

where $\mathcal{U}(a,b)$ is the uniform distribution in $[a,b]$. This is an additive noise, $\tilde{X} \sim \hat{U}\hat{V} + \tilde{N}$ with $\tilde{N}_{ij} \sim \mathcal{U}(-c,c)$ for all i,j. From the maximum likelihood estimation we get the following error measure

$$D(X,UV) = \max_{i,j} |X - UV|_{ij} = \|X - UV\|_\infty. \tag{2.5}$$

- *Laplace*: the entries of $\tilde{X}$ follow the Laplace distribution, whose probability density function is

$$p(\tilde{X}_{ij}, (\hat{U}\hat{V})_{ij}, \sigma) = \frac{1}{2\sigma} e^{-\frac{1}{\sigma}|\tilde{X}_{ij} - (\hat{U}\hat{V})_{ij}|} \quad \forall i,j.$$

  We then get the corresponding maximum likelihood estimator

$$D(X,UV) = \sum_{i,j} |X - UV|_{ij} = \|X - UV\|_1. \tag{2.6}$$

Since the property of the noise are in most case unknown, there are several techniques in order to choose a suitable error measure such as cross validation or other statistical approaches. In many cases the objective function is chosen empirically based on the type of data to handle. In our case, we will mostly deal with images and the Frobenius norm turns out to be particularly effective. Therefore, from now on we will focus on the model whose objective function is described in (2.4), and we will refer to it as FRO-NMF.

2.3 Algorithms for Frobenius NMF

In this subsection we present some algorithms that can be used to tackle the standard NMF problem (2.1).

We will focus on the particular case where the error measure D is the Frobenius norm, so problem (2.1) becomes

$$\min_{U \geq 0,\ V \geq 0} \quad \|X - UV\|_F^2. \tag{2.7}$$

In general, problem (2.7) is NP-hard (a proof of it can be found in [14]) and it is not possible to guarantee the converge to a global minimum. Hence all the algorithms that are mentioned in this section are iterative methods that attempt to attain convergence to a critical point of problem (2.7).

Alternating minimization scheme Most of the iterative schemes used to solve NMF problems are designed minimizing alternatively over the two factors U and V; in other words one minimizes U while keeping V fixed and viceversa. This is, in fact, a 2-block coordinate descent (2-BCD) method and it is described in Algorithm 3.

Let us observe the following properties:

Algorithm 3 Two-block coordinate descent framework

Input: nonnegative matrix X and factorization rank r.
Output: Two matrices U and V s.t. $X \approx UV$.
1: Generate initial matrices $U^0 \geq 0$ and $V^0 \geq 0$.
2: **for** $k = 1, 2, \ldots,$ maxit **do**
3: U^k=update(X, U^{k-1}, V^{k-1}), typically such that

$$\|X - U^k V^{k-1}\|_F \leq \|X - U^{k-1} V^{k-1}\|_F.$$

4: $(V^k)^T$=update$(X^T, (U^k)^T, (V^{k-1})^T)$, typically such that

$$\|X - U^k V^k\|_F \leq \|X - U^k V^{k-1}\|_F.$$

5: **end for**

1. For almost all the error measures D, including the Frobenius norm, alternating the minimization in U and V yields convex and easily solvable, either exactly or approximately, subproblems.

2. It holds true that $\|X - UV\|_F = \|X^T - V^T U^T\|_F$, so the FRO-NMF problem is symmetric in the variables U and V. We will focus on the following subproblem for V but everything can be applied also to the one in U:

$$\min_{V \geq 0} \|X - UV\|_F^2, \tag{2.8}$$

note also that fixing U the subproblem in V is a linear regression model.

3. Provided that it holds $\|X - UV\|_F^2 = \sum_{j=1}^n \|X(:, j) - UV(:, j)\|_2^2$, then solving (2.8) is equivalent to solve

$$\min_{V(:,j) \geq 0} \|X(:, j) - UV(:, j)\|_2^2 \quad j = 1, \ldots, n. \tag{2.9}$$

However, in practice, these problems are rarely solved independently.

First order optimality conditions We now state the first order optimality condition for NMF. In the general case, consider the error measure $D(X, UV)$, suppose it is differentiable and denote $\nabla_V D(X, UV) \in \mathbb{R}^{r \times n}$ the gradient of $D(X, UV)$ with respect to V. Note that $\nabla_V D(X, UV)$ is a matrix of the form

$$[\nabla_V D(X, UV)]_{k,j} = \frac{\partial D(X, UV)}{\partial V(k, j)}.$$

For the Frobenius norm we get

$$\nabla_V \|X - UV\|_F^2 = 2U^T(UV - X).$$

Most NMF algorithms are first-order methods, which means that only the gradient needs to be computed at each iteration and only convergence to a stationary point can be achieved. The point (U, V) satisfies the optimality condition of (2.7), also known as Karush–Kuhn–Tucker (KKT) conditions if

$$\begin{aligned} U \geq 0, \quad \nabla_U \|X - UV\|_F^2 \geq 0, \qquad \langle U, \nabla_U \|X - UV\|_F^2 \rangle = 0, \\ V \geq 0, \quad \nabla_V \|X - UV\|_F^2 \geq 0, \qquad \langle V, \nabla_V \|X - UV\|_F^2 \rangle = 0. \end{aligned} \tag{2.10}$$

Let us observe that from (2.10) , it can be shown that for the Frobenius norm, for any stationary point (U, V), it holds

$$\underset{\alpha}{\operatorname{argmin}} \|X - \alpha UV\|_F^2 = \frac{\langle X, UV \rangle}{\langle UV, UV \rangle} = 1,$$

from which we get

$$\|X - UV\|_F^2 = \|X\|_F^2 - 2\langle X, UV \rangle + \|UV\|_F^2 = \|X\|_F^2 - \|UV\|_F^2, \tag{2.11}$$

so at stationarity it holds $\|UV\|_F \leq \|X\|_F$.

We now present some algorithms which follows the 2-BCD framework of Algorithm 3 and that differ for the strategies used to solve subproblems for U and V. All these subproblems are Non Negative Least Square problems (NNLS) of the form

$$\min_{V \geq 0} \|X - UV\|_F^2. \tag{2.12}$$

2.3.1 Multiplicative Update (MU)

Before introducing the Multiplicative Update (MU) method, we review the rescaled gradient descent method for constrained optimization problem:

$$\min_{v \geq 0} f(v), \tag{2.13}$$

where f is twice continuously differentiable. The projected rescaled gradient updating step takes the form

$$v^+ = \mathcal{P}(\tilde{v} - B\nabla f(\tilde{v})), \tag{2.14}$$

where $\tilde{v}$ is the current iterate, v^+ is the next iterate, B is a diagonal matrix with positive diagonal entries and it can be interpreted as an approximation of the Hesian matrix of f in a Quasi-Newton method, and lastly, $\mathcal{P}(v) = \max(0, v)$ is the projection on the feasible set.

Let $\nabla_+ f(v) > 0$ and $\nabla_- f(v) > 0$ be such that $\nabla f(v) = \nabla_+ f(v) - \nabla_- f(v)$. Take $B = \text{diag}\left(\frac{[\tilde{v}]}{[\nabla_+ f(v)]}\right)$ and let $A \odot B$ be the elementwise product between matrices A and B, the projected rescaled gradient descent method becomes

$$v^+ = \tilde{v} - \frac{[\tilde{v}]}{[\nabla_+ f(v)]} \odot (\nabla_+ f(v) - \nabla_- f(v)) = \tilde{v} \odot \frac{[\nabla_- f(v)]}{[\nabla_+ f(v)]}. \tag{2.15}$$

This method can be easily applied to each block-update in Algorithm 3. As an example, for the FRO-NMF the explicit MU update takes the form

$$V \quad \leftarrow \quad V \odot \frac{[XV^T]}{[U^T U V]},$$

where we denote $\frac{[A]}{[B]}$ the componentwise division between the matrices A and B.

2.3.2 Alternating Nonnengative Least Square (ANLS)

The Alternating Nonnengative Least Square (ANLS) algorithm was first introduced by Paatero and Tapper in [15] where they solved the subproblems in U and V in Algorithm 3 up to global optimality. This approach is in fact a 2-BCD method and we state the following theorem which guarantees the convergence to a stationary point. In the general 2-BCD framework the convergence is guaranteed provided that the objective function in continuously differentiable and if each block of variables belongs to a closed, convex set, which is the case for FRO-NMF. Then it holds

Theorem 2.3.1. *[5] The limit points of the iterates of an exact 2-BCD algorithm applied to problem (2.7) are stationary points*

Let us simplify the problem considering the case in which X and V have only one column, denoted by x and v respectively and consider the problem:

$$\min_{v \geq 0} f(v), \quad \text{where } f(v) = \|x - Uv\|_F^2. \tag{2.16}$$

The KKT conditions become

$$v \geq 0, \quad \nabla_v f(v) = U^T(Uv - x) \text{ and } v^T \nabla_v f(v) = 0.$$

Note that the KKT conditions are necessary and sufficient for the global optimality since (2.16) is convex and it exists at least one point having $v > 0$, which means that the interior of the feasible domain is not empty.

One of the possible strategies that can be used to tackle the problem and to solve it with high accuracy is active-set method, see [16] for more details. Let us denote v^* an

optimal solution of the NonNegative Least Square (NNLS) problem in (2.16). Assume to know the set

$$\mathcal{I} = \{i | v_i^* > 0\},$$

The complement of $\mathcal{I}$ contains the indices such that $v_i^* = 0$ and it is called the active set.
Let us restrict the problem to $v(\mathcal{I})$, such that the unconstrained least square problem becomes

$$\min_{v(\mathcal{I})} \|U(:, \mathcal{I})v(\mathcal{I}) - x\|_2. \tag{2.17}$$

Solving the normal equations deriving from (2.17) we can compute the nonzero entries of v^*, hence we solve the linear system

$$\begin{aligned} [\nabla_v f(v)]_{\mathcal{I}} = 0 \quad &\iff [U^T(Uv - x)]_{\mathcal{I}} = 0 \\ &\iff U(:, \mathcal{I})^T U(:, \mathcal{I})v(\mathcal{I}) = U(:, \mathcal{I})^T x. \end{aligned}$$

This method is implemented in the MATLAB function `lsqnonneg`. Other methods can be used to solve NNLS, for example, second-order methods such as interior point method, can be used in order to reach high accuracy in the solution. The drawback of these alternative approaches is the fact that in general, this higher order methods are consistently more expensive and they can become not efficient if the dimension of the problem is large.

2.3.3 Alternating least square heuristic (ALS)

An heuristic approach that can be applied to solve NMF is conceptually simple and consists in solving the problem getting rid of the nonnegativity constraint and projecting the solution in the feasible set. Hence we compute the NNLS solution for the block Y in the following way

$$\max\left(0, \operatorname*{argmin}_Y \|X - UY\|_F^2\right).$$

The main advantage of this approach is that it is easy to implement and in some cases it provides reasonable solution, for example when the input matrix is sparse and can be well approximated using an elementwise operator such as the $\max(0, \cdot)$ function. However, it comes with no theoretical guarantees of convergence and in fact, it often diverges in practice, especially for dense input matrices.

2.3.4 Hierarchical Alternating Least Squares (HALS)

Hierarchical Alternating Least Squares (HALS) is an exact block coordinate descent method where the blocks of variables are the rows of V. Each of the NNLS subproblem

is then decomposed into smaller problems, one for each block. This is due to the fact that the variables on a single row of V are independent, in fact it holds

$$\|X - UV\|_F^2 = \sum_{i=1}^{n} \|X(:,j) - UV(:,j)\|_F^2. \tag{2.18}$$

Let us reformulate the univariate least squares problem as

$$\underset{v \in \mathbb{R}_+}{\operatorname{argmin}} \|x - uv\|_2^2 = \underset{v \in \mathbb{R}_+}{\operatorname{argmin}} \|x\|_2^2 - 2uv^T x + v^2 \|u\|_2^2. \tag{2.19}$$

The function in (2.19) is a quadratic with positive values and so convex function which has a unique minimizer of the form $\max(0, \frac{u^T x}{\|u\|_2^2})$.

Let us now consider only the ℓ-th row of V, while other variables are kept fixed:

$$\min_{V(\ell,:) \geq 0} \|X - \sum_{k \neq \ell} U(:,k)V(k,:) - U(:,\ell)V(\ell,:)\|_F^2 \tag{2.20}$$

Define

$$R_\ell = X - \sum_{k \neq \ell} U(:,k)V(k,:) = X - UV + U(:,\ell)V(:,\ell),$$

then (2.20) can be written as

$$\min_{V(\ell,:) \geq 0} \|(R_\ell - U(:,\ell)V(\ell,:))\|_F^2.$$

It holds

$$\|(R_\ell - U(:,\ell)V(\ell,:))\|_F^2 = \sum_{j=1}^{n} \|R_\ell(:,j) - U(:,\ell)V(\ell,j)\|_F^2,$$

which are n independent univariate least square problems with closed-form solution

$$\underset{V(\ell,j) \geq 0}{\operatorname{argmin}} \|X - UV\|_F^2 = \max\left(0, \frac{U(:,\ell)^T R_\ell(:,j)}{\|U(:,\ell)\|_2^2}\right) \quad \forall \ell, j.$$

In vector form we have

$$\underset{V(\ell,:) \geq 0}{\operatorname{argmin}} \|X - UV\|_F^2 = \max\left(0, \frac{U(:,\ell)^T R_\ell}{\|U(:,\ell)\|_2^2}\right) \quad \forall \ell.$$

HALS cyclically updates each row of V as above, and similarly for the columns of U; see Algorithm 4.

