Descent methods for Nonnegative Matrix Factorization

Ngoc-Diep Ho, Paul Van Dooren and Vincent D. Blondel

CESAME, Université catholique de Louvain,
Av. Georges Lemaître 4, B-1348 Louvain-la-Neuve, Belgium.
Email: {*ngoc.ho, paul.vandooren, vincent.blondel*}*@uclouvain.be*
Tel : +32(10) 47 22 64 Fax : +32(10) 47 21 80

Abstract

In this paper, we present several descent methods that can be applied to nonnegative matrix factorization and we analyze a recently developped fast block coordinate method. We also give a comparison of these different methods and show that the new block coordinate method has better properties in terms of approximation error and complexity. By interpreting this method as a rank-one approximation of the residue matrix, we also extend it to the nonnegative tensor factorization and introduce some variants of the method by imposing some additional controllable constraints such as: sparsity, discreteness and smoothness.

1 Introduction

Linear algebra has become a key tool in almost all modern techniques for data analysis. Most of these techniques make use of linear subspaces represented by eigenvectors of a particular matrix. In this paper, we