HALS is an exact Block coordinate descent (BCD) method with $2r$ blocks of variables updated cyclically. For a general BCD methods, in order to guarantee the convergence, the following theorem can be invoked and relies on four conditions.

Algorithm 4 Hierarchical alternating least squares (HALS)

Input: nonnegative matrix X and factorization rank r.
Output: Two matrices U and V s.t. $X \approx UV$.

1: Generate initial matrices $U^0 \geq 0$ and $V^{(0)} \geq 0$.
2: **for** $k = 1, 2, \ldots,$ maxit **do**
3: **for** $\ell = 1, 2, ..., r$ **do**
4: $V(\ell, :) \leftarrow \max\left(0, \frac{U(:,\ell)^T X - \sum_{k \neq \ell}(U(:,\ell)^T U(:,k))V(k,:)}{\|U(:,\ell)\|_2^2}\right)$
5: **end for**
6: **for** $\ell = 1, 2, ..., r$ **do**
7: $U(:, \ell) \leftarrow \max\left(0, \frac{XV(\ell,:)^T - \sum_{k \neq \ell} U(:,\ell)(V(\ell,:)V(\ell,:)^T)}{\|V(\ell,:)\|_2^2}\right)$
8: **end for**
9: **end for**

Theorem 2.3.2. *[17]The limit points of the iterates of an exact BCD algorithm to minimize a given objective function are stationary points provided that the following conditions hold:*

1. *the objective function is continuously differentiable,*
2. *each block of variables is required to belong to a closed convex set,*
3. *the minimum computed at each iteration for a given block of variables is uniquely attained,*
4. *the objective function values in the interval between all iterates and the next (which is obtained by updating a single block of variables) is monotonically decreasing.*

As we have already seen for the ALS algorithm, the first two conditions are satisfied by FRO-NMF in (2.7). Condition 3 and 4 are again satisfied since the algorithm for the solution of FRO-NMF has closed-form solution for its subproblem and each of the update monotonically decrease the objective function.

2.3.5 Accelerated HALS (A-HALS)

In all the first order methods, such as MU and HALS, it is crucial to tune properly the number of inner iterations when solving the NNLS subproblems in U and V. This aspect goes beyond the choice of the optimization method and it has to be taken into account in the designing of every 2-BCD for NMF. More in details, the main computational cost is in computing the gradient. Hence, to update V, we need to compute both $U^T X$ in $\mathcal{O}(nnz(X)r)$ and $(U^T U)V$ in $\mathcal{O}((m+n)r^2)$ operations. Note that for ANLS, since the subproblems are solved exactly, there is no need to choose the number of inner iterations.

The main idea in Alternating HALS (A-HALS) is to update several times the factor V, while keeping U fixed. In this way the computational cost of the alternating procedure is considerably decreased but we loose in terms of guarantees of convergence. Indeed, as explained above, the first gradient when updating V has a computational cost of $\mathcal{O}(\, nnz(X)r + (m+n)r^2)$ and it includes the computation of U^TX in in $\mathcal{O}(\, nnz(X)r)$ and U^TU in in $\mathcal{O}(mr^2)$. As long as U is not updated, those quantities may be stored and the next computation of the gradient in the subproblem for V requires only $\mathcal{O}(nr^2)$ from $(U^TU)V$. Therefore, the first computation of the gradient is much more expensive than the following ones, in particular it is $(1 + \frac{nnz(X)+mr}{nr})$ times more expensive. This reasoning suggests that updating V only once would be a waste of computation. Of course, it is not necessary to update V to many times since U will be modified at the next iteration, hence high accuracy in solving NNLS problem is not required.

The following heuristic works well in practice [7]:

- Perform at most $1 + \alpha\frac{nnz(X)+mr}{nr}$ updates of V, where $\alpha \in [0.5, 1]$,
- Add a stopping criteria based on the comparison between two successive iterations and the first update:

$$\|V^{(t)} - V^{(t-1)}\|_2 \leq \delta\|V^{(1)} - V^{(0)}\|_2,$$

  where $V^{(t)}$ is the current iterate of the NNLS problem and usually δ is fixed to 0.1.

2.3.6 Extrapolated NMF algorithm

As it usually happens in first order methods [18], one can consider the possibility to add momentum between the updates of U and V in the 2-BCD algorithm for NMF (Algorithm 3). Let us first introduce the general extrapolation scheme that will be used to accelerate NMF algorithms. Assume to have an optimization scheme such that the next iterate is computed only based on the previous one (e.g. gradient descent or a coordinate descent), that is

$$x_{k+1} = \text{ update}(x_k),$$

for some function update$(\cdot)$ that depends on the objective function and the feasible set. The idea is to define a new sequence of iterates y_k with $y_0 = x_0$ and modify the above scheme as follows

$$x_{k+1} = \text{ update}(x_k) \quad y_{k+1} = x_{k+1} + \beta_k(x_{k+1} - x_k),$$

where β_k is the momentum parameter at iteration k, and it can be either kept fixed or chosen adaptively.

Figure 2.4 allows to have an intuition of the meaning of the extrapolation scheme: the direction of $(x_{k+1} - x_k)$ is in between the direction obtained with the original update

applied to y_k and it allows to accelerate the convergence. For example, in gradient descent in smooth, convex optimization, it improves the convergence of the method from $\mathcal{O}(1\backslash k)$ to $\mathcal{O}(1\backslash k^2)$.

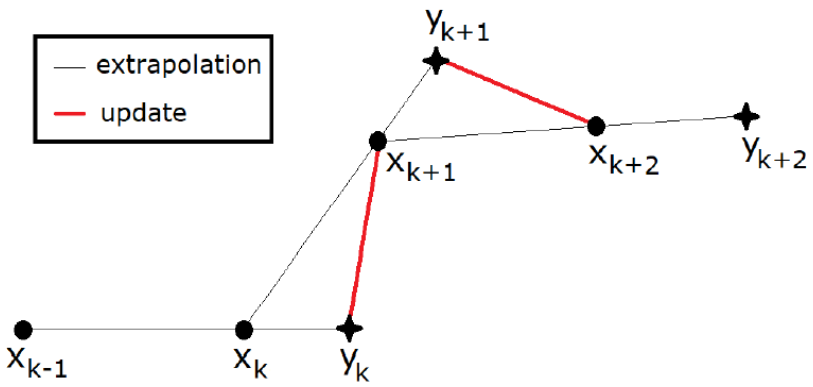


Figure 2.4: Illustration of extrapolation scheme[19].

The scheme can be applied to NMF as follows and it can be included in every method seen so far, in order to solve the NNLS subproblems. Given $Y^0 = V^0$ and $Z^0 = V^0$, compute for $t = 1, 2, \ldots$

$$\begin{aligned} U^t &= \text{update}(X, Y^{t-1}, Z^{t-1}), \\ Z^t &= Z^t(\beta_t) = U^t + \beta_t(U^t - U^{t-1}), \\ V^t &= \text{update}(X^T, (Z^t)^T, (Y^{t-1})^T), \\ Y^t &= Y^t(\beta_t) = V^t + \beta_t(V^t - V^{t-1}). \end{aligned}$$

Since NMF is nonconvex, tuning the parameter β_t is not trivial and it can be either kept fixed or various heuristic approaches can be applied [19]. In general, the extrapolation strategy increases the speed of convergence of the NMF algorithms as it can be seen for extrapolated A-HALS (E-A-HALS) in [1], but convergence to stationary points is not yet understood for such schemes. Furthermore, note that after the extrapolation step the factors are not guaranteed to be nonnegative, for this reason it is also possible to add a projection step of the form $Z^k = \max(0, Z^k)$ either at the end of the minimization or at each iteration; see [19] for further information.

Let us now present one of the possible heuristic that can be efficiently introduced in order to tune the parameter β_t in the extrapolation scheme. The first question to answer is: is it possible to have a closed-form expression for the best β parameter? The answer is yes, and it is the solution of the problem

$$\beta^* = \underset{\beta}{\text{argmin}} \|X - Z^t(\beta)Y^t|_F^2 \tag{2.21}$$

but empirically, evaluating β^* at each iteration does not work well because β^* is closed to zero for most steps of the procedure.

Hence in [19], Andersen proposes proposes a new strategy to choose β. It will increase the objective function in some cases $\|X - Z^t(\beta)Y^t|_F^2 > \|X - Z^t(0)Y^t|_F^2$, but it will allow a larger decrease at the next step. At first, fix the parameters $1 < \bar{\gamma} < \gamma < \eta$, $\beta_1 \in (0, 1)$. The heuristic update of β_k starts from an initial value $\beta_0 \in [0, \bar{\beta}]$, where $\bar{\beta} = 1$ is an upper bound. As long as the error decreases, increase the value of β_{k+1} by a factor γ, that is $\beta_{k+1} = \min(\gamma\beta_k, \bar{\beta})$. Increase also the upper bound by a factor $\bar{\gamma} < \gamma$ if it is smaller than one, hence $\bar{\beta} = \min(\gamma\bar{\beta}, 1)$. The usefulness of $\bar{\beta}$ is to keep in memory the last value of β_k that allowed the decrease of the objective function. In fact if the error increases, β_{k+1} is reduced by a factor $\eta > \gamma$ and the upper bound $\bar{\beta}$ is set to the previous value that allowed the decrease of the objective function, that is, β_{k-1}. See Algorithm 5 that summarize all the procedure.

Algorithm 5 Update of β_k

Input: $1 < \bar{\gamma} < \gamma < \eta$, $\beta_1 \in (0, 1)$.
Output: β_k parameter for the extrapolation step.

Set $\bar{\beta} = 1$.
if error decreases at iteration k **then**
    Increase β_{k+1} : $\beta_{k+1} = \min(\gamma\beta_k, \bar{\beta})$,
    Increase $\bar{\beta}$: $\bar{\beta} = \min(\bar{\gamma}\bar{\beta}, 1)$.
else
    Decrease β_{k+1} : $\beta_{k+1} = \beta_{k+1} \backslash \eta$,
    $\bar{\beta} = \beta_{k-1}$.
end if

2.3.7 The Inertial Block Proximal method for NMF

We present now the Inertial Block Proximal method (IBP) alternating minimization introduced in Section 1.1.3 applied to NMF problem. In particular we give an alternative formulation of the general NMF problem, which we recall here

$$\min_{V \geq 0, U \geq 0} \|X - UV\|_F^2. \tag{2.22}$$

We want to express it in an equivalent form that is similar to the general problem we considered to introduce IBP algorithm in (1.23), that is

$$\min_{x \in E} F(x), \quad \text{where } F(x) := f(x) + r(x). \tag{2.23}$$

In our case we consider $f(U, V) = \|X - UV\|_F^2$ and $r_1(U) = I_{\mathbb{R}_+^{m \times r}}(U)$, and $r_2(V) = I_{\mathbb{R}_+^{r \times n}}(V)$, where I_S is the indicator function of the set S. Furhtermore notice

that $UV = \sum_{i=1}^{r} U_{:i}V_{i:}$, hence NMF can be written as a function of $2 \times r$ variables $U_{:i}$ and $V_{i:}$. We recall that IBP algorithm includes a proximal step of the form

$$x_i^{(k,j)} = \underset{x_i}{\operatorname{argmin}} F_i^{(k,j)}(x_i) + \frac{1}{2\beta_i^{(k,j)}}\|x_i - \hat{x}_i\|^2,$$

where β_i is the proximal parameter. So we want to solve the subproblem with respect to the variable U, keeping V fixed, that is solving the problem

$$\underset{U:i\geq 0}{\operatorname{argmin}} \sum \|X - \sum_{q=1}^{i-1} U_{:q}V_{q:} - \sum_{q=i+1}^{r} U_{:q}V_{q:} - U_{:i}V_{i:}\|^2 + \frac{1}{2\beta_i}\|U_{:i} - \hat{U}_{:i}\|^2, \tag{2.24}$$

where $\hat{U}_{:i}$ comes from the extrapolation step, see Algorithm 2.

The solution of problem (2.24) is

$$\max\left(0, \frac{XV_{i:}^T - (UV)V_{i:}^T + U_{:i}V_{i:}V_{i:}^T + 1\backslash\beta_i\hat{U}_{:i}}{V_{i:}V_{i:}^T + 1\backslash\beta_i}\right). \tag{2.25}$$

As it happens for HALS, in order to reduce the computational cost, it is better to update each variable several times, before doing so for the other one.

Chapter 3

Nonlinear Matrix Decomposition (NMD)

Although it is widely used, LDR is not the unique approach that one can choose in order to perform dimensionality reduction of a given data set. In fact, depending on the type of data and on the relation that each variable has with the others, a linear estimator may not be the best possible choice and it is worthy to introduce a Nonlinear Matrix Decomposition (NMD). In this chapter we will follow the presentation in [2, 3]. We have seen in Chapter 2, that when dealing with nonnegative data, a given matrix X can be approximated by two nonnegative factors U and V, decreasing the rank of the original matrix and compressing the information. Other techniques may be used instead, for example, as mentioned in Chapter 1, one can compute the Singular Value Decomposition (SVD) of the matrix X, obtaining a lower rank approximation Θ. In many cases, such decompositions can help to analyze high-dimensional data in terms of a much smaller number of degrees of freedom.

In some practical applications and in general, when we want to describe physical processes, introducing nonlinearity is needed. For example, almost all neural networks uses non linear functions as activations in the different layers. Having this fact in mind, we introduce the NMD problem.

Given a sparse nonnegative matrix X, we want to estimate a low rank matrix Θ from which it is easy to recover the original one by applying an elementwise non linear function f. In general, we look for a matrix Θ such that

$$X \approx f(\Theta).$$

The form of this nonlinearity can be purposefully tailored to reveal low-dimensional structure in sparse data. Thanks to its close connection with neural networks, a common choice between nonlinearities is the ReLU function

$$f_{ReLU}(\cdot) = \max(0, \cdot). \tag{3.1}$$

In the following sections we will construct some nonlinear models suitable for sparse nonnegative matrices and we will present some iterative strategies that can be applied to solve them. *Sparse* nonnegative matrices arise in many fields of application. Widely recurrent in image processing where they can represent the edges of objects, they can also record links in a social networks or the word count in a corpus of documents. When those matrices are very large one is usually interested in finding a low-rank matrix, reducing the degrees of freedom, that does not match exactly the original one but still provides a close approximation.

3.1 Motivation and intuitive idea behind ReLU-NMD

Starting from a sparse nonnegative matrix X, the aim of the NMD is to find a new matrix Θ such that $X \approx f(\Theta)$ and f is an elementwise non linearity. From now on we will focus on the case where f is equal to the ReLU function in (3.1), commonly used in hidden layers of neural networks. Hence the problem consists in finding Θ such that the approximation

$$X \approx \max(0, \Theta),$$

is accurate up to a certain degree of freedom. In this section, we present the main ideas that motivate NMD models, the main differences with linear models for low-rank approximations, in particular with NMF, and the connection of NMD with neural networks.

3.1.1 Mining zeros of sparse data

The main and most intuitive idea that stands behind NMD and in particular behind dimensionality reduction using the ReLU function, is mining zeros of sparse data, as explained in [2]. In other words, the NMD model uses the ReLU function to map all the negative values of the matrix Θ into zeros of the original matrix X. In this way it is possible to strategically choose the negative values in Θ, so that the rank of the matrix is reduced, as shown in Figure 3.1. The idea is somehow similar to what usually happens in matrix completion, in which we construct an approximation of a given matrix by filling its missing entries in such a way that the rank is reduced. In both cases, the larger the number of missing elements, the more flexibility one has to complete the matrix with a low-rank model; this is the reason why the sparsity of the original matrix is an unavoidable hypothesis. As a consequence of this fact, if the matrix X contains mostly zero entries NMD has much more chances to discover low-rank decompositions than TSVD.

Let us consider the example in [2] to show how an elementwise nonlinearity can model large disparities in rank between the matrices Θ and X.

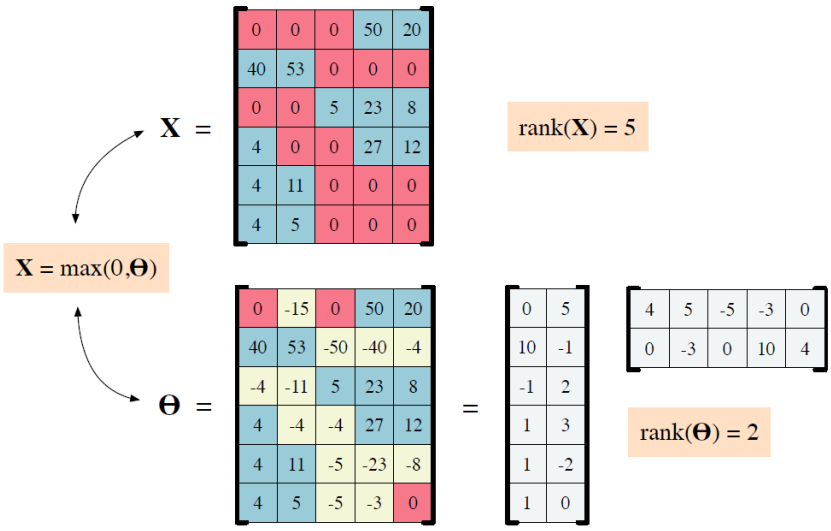


Figure 3.1: Example of mining zeros of sparse data in NMD approximation: X is a sparse nonnegative matrix and Θ is the NMD approximation such that $\max(0, \Theta) = X$. The idea is to replace zero values of X (shown in red) with strategically chosen negative values (shown in yellow). The more X is sparser and the more are the chances to lower the rank of Θ [2].

Theorem 3.1.1. *Let $\alpha > 0$, and consider the $n \times n$ circulant matrices with elements*

$$\Theta_{ij} = 1 - \alpha \left[1 - \cos \frac{2\pi}{n}(i - j) \right], \tag{3.2}$$

and

$$X_{ij} = \max(0, \Theta_{ij}). \tag{3.3}$$

It holds that X is full rank for $\alpha \geq \frac{1}{2 \sin \frac{\pi}{n} \sin \frac{\pi}{2n}}$, and rank($\Theta$)=3 for all $n \geq 3$.

At first we show that rank(Θ)=3. Let us define the orthogonal column vectors $c, s, u \in \mathbb{R}^n$, with elements

$$c_i = \cos \frac{2\pi i}{n}, \quad s_i = \sin \frac{2\pi i}{n}, \quad u_i = 1, \quad i = 1, \ldots, n. \tag{3.4}$$

These vectors are the eigenvectors of Θ with nonzero eigenvalues. This fact can be proved writing

$$\Theta = (1 - \alpha) u u^T + \alpha (c c^T + s s^T). \tag{3.5}$$

This decomposition establishes that $\text{rank}(\Theta) = 3$ for all $n \geq 0$. Note that this result holds for all values of $\alpha \notin \{0, 1\}$.

The second step is to prove that X has full rank. At first we show that the diagonal elements of X are all equal to one; then we show that the remaining (off diagonal) elements of X are either zero or sufficiently small that none of its eigenvalues can deviate too far from unity. To simplify the analysis we take into account a fixed value of α, but it is not difficult to show that the result also holds for all larger values of α. Let

$$\alpha \geq \frac{1}{2 \sin \frac{\pi}{n} \sin \frac{\pi}{2n}}, \tag{3.6}$$

substitute the value of α in (3.2), which after some calculations yields

$$\Theta_{ij} = 1 - \frac{\sin^2 \frac{\pi}{n}(i-j)}{\sin \frac{\pi}{n} \sin \frac{2\pi}{n}}. \tag{3.7}$$

Note that $\Sigma_{ii} = 1$, and hence from (3.3) it also holds $X_{ii} = 1$ for all terms in the diagonal. In addition, among the off-diagonal terms, we see that $\Sigma_{ij} < 0$ and hence $X_{ij} = 0$ unless $i - j = 1 \mod n$. Then we evaluate (3.7) for the elements just above or below the diagonal and we observe that each row of X has exactly two positive off-diagonal terms, bounded by

$$X_{i,i\pm1} = 1 - \frac{\sin \frac{\pi}{n}}{\sin \frac{2\pi}{n}} = 1 - \left(2 \cos \frac{\pi}{n}\right)^{-1} < \frac{1}{2}. \tag{3.8}$$

From this last result, we have shown not only that $X_{ii} = 1$ everywhere on the diagonal, but also that $\sum_{j \neq i} |X_{ij}| < 1$. It follows from the Gershgorin circle theorem that X has no zero eigenvalues and hence must be of full rank. The particular choice of α in (3.6) was convenient for this proof, but note that any larger choice pushes X even closer to the identity matrix (and its eigenvalues closer to unity). The theorem is then proved.

Furthermore, when α is very large, the matrix X reduces to the identity matrix. This means that the identity matrix of any size n can be approximated by a nonlinear model applied to a low-rank matrix of rank 3.

3.1.2 NMD vs NMF

As it has been explained in Chapter 2, given a nonnegative matrix X, the goal of NMF is to find a low-rank factorization $X \approx UV$, such that the factors U, V are constrained to be nonnegative. Such factors are usually discovered minimizing the error in Frobenius norm and in many cases, the minimization strategy consists in splitting the problem into two subproblems to be solved independently. The popularity of NMF is due both to the effectiveness in solving the subproblems using closed-form or multiplicative updates, and to the interpretability of its low-rank models. Thanks to the fact that the factor U and V are nonnegative, NMF is able to discover parts-based representation of data, meaning it can isolate all those different features, the data are made of. For example, NMF can

express some images of faces as a combination of the different facial features, such as eyes, nose and lips, which represent the basis elements of the linear decomposition. These representations are notably different than those discovered by SVD, whose factors do not have such an intuitive interpretation.

Even more evident is the difference in motivation between NMF and NMD. At first, NMD's approximation matrix Θ does not have a practical interpretation since it allows negative values. In this sense, NMD is more similar to PCA, in fact, both of the methods are based on the assumption that although the data are embedded in an high dimensional vector space, most of the variability is captured by much lower dimensional manifold[12]. NMD wants to detect those features that both the high-dimensional expression and his embedding in lower-dimensional manifold share, namely pattern manifold. While NMF seeks parts-based representation of data, NMD looks for pattern manifolds of the data. These are two types of low-dimensional structures that can exist in high-dimensional data. Furthermore, it is well known that not all pattern manifolds can be described by linear model and here it becomes essential to introduce nonlinearity in order to express the data in terms of its essential degrees of freedom. In conclusion, even though NMF can in some cases easily express relevant information in terms of much more easier factors, giving them an intuitive representation, it cannot reveal other types of low-dimensional structures.

Let us present an example of pattern manifold that can be revealed by the NMD model. This example is closely related to the construction shown in Theorem 3.1.1. In particular, we consider the case when a large value of the parameter α is chosen; hence, the original matrix X is close to the identity matrix of order n. Define

$$\Theta_{ij} = 1 - \alpha \left[1 - \cos \frac{2\pi}{n}(i-j) \right],$$

and

$$X_{ij} = \max(0, \Theta_{ij}).$$

As shown in Theorem 3.1.1, X has full rank, while rank(Θ)=3. This example arises naturally from a data set of sparse one-dimensional images, in which each image is a symmetric blip-three pixels wide, and darkest in its center pixel—against a light background, see Figure 3.2 in the left for an example of blip. All these images are related by simple translation, in fact, if we order each image based on the central pixel of the blips, it is clear that each successive image is obtained from the previous one, by shifting the pixels of one position. Moreover, the global data set is invariant under translation. This peculiarity is reflected also in the eigenvectors of the matrix Θ, which indeed turn out to be the simple Fourier modes. Let us point out that this structure is not obvious looking only at the original space of images, when viewed as vectors of pixel values, while it becomes evident looking at the eigenvectors of the low rank approximation. In addition the rank of the matrix Θ represents a dimensionality in which the data's underlying

manifold can be embedded. In conclusion, the data set X can be viewed as arising from a pattern manifold of one-dimensional translations, and the essential structure of this manifold is reflected in the much lower rank matrix Θ.

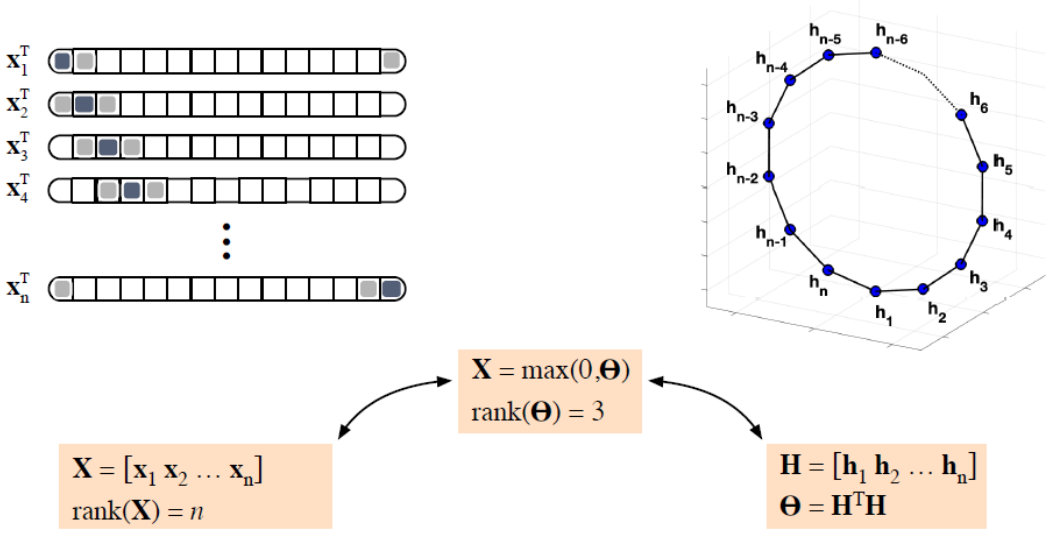


Figure 3.2: Example from [2], the matrix X contains a data set of sparse, one-dimensional images in which a blip is translated across a light background. The picture shows how the pattern manifold of X is reflected in the lower rank matrix Θ, such that $X = \max(0, \Theta)$. The structure of both the expression of the data is invariant under translation.

The idea of expressing data in terms of pattern manifolds in lower-dimensional spaces is becoming more and more popular since it has been associated to the the neural responses that underlie brain activity[20]. In particular, there may be a connection between lower-dimensional embedding and the way the brain perceives different stimuli from the same object, translations, rotations, different light exposure, etc. Let us give an intuitive idea of this connection. Suppose to have a collection M of images deriving from the same object but varying its orientation. Each image can be seen as a point in the Cartesian plane. The set M is a continuous curve in the image space and it can be described by one degree of freedom: the angle of rotation of the object. Even though the dimension of the image space M is equal to the number of images, it is a one dimensional manifold embedded in that higher-dimensional space. The retinal image is a collection of signals from photoreceptor cells. If this number are taken to be coordinates in an abstract image space, then an image is represented by a point. Hence, images from the same object will lay on the same manifold. Human brain is able to understand that a given image comes from the same object even though it is, for example rotated. This fact suggests that it can also equate between images coming from the same manifold and distinguish between

the images from different manifolds. How the brain represents image manifold is yet unknown.

3.1.3 NMD as neural network

The popularity of nonlinear models is increasing since neural networks became more and more relevant, especially in machine learning applications. Indeed most of the activation functions used in neural networks' layers are nonlinear. Consider a neural network with bottom layer of r hidden units, a top layer of d visible units and an $r \times n$ matrix W connecting nodes in these layers (see Figure 3.3). Assume $r < d$ such that the net is trying to explain a larger set of activities in the visible layer by a smaller one in the hidden layer. Mathematically, starting from n points of dimensionality d, $(x_1, x_2, \ldots, x_n)$, we want to estimate a weight matrix W and infer a corresponding set $(h_1, h_2, \ldots, h_n)$ of hidden patterns of dimension r, such that

$$x_i \approx f(Wh_i), \tag{3.9}$$

where f is an elementwise nonlinearity. The ReLU function for example, is commonly used in hidden layers of several nets. From equation (3.9) it is clear the connection of this type of models with the low-rank, nonlinear models that we are presenting in this thesis. Indeed, given a $d \times n$ data set X, and let the $r \times n$ matrix H represent the network's pattern, then (3.9) reduces to $X \approx f(\Theta)$, where $\Theta = WH$ is a matrix of rank r.

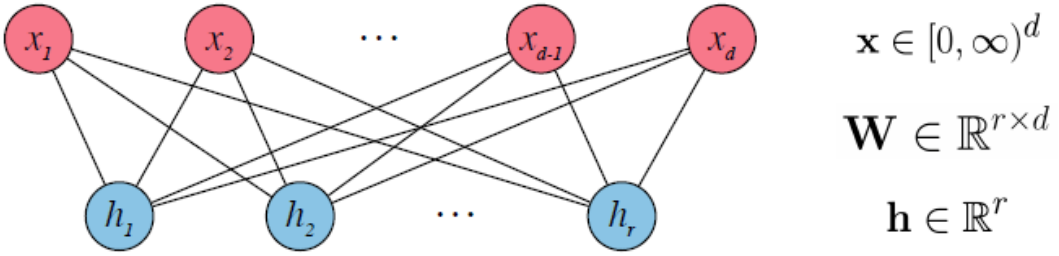


Figure 3.3: Two layers network in which the bottom layer encodes a lower-dimensional representation of the pattern of activities in the top layer, constrained to nonnegative values.

Chapter 4

Algorithms for ReLU-NMD

In this chapter we present the main algorithms that have been developed in order to solve NMD models. The NMD problem we consider in this thesis is the following: Given $X \in \mathbb{R}^{m\times n}$ and $r < \min(m, n)$, solve

$$\min_{\Theta\in\mathbb{R}^{m\times n}} \|X - \max(0, \Theta)\|_F^2 \quad \text{such that} \quad \operatorname{rank}(\Theta) = r. \tag{4.1}$$

We will refer to this problem as ReLU-NMD. ReLU-NMD makes sense only if X is nonnegative, since $\max(0, \Theta) \geq 0$. Moreover, X should be relatively sparse for ReLU-NMD to provide advantages compares to the TSVD: if X has mostly positive entries, the solution of ReLU-NMD will be similar to that of the TSVD, since Θ will need to contain mostly positive entries.

As far as we know there are no existing algorithms designed to tackle the problem directly. In fact, it is neither differentiable nor convex and the nonlinearity arising in the objective function in (4.1), makes the problem difficult to solve. ReLU-NMD has been recently investigated by Saul in [2], where he introduced another formulation for it, the so-called latent variable model. The main advantage of the latent variable formulation is that it allows one to move the nonlinearity from the objective function to the constraints, opening the possibility of exploring new solution strategies.

In this section, we quickly resume the general theory of latent variable models and we introduce expectation-minimization (EM) algorithm, which is commonly used to solve this type of problems. Then we formulate the latent variable model for NMD and we present some possible solution strategies: Naive-NMD algorithm, EM-NMD algorithm, introduced at first in [2], then the new methods A-NMD and 3B-NMD, which constitute the main contribution of this thesis, see [21].

4.1 The latent variable model theory

The word "latent" simply means "unobserved". Latent variables are random variables that we assume to exist underlying our data. In contrast with the usual approach of a regression model, in which one tries to learn a mapping from stimuli to responses, in latent variable models, the goal is to identify simplified structures in a set of observed responses. In the following we will consider x to be the observed data, and z to be the latent random variable. In particular, we will construct a general latent variable model in terms of two pieces:

- Prior probability over the latent: $z \sim p(z)$,
- Conditional probability of observed data $x|z \sim p(x|z)$

Starting from the quantities above, it is possible to derive the probability of the observed variable x, which is obtained integrating over the latent variable. This means that in the continuous case, we have:

$$p(x) = \int p(x|z)p(z)\, dz, \tag{4.2}$$

or summing in the case of discrete latent variables $z_1, z_2, \ldots, z_m$:

$$p(x) = \sum_{i=1}^{m} p(x|z = z_i)p(z = z_i), \quad z \in \{z_1, z_2, \ldots, z_m\}. \tag{4.3}$$

We now introduce two steps which are at the base of expectation-minimization (EM) algorithm, which is an iterative method for finding the maximum likelihood estimate for a latent variable model. It consists of iterating between two steps ("Expectation step" and "Maximization step", or "E-step" and "M-step" for short) until convergence.

1. **Inference step:** refers to the problem of inferring the latent variable z from the data x. The posterior over the latent given the data is specified by Bayes' rule:

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)}, \tag{4.4}$$

   recalling that the denominator is obtained integrating the numerator, in fact $p(x) = \int p(x|z)p(z)\, dz$. This is the so called E-step in which one tries to infer the distribution of the latent variable z. In practice, the standard way of proceeding is assuming z to have a certain distribution, for example Gaussian and in this phase, the statistics (mean and variance) of the random variable are estimated.

2. **Learning step:** in order to be precise we introduce the dependency to the parameters of the model, which we will call σ, and we write the model as specified by

$$p(x, z|\sigma) = p(x|z, \sigma)p(z|\sigma). \tag{4.5}$$

The learning step consists in updating the model parameters σ such that the likelihood is increased, so we maximize the marginal probability:

$$\hat{\sigma} = \arg\max_{\sigma} p(x|\sigma) = \arg\max_{\sigma} \int p(x, z|\sigma)\, dz. \tag{4.6}$$

In this way we perform the M-step which gives a new estimate of the model's parameters.

4.2 Latent NMD and Naive algorithm

We now introduce the latent variable Z, such that $\max(0, Z) = X$ that allows to bring the nonlinearity from the objective to the constraints in ReLU-NMD model. Note that Z is not constrained to be non negative. We directly write the latent model, in order to give some intuitive ideas about it and we will refer to it as Latent NMD. Latent NMD is formulated in [2] as follows

$$\min_{Z,\Theta} \|Z - \Theta\|_F^2 \quad \text{such that} \quad \begin{cases} \text{rank}(\Theta) = r, \\ \max(0, Z) = X. \end{cases} \tag{4.7}$$

Furthermore, observe that: first, the bottom constraint enforces that the elements of X can be perfectly recovered from those of Z, and second, the objective function in (4.7) is bounded below by zero and only obtains this minimum value when $X = max(0, \Theta)$. Let us now present some possible techniques that can be used to solve this latent NMD problem.

Naive algorithm A simple algorithm to tackle the reformulation (4.7) is based on alternating optimization [2], see Chapter 1. In other words, at each step, we minimize at first with respect to the variable Z keeping Θ fixed and then we use the new approximation of Z to update Θ. Given a sparse, nonnegative matrix X, let $I_+ = \{(i, j) \mid X_{ij} > 0\}$ and $I_0 = \{(i, j) \mid X_{ij} = 0\}$. At each iteration, Z and Θ are computed alternatively: the optimal solution for Z, when Θ is fixed in (4.7), is given by

$$Z_{ij} = \begin{cases} X_{ij} & \text{if} \quad (i, j) \in I_+, \\ \min(0, \Theta_{ij}) & \text{if} \quad (i, j) \in I_0. \end{cases} \tag{4.8}$$

The optimal solution for Θ when Z is fixed is the rank-r TSVD of Z. We will refer to this algorithm as Naive NMD, see Algorithm 6.

Algorithm 6 Naive NMD

Input: X, Z^0, Θ^0, r, maxit.
Output: A rank-r matrix Θ s.t. $X \approx \max(0, \Theta)$.

1: Set $Z_{ij}^k = X_{ij}$ for $(i,j) \in I_+$ and for $k = 0, 1$.
2: **for** $k = 0, 1, \ldots,$ maxit **do**
3: $\quad Z_{ij}^{k+1} = \min(0, \Theta_{ij}^k)$ for $(i,j) \in I_0$.
4: $\quad [U, D, V] = \text{TSVD}(Z^{k+1}, r)$.
5: $\quad \Theta^{k+1} = UDV^T$.
6: **end for**

Furthermore, in [3], Saul mentions that it is possible to accelerate the above simple scheme by an additional momentum term on the update of Z, with fixed Polyak momentum parameter [9]. Denoting by Z^{k+1}, the $k+1$-th iterate, after Z^{k+1} is computed via (4.8), it is updated as

$$Z^{k+1} \leftarrow Z^{k+1} + \alpha(Z^k - Z^{k-1}), \tag{4.9}$$

where $\alpha \in (0, 1)$. We will refer to the accelerated Naive algorithm as A-Naive, see Algorithm 7.

Algorithm 7 A-Naive NMD

Input: X, Z^0, Θ^0, r, maxit.
Output: A rank-r matrix Θ s.t. $X \approx \max(0, \Theta)$.

1: Set $Z_{ij}^k = X_{ij}$ for $(i,j) \in I_+$ and for $k = 0, 1$.
2: **for** $k = 0, 1, \ldots,$ maxit **do**
3: $\quad Z_{ij}^{k+1} = \min(0, \Theta_{ij}^k)$ for $(i,j) \in I_0$.
4: $\quad Z^{k+1} \leftarrow Z^{k+1} + \beta_k(Z^{k+1} - Z^k)$.
5: $\quad [U, D, V] = \text{TSVD}(Z^{k+1}, r)$.
6: $\quad \Theta^{k+1} = UDV^T$.
7: **end for**

Note that in both algorithms the TSVD must be computed. This can be easily done in MATLAB using the function `svd` for full matrices or `svds` for matrices in sparse format. In alternative it also possible to evaluate the eigenvectors and eigenvalues, of the matrix $(Z^{k+1})^T Z^{k+1}$, which correspond to the singular values. This task can be completed using the function `eig` in MATLAB. We observed that for small matrices and in particular as long as $(Z^{k+1})^T Z^{k+1}$ can be computed explicitly, `eig` is more efficient than `svd` or `svds`.

4.3 The EM-NMD algorithm

Let us now construct the expectation-minimization algorithm for NMD, in order to solve problem (4.7). Suppose to start from an original matrix X of dimension $m \times n$, where each m-dimensional column represents a single instance of the data set and n denotes the number of such examples. The idea is to consider a $m \times n$ matrix Θ, such that $\text{rank}(\Theta) = r$ and use it to parameterize a Gaussian latent variable model. In particular, given the matrix Θ, a distribution over the nonnegative matrices is generated. We sample a gaussian random variable of variance σ^2, namely Z_{ij} for each of the element Θ_{ij}

$$Z_{ij} \sim \mathcal{N}(\Theta_{ij}, \sigma^2). \tag{4.10}$$

Note that the model is parameterized by two parameters, Θ and σ^2 which as to be estimated in the M-step. Then we get the matrix X deterministically from the nonlinear map, in our case, from the ReLU function and we have

$$X_{ij} = \max(0, Z_{ij}). \tag{4.11}$$

The next step is to construct the likelihood of the data X, which, as shown in (2.2), depends elementwise on the marginal distribution $p(X_{ij}|\Theta_{ij}, \sigma^2)$. The goal is to maximize the likelihood with respect to the matrix Θ and to the variance σ^2. Let us first compute the marginal ditribution for an individual observed element by integrating over the value of its corresponding latent variable Z_{ij} that are consistent with the observation $X_{ij} = max(0, Z_{ij})$.

We have to take into account two possible cases: $X_{ij} = 0$ and $X_{ij} > 0$. Observe that for the zero elements of X, $max(0, z) = 0$ if and only if $z \leq 0$. Thus we have

$$p(X_{ij} = 0|\Theta_{ij}, \sigma^2) = \int_{-\infty}^{0} p(Z_{ij} = z|\Theta_{ij}, \sigma^2)\, dz = \Phi(-\Theta_{ij}\sigma^{-1}), \tag{4.12}$$

where $\Phi(\cdot)$ is the cumulative distribution function for a normal distribution with zero mean and unit variance.

For what it concerns non negative elements, since $X_{ij} = \max(0, Z_{ij})$, it is clear that $X_{ij} = Z_{ij}$ if and only if $Z_{ij} > 0$. Hence, for positive values of x, it holds

$$p(X_{ij} = x|\Theta_{ij}, \sigma^2) = p(Z_{ij} = x|\Theta_{ij}, \sigma^2), \tag{4.13}$$

where Z_{ij} is normally distributed by (4.10). We now observe that given the elements of Θ, the elements of X are conditionally independent, which means that we can write

$$\log p(X|\Theta, \sigma^2) = \sum_{ij} \log p(X_{ij}|\Theta_{ij}, \sigma^2). \tag{4.14}$$

Maximizing this sum we estimate the parameter of our model. Note that since we have the expression of $p(X_{ij} = 0|\Theta_{ij}, \sigma^2)$ and $p(X_{ij} = x|\Theta_{ij}, \sigma^2)$, we can proceed by applying the EM scheme, in order to solve the optimization problem of maximizing the expression in (4.14).

The EM algorithm consists in two phases, first the E-step or Inference in which we try to infer the distribution of the latent variable Z. In particular, since Z in our case, is gaussian distributed, we compute the posterior mean and variance. The second step is the M-step, in which we use the posterior statistics to reestimate the model's parameters Θ and σ^2.

Inference (E-step) At first, recall that to each observed matrix element X_{ij} is associated a latent variable Z_{ij}, whose posterior distribution is given by Bayes' rule as in (4.4)

$$p(Z_{ij}|X_{ij}, \Theta_{ij}, \sigma^2) = \frac{p(X_{ij}|Z_{ij}, \Theta_{ij}, \sigma^2)p(Z_{ij}|\Theta_{ij}, \sigma^2)}{p(X_{ij}|\Theta_{ij}, \sigma^2)}. \tag{4.15}$$

Note that the first term in the numerator, $p(X_{ij}|Z_{ij}, \Theta_{ij}, \sigma^2)$ is equal to one if $X_{ij} = \max(0, Z_{ij})$ and zero otherwise. Hence the posterior distribution in (4.15) splits in two cases:

- $X_{ij} = 0$, it is a right-truncated gaussian, that has no probability mass for the values of $Z_{ij} > 0$.
- $X_{ij} > 0$, the posterior distribution reduces to a Dirac delta centered at $Z_{ij} = X_{ij}$. This case is trivial.

At each iteration of the EM-NMD, in the E-step we compute the posterior mean and variance of Z_{ij} which are denoted by

$$\bar{Z_{ij}} = E[Z_{ij}|X_{ij}, \Theta_{ij}, \sigma^2], \tag{4.16}$$

$$\delta\bar{Z_{ij}}^2 = E[(Z_{ij} - \bar{Z_{ij}})^2|X_{ij}, \Theta_{ij}, \sigma^2], \tag{4.17}$$

respectively. In order to explicitly compute those statistics, we introduce some elementary functions, in particular let

$$\phi(z) = \frac{1}{\sqrt{2\pi}} e^{-z^2\backslash 2},$$

be the probability density function for a normal distribution with zero mean and unit variance, Φ, its cumulative distribution function, and $\psi(z) = \phi(z)\backslash\Phi(z)$ their ratio. Define $\gamma_{ij} = \sigma^{-1}\Theta_{ij}$, then we rewrite the likelihood and the statistics of the latent variables as follows; we do not include the explicit computation which is out of the purposes of this work.

- Likelihood
$$p(X_{ij} = 0|\Theta_{ij}, \sigma^2) = \Phi(-\gamma_{ij}),$$
$$p(X_{ij} = x|\Theta_{ij}, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x-\Theta_{ij})^2}.$$

- Posterior mean($\bar{Z_{ij}}$)
$$E[Z_{ij}|X_{ij} = 0, \Theta_{ij}, \sigma^2] = \Theta_{ij} - \sigma\psi(-\gamma_{ij}),$$
$$E[Z_{ij}|X_{ij} = x, \Theta_{ij}, \sigma^2] = x.$$

- Posterior variance ($\delta\bar{Z}_{ij}{}^2$)
$$Var[Z_{ij}|X_{ij} = 0, \Theta_{ij}, \sigma^2] = \sigma^2[1 + \gamma_{ij}\psi(-\gamma_{ij}) - \psi(-\gamma_{ij})^2],$$
$$Var[Z_{ij}|X_{ij} = x, \Theta_{ij}, \sigma^2] = 0.$$

Learning: In the M-step, starting from the current estimate of Θ and σ^2 and including the new statistics from the E-step (posterior mean and posterior variance), we get the updated parameters $\tilde{\Theta}$ and $\tilde{\sigma}^2$. Also in this case, the computation of the final results is not included, since it may be of less interest than the final result itself. We simply recall that the EM algorithm works, at each iteration, by calculating a surrogate for the log-likelihood that is easier to optimize [22]. Following this idea the surrogate that it has been chosen in [2] is given by

$$\begin{aligned} &E\left[\log p(Z|\tilde{\Theta}, \sigma^2)|X, \Theta, \sigma^2\right] = \\ &E\left[-\frac{mn}{2}\log(2\pi\tilde{\sigma}^2) - \frac{1}{2\tilde{\sigma}^2}\|Z - \tilde{\Theta}\|_F^2|X, \Theta, \sigma^2)\right] =, \\ &-\frac{mn}{2}\log(2\pi\tilde{\sigma}^2) - \frac{1}{2\tilde{\sigma}^2}\sum_{ij}\left[(\bar{Z_{ij}} - \tilde{\Theta_{ij}})^2 + \delta\bar{Z_{ij}}{}^2\right]. \end{aligned} \tag{4.18}$$

By maximizing the expression in (4.18) we complete the M-step and we get the new estimate of the model's parameters. This maximization is straightforward.

In particular, $\bar{\Theta}$ is updated solving the following optimization problem

$$\bar{\Theta} = \underset{\Theta}{\operatorname{argmin}}\|\Theta - \bar{Z}\|_F \quad \text{such that} \quad \operatorname{rank}(\Theta) = r. \tag{4.19}$$

The optimal solution of (4.19) is the TSVD of rank r of the matrix $\bar{Z}$ which comes from the previous E-step. This step represents the main computational cost of the algorithm, since computing TSVD may be very expensive for large dimensional matrices. The variance of the latent variable Z is updated as follows

$$\tilde{\Theta}^2 = \frac{1}{mn}\sum_{ij}\left[(\bar{Z}_{ij} - \tilde{\Theta}_{ij})^2 + \delta\bar{Z}_{ij}{}^2\right]. \tag{4.20}$$

The final result of the EM algorithm is a nonlinear low-rank decomposition
$X \approx \max(0, \Theta)$, where the error of the approximation is modeled by the magnitude of

σ^2. Note that the optimization of the loglikelihood in (4.14) is not convex so it is not possible to guarantee the convergence of the EM algorithm to a global maximum. This also mean that the final result may also depend on the initialization of the method. Furthermore, in [3] a momentum step as in (4.9) is added to the algorithm, we will refer to the accelerated EM algorithm as A-EM.

4.4 Aggressive momentum NMD (A-NMD)

In this section we present an heuristic acceleration technique that speeds up the convergence of the naive algorithm (Algorithm 6). In the naive algorithm, it is used a Polyak-type extrapolation as shown in (4.9), with a fixed momentum parameter. In addition, even though the variables involved in the alternating minimization process are two, Z and Θ, the extrapolation is applied to the variable Z only. One possible explanation for this choice is the fact that the extrapolation procedure on the variable Θ, i.e.

$$\Theta^{k+1} \leftarrow \Theta^{k+1} + \beta_k(\Theta^k - \Theta^{k-1}), \tag{4.21}$$

can modify the rank of the matrix. Recall that the constraint that we have on the approximation matrix Θ is that it has to be of a fixed $\text{rank}(\Theta) = r$. In other words, performing extrapolation on Θ increases the rank of the approximation, meaning that we are adding more information during the minimization process. Note that if one wants to apply extrapolation to the variable Θ, this should be avoided in the last iteration of the algorithm. On the other hand, if we are not at convergence, we can use the extrapolated Θ^k to compute the new Z^{k+1} in the successive iteration, and in practice, it considerably accelerates the convergence of the Naive algorithm.

Following this idea, we now present the aggressive momentum NMD (A-NMD), that was firstly introduced in [21]. The A-NMD algorithm uses a more aggressive Nesterov-type extrapolation with an heuristic approach to tune the momentum parameter β_k. The $Z-$update takes the form

$$Z^{k+1} \leftarrow Z^{k+1} + \beta_k(Z^{k+1} - Z^k), \tag{4.22}$$

where β_k is chosen adaptively. The first difference between A-NMD and A-Naive is that, instead of the Polyak-type extrapolation, here we have a Nesterov-type extrapolation. The overall procedure is described in Algorithm 8 and we note that the algorithm uses extrapolation for both variables, Z and Θ. The adaptive choice of the momentum parameter allows the algorithm to be less sensitive to that parameter, and to adapt depending on the problem at hand.

The adaptive scheme is based on the procedure in [19] for NMF. In that scheme, the momentum parameter, β_k at iteration k, is updated based on the decrease/increase of the objective function. Let the hyperparameters be $1 < \bar{\gamma} < \gamma < \eta$. The momentum

parameter is multiplied by γ as long as the objective function is decreasing, unless it reaches the adaptive upper bound $\bar{\beta}$. If the objective function decreases, we set $\bar{\beta} = \min(1, \bar{\gamma}\bar{\beta})$, meaning that we increase $\bar{\beta}$ by a factor $\bar{\gamma} < \gamma$. On the contrary, if at iteration k the error increases, the momentum parameter is divided by the factor η and we update the upper bound $\bar{\beta}$ as β_{k-1}, meaning that $\bar{\beta}$ keeps track of the latest value of β that allowed the decrease of the objective function. Algorithm 8 summarizes this strategy. The update of Z and Θ is only accepted if the error decreases, otherwise the parameters are updated while they keep the same value for the next iteration.

Algorithm 8 Aggressive momentum NMD (A-NMD)

Input: X, Z^0, Θ^0, r, $1 < \bar{\gamma} < \gamma < \eta$, $\beta_0 \in (0, 1)$, maxit.
Output: A rank-r matrix Θ s.t. $X \approx \max(0, \Theta)$.

Set $\bar{\beta} = 1$, $Z_{ij}^k = X_{ij}$ for $(i, j) \in I_+$ and for $k = 0, 1$.
for $k = 0, 1, \ldots,$ maxit **do**
  $Z_{ij}^{k+1} = \min(0, \Theta_{ij}^k)$ for $(i, j) \in I_0$.
  $Z^{k+1} \leftarrow Z^{k+1} + \beta_k(Z^{k+1} - Z^k)$.
  $[U, D, V] = \text{TSVD}(Z^{k+1}, r)$.
  $\Theta^{k+1} = UDV^T$.
  $\Theta^{k+1} \leftarrow \Theta^{k+1} + \beta_k(\Theta^{k+1} - \Theta^k)$.
  if $\|X - \max(0, \Theta^{k+1})\|_F < \|X - \max(0, \Theta^k)\|_F$ **then**
    $\beta_{k+1} = \min(\bar{\beta}, \gamma\beta_k)$, $\bar{\beta} = \min(1, \bar{\gamma}\bar{\beta})$.
  else
    $\beta_{k+1} = \beta_k \backslash \eta$, $\bar{\beta} = \beta_{k-1}$,
    $Z^{k+1} = Z^k$, $\Theta^{k+1} = \Theta^k$.
  end if
end for
$\Theta = \Theta^{k+1}$.

4.5 Three blocks NMD (3B-NMD)

All the algorithms presented so far share a common aspect: they all require the computation of a rank-r TSVD at each step. This fact may be a problem when considering problems of large dimension. For this reason, we now introduce the new Three block NMD (3B-NMD) [21] that exploits a three blocks variable splitting, which allows to avoid the TSVD computation. In particular, we substitute $\Theta \in \mathbb{R}^{m\times n}$ by the product WH, where $W \in \mathbb{R}^{m\times r}$ and $H \in \mathbb{R}^{r\times n}$, being r the desired rank of the NMD. Hence we reformulate the latent NMD problem in (4.7) as follows

$$\min_{Z,W,H} \|Z - WH\|_F^2 \quad \text{such that} \quad \max(0, Z) = X.$$

Note that it is not needed anymore to require $\text{rank}(\Theta) = r$, indeed, due to the splitting $\Theta = WH$, it holds $\text{rank}(\Theta) \leq \text{rank}(W) * \text{rank}(H) \leq r$. In other words we are solving a slightly more general case of the Latent NMD problem (4.7), in which we require only $\text{rank}(\Theta) \leq r$. In practice nothing changes because usually, the original, sparse, nonnegative matrix X has a rank of order $\min(n, m) > r$. Even though the rank of the approximation matrix Θ can be smaller than r, there is no reason for the algorithms to give as output a matrix of $\text{rank}(\Theta) < r$, since it has to approximate an higher rank matrix.

The overall procedure is presented in Algorithm 9 and it is again an alternating optimization between the variables Z, W, and H. As for Θ-update, the minimization subproblems for W and H have closed-form solutions; in fact, they can be obtained simply by solving matrix least square problems (backslash in MATLAB) which requires $O(mnr)$ operations, instead of the $O(mnr^2)$ operations for the TSVD. In addition, both the matrix $H^k(H^k)^T$ and $(W^{k+1})^T W^{k+1}$ are $r \times r$ with $r << min(m, n)$ so in general the solution of each subproblem can be computed easily also when dealing with large data. Solving the subproblems for W and H in more efficient ways is an open field of research, for example some stochastic techniques, such as randomized least squares, may be applied in order to further reduce the computational cost of the algorithm. To accelerate this block-coordinate descent method scheme, we also use Nesterov momentum extrapolation, after the computation of Z and of $\Theta = WH$.

Algorithm 9 Momentum three-block NDM (3B-NMD)

Input: X, Z^0, W^0, H^0,r, β, maxit.
Output: Two matrices W and H s.t. $X \approx \max(0, WH)$.

Set $Z_{ij}^k = X_{ij}$ for $(i, j) \in I_+$ and $k = 0, 1$.
for $k = 0, 1, \ldots,$ maxit **do**
  $Z_{ij}^{k+1} = \min(0, \Theta_{ij}^k)$ for $(i, j) \in I_0$.
  $Z^{k+1} \leftarrow Z^{k+1} + \beta(Z^{k+1} - Z^k)$.
  $W^{k+1} \leftarrow \arg\min_W \|Z^{k+1} - WH^k\|_F^2$.
  $H^{k+1} \leftarrow \arg\min_H \|Z^{k+1} - W^{k+1}H\|_F^2$.
  $\Theta^{k+1} \leftarrow W^{k+1}H^{k+1}$
  $\Theta^{k+1} \leftarrow \Theta^{k+1} + \beta(\Theta^{k+1} - \Theta^k)$.
end for
$W = W^{k+1}$, $H = H^{k+1}$.

Note that 3B-NMD does not use an adaptive strategy for the momentum parameter β, because we have observed that it is not as effective as in the naive case, described in the previous section. This is a topic for further research.

Chapter 5

Numerical results

In this chapter, we present the numerical results and some applications of ReLU-NMD on different data sets. At first, we describe one possible initialization strategy based on nuclear norm theory that can be used to detect a suitable starting point. Recall that up to our knowledge, there are no convergence results for ReLU-NMD, and choosing an appropriate starting point may improve consistently the performance of the algorithm.

We compare the following algorithms for ReLU-NMD: A-NMD, 3B-NMD from [21], Naive-NMD, A-Naive-NMD, EM-NMD and A-EM by Saul [2], applied to synthetic and real-world data set. As a baseline, we will also report the result of the projection of the TSVD, that is, $\max(0, X_r)$ where X_r is the rank-r TSVD of X. The first goal, indeed, is to show that in some cases, ReLU-NMD models can be used to achieve a better low-rank approximation than the one obtained by the TSVD, which is the state-of-the-art in this field.

All tests are preformed using Matlab R2021b on a laptop Intel CORE i5-1135G7 @ 2.40GHz 8GB RAM. The codes are available online at `https://gitlab.com/ngillis/ReLU-NMD`.

5.1 Initialization strategy

We provide at first an initialization strategy that can be adapted to all the mentioned algorithms, using the nuclear norm. Recall that not having any global convergence guarantees suggests that choosing a suitable starting point may have an impact in the minimization process.

Definition 5.1. *Let X be a matrix, and denote $\sigma_i(X)$ its singular values, then the nuclear norm of X is the sum of its singular values,*

$$\|X\|_* = \sum_i \sigma_i(X)$$

.

The nuclear norm has been used as a convex surrogate of the rank function, akin to the ℓ_1 norm used as a convex surrogate for the ℓ_0 norms [23], this particular property of nuclear norm makes it useful in several contexts in which the minimization of the rank is substituted by its convex surrogate. We the give the following definitions.

Definition 5.2 (Convex Envelope)**.** *Let $\mathcal{C}$ be a given convex set. The convex envelope of a (possibly nonconvex) function $f : \mathcal{C} \to \mathbb{R}$ is defined as the largest convex function g such that $g(x) \leq f(x)$ for all $x \in \mathcal{C}$.*

Theorem 5.1.1. *The convex envelope of* $\operatorname{rank}(X)$ *on the set* $\{X \in \mathbb{R}^{m\times n} : \|X\| \leq 1\}$ *is the nuclear norm* $\|X\|_*$.

Definition 5.3. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function. The subdifferential of f at $x_0 \in \mathbb{R}^n$ is the compact convex set*

$$\partial f(x) = \{d \in \mathbb{R}^n : f(x) - f(x_0) \geq \langle d, x - x_0\rangle, \quad \forall x \in \mathbb{R}^n\}$$

We now formulate a new problem which is closely related to ReLU-NMD, but in this new instance we focus on minimizing the rank as objective function. Note that this is a remarkable difference between problem (5.1) and ReLU-NMD, since in the second case, the rank is fixed to r. Assume there exists an exact rank-r ReLU-NMD solution but we do not know the rank r, the rank identification problem can be reformulated as follows:

$$\min_{\Theta} \operatorname{rank}(\Theta) \;\text{ such that }\; X = \max(0, \Theta), \tag{5.1}$$

which is a hard problem in general, since it is neither differentiable or convex. We then recall Theorem 5.1.1 that allows to built a slight relaxation of problem (5.1) which is easier to solve. Replacing the rank with the nuclear norm, obtaining the following convex relaxation:

$$\begin{aligned} \min_{\Theta} \|\Theta\|_* \;\text{ such that }\; & \Theta_{ij} = X_{ij} \text{ for } (i,j) \in I_+, \\ & \Theta_{ij} \leq 0 \text{ for } (i,j) \in I_0. \end{aligned} \tag{5.2}$$

where $I_+ = \{(i,j) \mid X_{ij} > 0\}$ and $I_0 = \{(i,j) \mid X_{ij} = 0\}$.

To solve (5.2), we resort to a standard projected subgradient strategy [23]. A subgradient method is a more general case of gradient descent, which can be applied to functions which are not differentiable but convex as it happens in (5.2) and it gives at each step a descend direction which lies in the subdifferential of the objective function. The new iterate is then computed projecting the subgradient step into the feasible set. Therefore, update Θ as follows

$$\Theta_{k+1} = \Pi(\Theta_k - \alpha_k Y_k), \qquad Y_k \in \partial\|\Theta_k\|_*,$$

where $\Pi(\cdot)$ is the projection onto the feasible set (which is easy to compute), and $\partial\|\Theta_k\|_*$ is a subgradient of the nuclear norm at Θ_k, given by [24]

$$\partial\|\Theta\|_* = \Big\{UV^T + P : P \text{ and } \Theta \text{ have orthogonal row and column spaces, and } \|P\| \leq 1\Big\}, \quad (5.3)$$

where $(U, \Sigma, V) \in \mathbb{R}^{m\times r} \times \mathbb{R}^{r\times r} \times \mathbb{R}^{n\times r}$ is a TSVD of Θ and $\|.\|$ is the operator norm (or induced 2-norm) of a matrix. In our implementation, we used $P = 0$.

Note that the solution is only used for initialization and hence we do not need high accuracy. In addition, in the experiment part the step α_k is selected using a backtracking approach. Moreover, the solution of (5.2) is not guaranteed to be of rank smaller than r, and hence we use the rank-r TSVD of the last iterate as an initialization for the ReLU-NMD algorithms presented in the previous sections. The idea that stands behind this initialization procedure is the fact that imposing the constraint $X = \max(0, \Theta)$ and decreasing the rank of Θ at each step of the projected gradient algorithm, we are getting closer to the optimal solution of rank-r ReLU-NMD problem (3.1).

5.2 Datasets

We present now the data sets used to validate the results produced by each algorithm. At first, we show a preliminar analysis on synthetic data set, that is a data set built to fit the ReLU-NMD problem and then we also test some real word data sets, which are well-known in the numerical analysis litterature.

- **Synthetic data set:** the matrix $X \in \mathbb{R}^{m\times n}$ is generated as $X = \max(0, WH)$, where the entries of $W \in \mathbb{R}^{m\times r}$ and $H \in \mathbb{R}^{r\times n}$ are obtained from the normal distribution, that is, `W = randn(m,r)` and `H = randn(r,n)` in MATLAB. Since the probability for X to have a positive entry is equal to that of having a negative entry (by symmetry), X has, on average, 50% of its entries equal to zero. Note that X can be potentially full rank but at least, we know that it exists one solution to the problem, which is given by $\Theta = WH$. When working with synthetic data, all algorithms are stopped when

$$\text{relative error} = \frac{\|X - \max(0, \Theta)\|_F}{\|X\|_F} \leq 10^{-4}, \quad (5.4)$$

  where Θ is the current solution. Note that, given the non-convexity of ReLU-NMD, there is no guarantee that an algorithm converges to such a small relative error. However, for the synthetic data as generated above, this is always the case.

- **MNIST dataset:** MNIST is a large database of small, square 28×28 pixel grayscale images of handwritten single digits between 0 and 9. It consists of a total of 70,000 handwritten images of digits. All images are labeled with the respective digit that they represent. There are a total of 10 classes of digits (from 0 to 9). The data set is widely used to test several machine learning algorithms and it is particularly suitable for recognition and compression methods. See Figure 5.1 for an example of different digits.

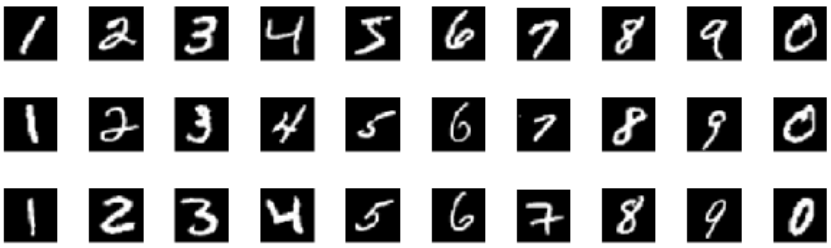


Figure 5.1: Example of randomly picked 28×28, greyscale images of handwritten digits from MNIST data set.

- **CBCL data set:** the CBCL data set, was used in the seminal paper of Lee and Seung [25] in the context of NMF. Each column of the data set matrix contains a greyscale, vectorized facial image of size 19×19, see Figure 5.2. The total number of images in is 2429. This data set is particularly suitable for features extraction applications because of the variety of the images in it. It is one of the most used in NMF because the factors of the decomposition can be easily interpretated and they correspond to different facial features such as eyes, nose and mouth.

Figure 5.2: Example of facial 19×19, greyscale images from CBCL data set.

5.3 Numerical results for synthetic data set

Let us now show the results of the different NMD-algorithms when applied to synthetic data, meaning to data designed in order to produce the results expected from the theory. Of course, this is not a complete validation process for the algorithms, but it gives an idea of the behaviour of each algorithm and it is also useful for tuning the parameters of the models. In the following, we will show different preliminary analysis for the initialization strategy, we test different acceleration techniques for the naive algorithm, that is Algorithm 7 and in conclusion, we will include a comparison between the new algorithms A-NMD and 3B-NMD with the-state-of-the-art EM-NMD and A-EM.

5.3.1 Effectiveness of initialization

Let us first validate the effectiveness of the nuclear norm initialization. As stated before, this initialization strategy provides us with a better starting point for our algorithms. To prove this, we look at the relative error in (5.4) for the following initialization techniques:

- Random initialization where Θ is a rank-r matrix, generated in the same way as X, and which we scale optimally as follows $\Theta \leftarrow \alpha^*\Theta$ where

$$\alpha^* = \underset{\alpha}{\operatorname{argmin}} \|X - \alpha \ \max(0, \Theta)\|_F = \frac{\langle X, \max(0,\Theta)\rangle}{\|\max(0,\Theta)\|_F^2}.$$

- The initialization Θ taken as the rank-r TSVD of X.
- The nuclear norm minimization strategy: few iterations of projected subgradient method starting from random initialization and optimal scaling as described above.

We test the initialization for different values of the rank $r = 8, 16$ and size of the matrix $X \in \mathbb{R}^{m\times n}$, with $m = n = 500, 1000, 1500, 2000$. The results are collected in Table 5.1. We observe that the nuclear norm allows an initial solution with significantly

	$r = 8$			$r = 16$		
$m = n$	rand	TSVD	$\|.\|_*$	rand	TSVD	$\|.\|_*$
500	0.95	0.40	**0.36**	0.95	0.36	**0.32**
1000	0.95	0.41	**0.38**	0.95	0.37	**0.33**
1500	0.95	0.41	**0.38**	0.95	0.38	**0.33**
2000	0.95	0.41	**0.38**	0.95	0.38	**0.33**

Table 5.1: Initial relative error of three initializations: random initialization and scaling, the rank-r TSVD, and the nuclear norm initialization ($\|.\|_*$).

smaller relative error: a reduction from a factor 2 to 3 compared to a random initialization, and about 10% improvement compared to the TSVD. Hence, all the experiments in the following of this work are initialized using the nuclear norm strategy. Note that in general, solving problems such as the one in (5.2) that require to minimize the nuclear norm is a difficult task. It can be tackled using some sophisticated and computationally expensive techniques to solve it at optimality, such as interior point methods, but we do not look for an accurate solution in this context. This is the reason why few iterations are performed and it is used only as initialization.

5.3.2 Naive algorithms comparison

We now show the results on synthetic data for the algorithms that share the naive scheme in Algorithm 6, with different acceleration strategies. In particular, we compare:

- **Naive Algorithm**: the standard and easiest algorithm that can be used to solve ReLU-NMD [2]; see Algorithm 6.
- **A-Naive**: Naive algorithm adding a Polyak extrapolation step only on the variable Z, Θ is not modified; see Algorithm 7
- **A-NMD**: Naive algorithm using Nesterov acceleration step on both the variables Z and Θ with adaptive momentum parameter [19];

We considered matrices of different sizes fixing the rank to $r = 32$. Even though there are no convergence guarantees for any of the presented algorithms, when synthetic data are taken into account, the tolerance in (5.4) is always achieved. Table 5.2 reports the total time and iterations needed to reach a relative error as in (5.4). We considered averaged values over 5 different synthetic matrices, with 5 different random initializations, post-processed using the nuclear norm algorithm.

	Naive		A-Naive		A-NMD	
Size	time	iter	time	iter	time	iter
500	1.84	110	0.78	44	**0.57**	32
1000	7.21	87	2.85	33	**2.28**	27
1500	11.4	80	4.40	29	**3.71**	25
2000	21.2	78	8.19	28	**6.80**	24

Table 5.2: Average computational time needed to satisfy condition in (5.4) on synthetic data with $r = 32$ [21].

We observe that A-NMD outperforms Naive and A-Naive: it is about 4 times faster than Naive, and 20% faster, on average, than A-Naive. Note that the cost per iteration

between the algorithms is almost the same for all of them, but A-NMD needs less iterations to reach the given tolerance, thanks to the adaptive technique used to tune the momentum parameter. In the following experiments, we will therefore only compare the other algorithms with A-NMD.

5.3.3 New algorithms VS state-of-the-art algorithms

Let us now present the results of the-state-of-the-art algorithms in [2] and [3] on synthetic data, compared to the new solution strategies introduced in section 4.4 and 4.5. In more details, we highlight the improvements in the performance, in terms of reduction of the relative error in different cases. In particular, we compare the expectation-minimization algorithms in simple form and with Polyak extrapolation, EM-NMD and A-EM respectively with A-NMD and 3B-NMD.

Table 5.3 reports the total time and iterations needed to satisfy condition in (5.4), for a fixed rank $r = 32$, for the EM algorithm of Saul (EM-NMD), its accelerated variant (A-EM), as well as our proposed algorithms A-NMD and 3B-NMD. Table 5.4 reports the same quantities for fixed dimensions $m = n = 1000$, but with different values of the rank, r. We run the experiments on 5 synthetic matrices, with 5 different random initializations, post-processed using nuclear norm algorithm and we display the average values.

	A-NMD		3B-NMD		EM-NMD		A-EM	
Size	time	iter	time	iter	time	iter	time	iter
500	0.64	33	**0.08**	23	2.2	101	1.0	43
1000	2.4	27	**0.24**	24	8.1	78	3.9	36
1500	3.6	24	**0.46**	24	12.8	70	6.5	35
2000	6.7	25	**0.81**	24	22.1	66	11.6	34

Table 5.3: Time needed to satisfy condition in (5.4) for synthetic matrices of increasing dimension with $r = 32$ [21].

	A-NMD		3B-NMD		EM-NMD		A-EM	
r	time	iter	time	iter	time	iter	time	iter
8	2.0	33	**0.22**	22	10.1	97	4.4	42
16	2.0	24	**0.20**	24	7.7	77	3.5	36
32	2.2	23	**0.22**	24	7.4	72	3.6	33
64	2.5	23	**0.25**	25	8.4	66	3.7	32

Table 5.4: Computational time needed to satisfy condition in (5.4) when approximating synthetic data of fixed size $n = m = 1000$ for different values of the rank, r [21].

We observe that A-EM performs better than EM-NMD, as expected, and hence we will report only the results for A-EM in the section on real-world data sets. Then, we observe that A-NMD performs better than A-EM (almost twice faster in all cases), while 3B-NMD outperforms all other algorithms, being more than 10 times faster than A-EM. This behaviour can be explained by the fact that 3B-NMD does not need to perform a TSVD in the minimization process, so the average computational cost per iteration is consistently reduced. Furthermore, note that, even though the cost per iteration between A-NMD and A-EM is similar, the total time to reach the convergence is lower for the A-NMD algorithm, since it takes less iterations that A-EM. In conclusion, when working with synthetic data, the new 3B-NMD and A-NMD algorithms are more effective then the state-of-the-art EM-NMD and A-EM.

5.4 Numerical results for the MNIST Data set

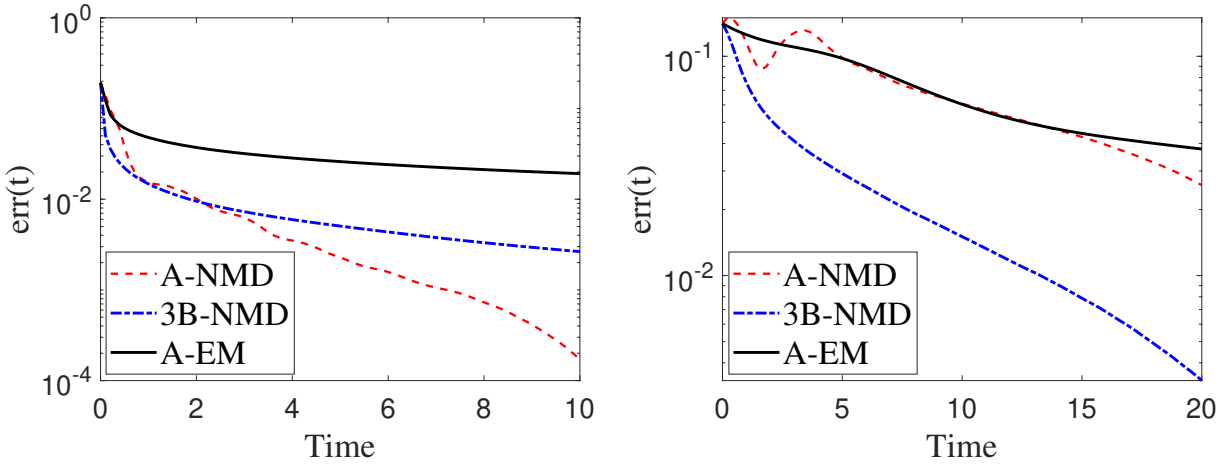
The experiments in this section are computed on 28×28 grayscale images of MNIST handwritten digits [26], where X is generated by concatenating vectorized images into a matrix of size $784 \times n$ where $n = 500$ or $n = 50000$, depending on the experiment. To compare the ReLU-NMD algorithms, we do not consider the usual relative error anymore, since, working with real world data, it does not allow to observe remarkable differences between the methods. This is because there is no more a convergence of the error to zero but the error becomes flat when it reaches a certain tolerance. We then define the following quantity:

$$err(t) = \frac{\|X - \max(0, \Theta(t))\|_F}{\|X\|_F} - e_{min}, \tag{5.5}$$

where $\Theta(t)$ is the solution at time t, and e_{min} is the smallest relative error obtained by any algorithms within the allotted time. Since $err(t)$ converges to zero for the algorithm that computed the best solution, we can represent the error in log scale. Thanks to this property, the results are better visualized both in terms of initial convergence and quality of the final approximation. We obtained a continuous approximation of $err(t)$ interpolating some samples using a cubic spline approach. Figure 5.3 (a) displays the results on the MNIST data set with $r = 32$ with 500 images (50 images of each digit), with a timelimit of 10 seconds. Although 3B-NMD converges initially faster, A-NMD eventually catches up and generates the best solution. As before, A-EM is outperformed. We can observe that using a relative small data set, A-NMD is the algorithm that has better performance since the adaptive strategy on the extrapolation parameter allows to have a faster decrease. Note that in this situation the cost per iteration is not extremely high, since we are taking into account only 500 images.

Figure 5.3 (b) displays the results on the MNIST data set with $r = 32$ with all the 50000 images, with a timelimit of 20 seconds. In this case, 3B-NMD converges initially

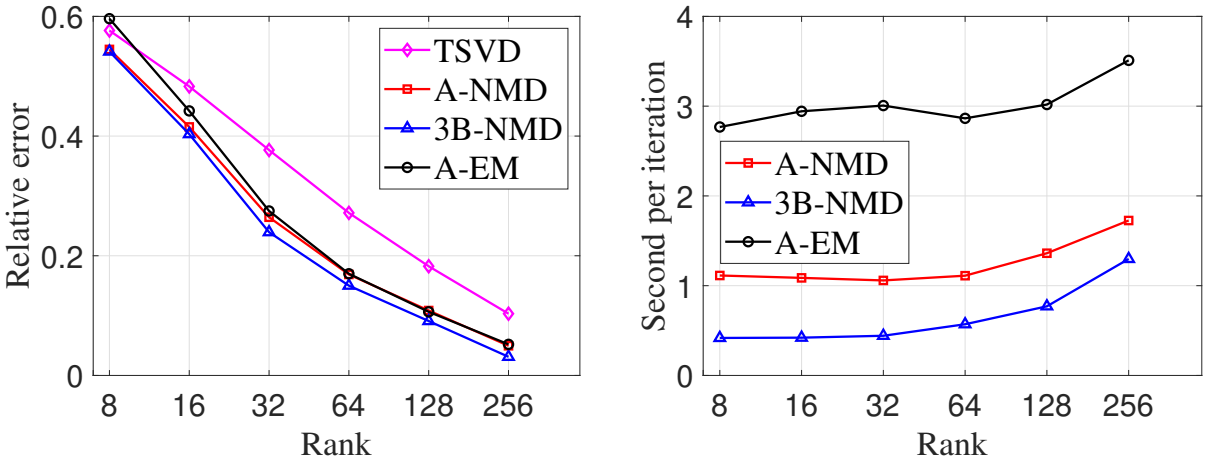
faster and A-NMD does not have time to catch up. Note that the more is increased the dimension of the data set and the more the computational cost of the TSVD becomes high. Due to this fact 3B-NMD is considerably faster than the other algorithms since it can perform more iterations in the given timelimit. In any case, 3B-NMD and A-NMD both perform well, outperforming the state-of-the-art algorithm A-EM.



(a) $n = 500$. (b) $n = 50000$.

Figure 5.3: Average value of the error (5.5) of A-NMD, 3B-NMD and A-EM on small (a) and large (b) portion of the MNIST data set.

Figure 5.4 compares the algorithms with the baseline, TSVD, in terms of relative error as the rank increases, and provides the average time per iteration, for 50000 images of MNIST and a runtime of 20 seconds. These results confirm the effectiveness of the 3B-NMD in order to deal with large data, the cost per iteration being smaller than that of A-NMD and A-EM. In addition, note that all the ReLU-NMD algorithms approximate the dataset with considerably higher accuracy than the TSVD, as expected. This means that solving ReLU-NMD model, we get a better low rank approximation than the one obtained using the TSVD approach.



(a) Relative error.

(b) Time (seconds).

Figure 5.4: Final relative error on $m = 50000$ images from MNIST dataset, after 20 seconds and average iteration time for increasing value of the rank r.

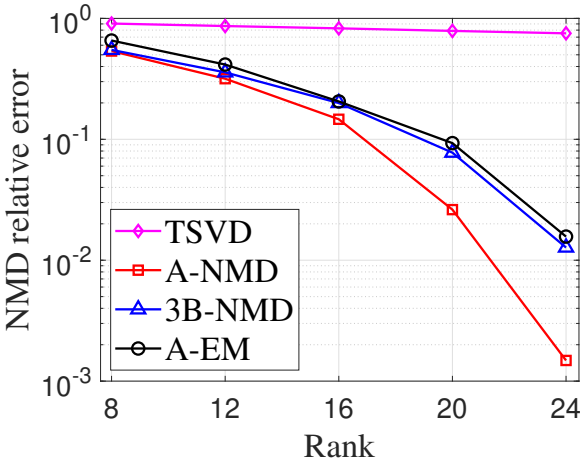
5.5 Application of ReLU-NMD: compression of sparse NMF basis

We here present a first application of ReLU-NMD, which is new to the best of our knowledge, that is the compression of sparse nonnegative dictionaries, e.g., the factors generated by NMF, see Chapter 2. For this experiments we considered the CBCL facial data set, see Figure 5.2 for an example. Each column of the data matrix $X \in \mathbb{R}^{361\times 2429}$ contains a vectorized facial image of size 19×19. The NMF decomposition, $X \approx UV$ where $U \geq 0$ and $V \geq 0$, allows one to extract sparse facial features as the columns of U. To do so, we have used one of the NMF code from `gitlab.com/ngillis/nmfbook/`, requiring some sparsity constraints, and used a rank-100 NMF to obtain $U \in \mathbb{R}^{361\times 100}$, a nonnegative sparse matrix (in fact, 85% of the entries are equal to zero); see Figure 5.6 (a) for an illustration. Further compressing the NMF factor U using the TSVD does not work as Figure 5.5 (a) shows, with a relative error larger than 70%; see also Figure 5.6 (b). This is because NMF tends to generate factors U whose singular values are all large. This means that the information is spread among all the singular values and considering only some of them, we loose some pieces of information. However, ReLU-NMD does not have this limitation, since it works using an elementwise operator and it looks for manifold patterns: it can approximate well such sparse full-rank matrices (e.g., the identity matrix [2], see Chapter 3). Denoting $\hat{U} = \max(0, \Theta)$ the approximation matrix obtained by a ReLU-NMD algorithm, we first evaluate the relative error as in (5.4) between U and $\hat{U}$. Results are displayed for increasing values of the rank in Figure 5.5 (a).

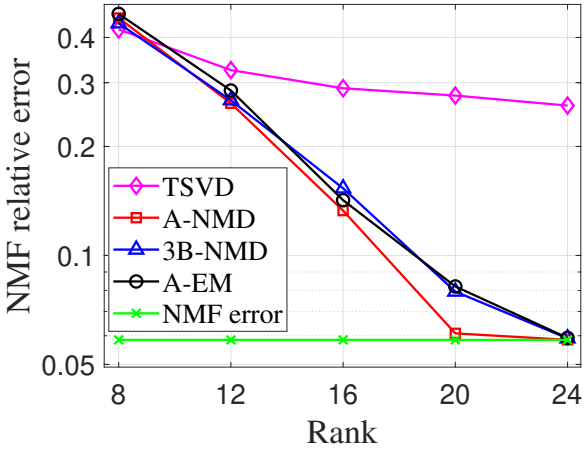
We also evaluate the error of the compressed NMF

$$e_{NMF} = \min_{\hat{V} \geq 0} \frac{\|X - \max(0, \hat{U})\hat{V}\|_F}{\|X\|_F}, \tag{5.6}$$

see Figure 5.5 (b). In these experiments, we fix a time limit of 20 seconds and a tolerance of 10^{-4} for the relative error as a stopping criteria. It is clear that the TSVD is not able to compress the features with high accuracy while all the other algorithms can reach a low relative error for small values of the rank. For such data sets, it appears that A-NMD reaches the best solution. For example, with $r = 20$, it is able to reach almost the same accuracy on the NMF problem as the original factor of size 100. This is the behaviour that could be predicted since the data set which is taken into account is quite small and the cost of the TSVD is not too high. However, increasing the dimension of the data set the 3B-NMD becomes more efficient, since the average cost per iteration is considerably lower than the other algorithms. Figure 5.6 shows an example of a rank $r = 20$ reconstruction of the original rank $r = 100$ NMF factor. It is clear that the reconstruction provided by A-NMD and 3B-NMD are visually more similar to the original factor, while the TSVD reconstruction is affected by some visible noise.

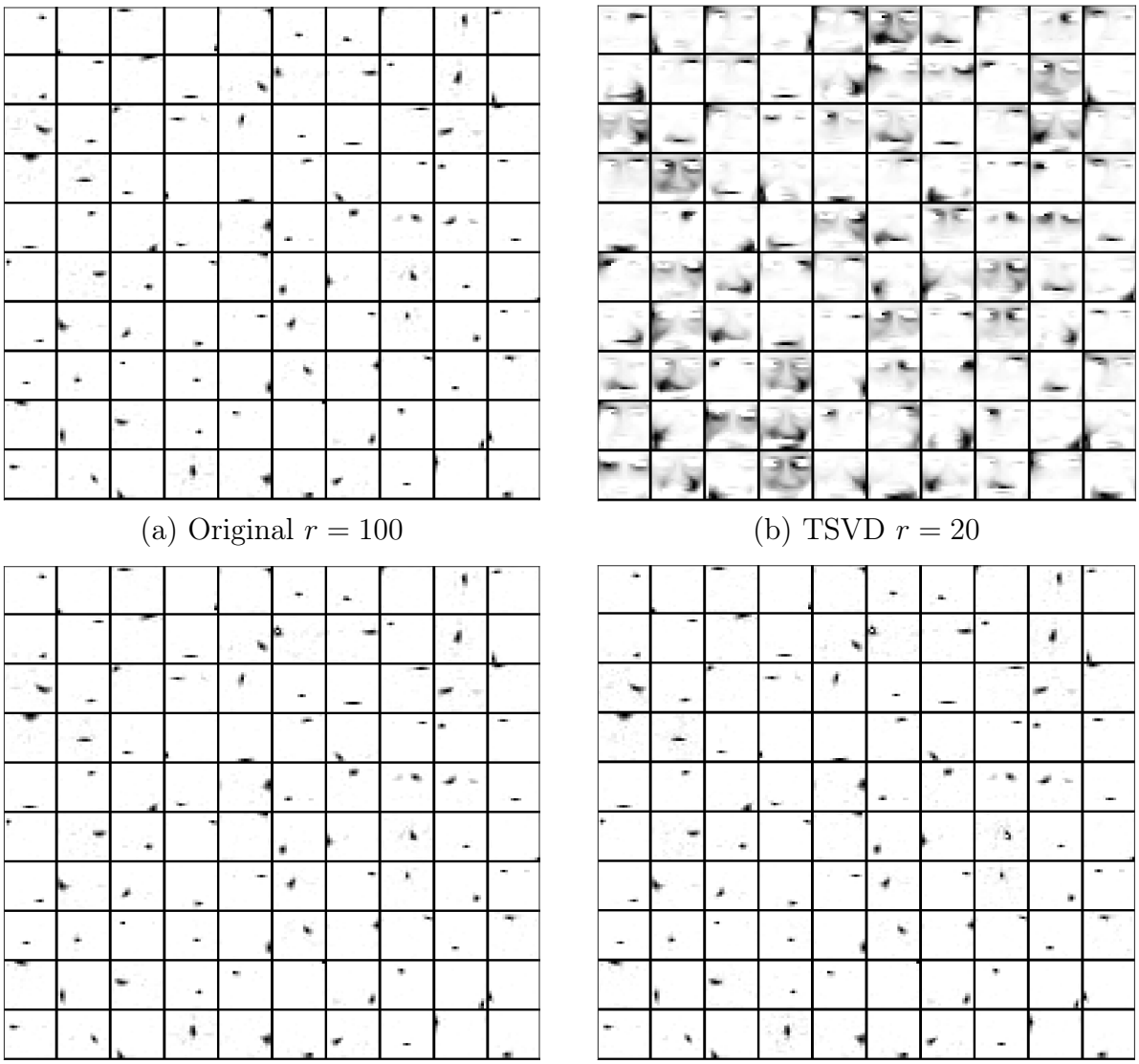


(a) Error on U.



(b) Error on X; see (5.6).

Figure 5.5: Compression of a 361-by-100 NMF basis, U, of the CBCL data set. Image (a) shows the error on the NMF basis $U \geq 0$. Image (b) shows the NMF error after U is replaced by its approximation; see (5.6).



(a) Original $r = 100$ (b) TSVD $r = 20$

(c) A-NMD $r = 20$ (d) 3B-NMD $r = 20$

Figure 5.6: (a) Original factor U of NMF with 100 columns reshaped as facial features, and its rank-20 approximations by (b) TSVD, (c) A-NMD, and (d) 3B-NMD.

Conclusions

Low-rank approximation models are an open and challenging field of research. The main goal is to express a given set of data, contained in a matrix X as another matrix Θ which has lower rank. We presented both linear and nonlinear models and discussed when one should be preferred to the other one.

Firstly, the well-known SVD and PCA methods have been described giving some existence guarantees and investigating the relation between the two. They both look for the most meaningful vector base to express a given data set, that means detecting which are the principal directions that can better express the highest possible amount of information.

Secondly, we introduced one example of Linear Dimensionality Reduction (LDR) which is the Nonnegative Matrix Factorization (NMF). It decomposes a given matrix X as the product of two nonnegative factors U and V. The strength of NMF factorization is the intuitive interpretation of its factors and the possibility to split the problem in two subproblems, one for each variable, which are in general easier to solve. These aspects make it very popular in data compression, features extraction from images, text mining, etc.

Thirdly, we presented the Nonlinear Matrix Decomposition (NMD) which becomes useful when the relation between the variables in the data is not linear. In contrast with NMF, NMD does not look for a representation of the data in a different base, while it seeks those features that remain constant both in the original matrix and in its low-rank approximation. In particular, we focused on the so called ReLU-NMD poblem which involves the ReLU function $f_{ReLU} = \max(0, \cdot)$, widely used in machine learning applications. We presented the state-of-the-art algorithms developed by Saul [3, 2] and we introduced the new methods A-NMD and 3B-NMD [21] which use new acceleration techniques and a new variable splitting which, to the best of our knowledge, are new in this context.

Finally, we validated the results testing the algorithms on different data sets. It is clear from the results that both the new algorithms perform in most cases better than the state-of-the-art methods. In particular, the adaptive strategy to select the momentum parameter in the A-NMD accelerates consistently the naive scheme described in [2] and makes this algorithms particularly efficient when dealing with small size data.

At the contrary, the 3B-NMD scheme does not include the Truncated Singular Value Decomposition (TSVD) step and it reduces the computational cost per iteration. Thanks to this property, the numerical results show that it is the most efficient algorithm for data sets of large dimension. Furthermore, the subproblem deriving from the variable splitting in the 3B-NMD have closed form solution and they can be easily solved.

Our future research plan includes further investigating the ReLU-NMD model. In particular, we want to focus on the case when the approximation rank of the matrix Θ is equal to 1, since we expect to prove that the problem is NP-hard. From the algorithmic point of view, we plan to develop a stochastic version of the 3B-NMD that will allow to further reduce the computational cost of the procedure. Furthermore, the convergence properties of both algorithms will be explored too.

Bibliography

[1] N. Gillis, *Nonnegative matrix factorization*. SIAM, 2020.

[2] L. K. Saul, "A nonlinear matrix decomposition for mining the zeros of sparse data," *SIAM Journal on Mathematics of Data Science*, vol. 4, no. 2, pp. 431–463, 2022.

[3] L. K. Saul, "A geometrical connection between sparse and low-rank matrices and its application to manifold learning," *Transactions on Machine Learning Research*, 2022.

[4] M. J. Powell, "On search directions for minimization algorithms," *Mathematical programming*, vol. 4, pp. 193–201, 1973.

[5] L. Grippo and M. Sciandrone, "On the convergence of the block nonlinear gauss–seidel method under convex constraints," *Operations research letters*, vol. 26, no. 3, pp. 127–136, 2000.

[6] L. Grippof and M. Sciandrone, "Globally convergent block-coordinate techniques for unconstrained optimization," *Optimization methods and software*, vol. 10, no. 4, pp. 587–637, 1999.

[7] N. Gillis and F. Glineur, "Accelerated multiplicative updates and hierarchical als algorithms for nonnegative matrix factorization," *Neural computation*, vol. 24, no. 4, pp. 1085–1105, 2012.

[8] H. Le, N. Gillis, and P. Patrinos, "Inertial block proximal methods for non-convex non-smooth optimization," in *International Conference on Machine Learning*, pp. 5671–5681, PMLR, 2020.

[9] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Comput. Math. Math. Phys.*, vol. 4, no. 5, pp. 1–17, 1964.

[10] Y. Xu and W. Yin, "A globally convergent algorithm for nonconvex optimization based on block coordinate update," *Journal of Scientific Computing*, vol. 72, no. 2, pp. 700–734, 2017.

[11] C. F. Van Loan and G. Golub, "Matrix computations (Johns Hopkins studies in mathematical sciences)," *Matrix Computations*, vol. 5, 1996.

[12] D. Lee and H. Sompolinsky, "Learning a continuous hidden variable model for binary data," *Advances in Neural Information Processing Systems*, vol. 11, 1998.

[13] D. M. Blei, "Probabilistic topic models," *Communications of the ACM*, vol. 55, no. 4, pp. 77–84, 2012.

[14] S. A. Vavasis, "On the complexity of nonnegative matrix factorization," *SIAM Journal on Optimization*, vol. 20, no. 3, pp. 1364–1377, 2010.

[15] P. Paatero and U. Tapper, "Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values," *Environmetrics*, vol. 5, no. 2, pp. 111–126, 1994.

[16] J. Nocedal and S. J. Wright, *Numerical optimization*. Springer, 1999.

[17] D. P. Bertsekas, "Nonlinear programming," *Journal of the Operational Research Society*, vol. 48, no. 3, pp. 334–334, 1997.

[18] Y. Nesterov *et al.*, *Lectures on convex optimization*, vol. 137. Springer, 2018.

[19] A. M. S. Ang and N. Gillis, "Accelerating nonnegative matrix factorization algorithms using extrapolation," *Neural computation*, vol. 31, no. 2, pp. 417–439, 2019.

[20] H. S. Seung and D. D. Lee, "The manifold ways of perception," *science*, vol. 290, no. 5500, pp. 2268–2269, 2000.

[21] G. Seraghiti, A. Awari, A. Vandaele, M. Porcelli, and N. Gillis, "Accelerated algorithms for nonlinear matrix decomposition with the relu function," *arXiv preprint arXiv:2305.08687*, 2023.

[22] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the royal statistical society: series B (methodological)*, vol. 39, no. 1, pp. 1–22, 1977.

[23] B. Recht, M. Fazel, and P. A. Parrilo, "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization," *SIAM Review*, vol. 52, no. 3, pp. 471–501, 2010.

[24] G. A. Watson, "Characterization of the subdifferential of some matrix norms," *Linear Algebra and its Applications*, vol. 170, pp. 33–45, 1992.

[25] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[26] L. Deng, "The MNIST database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.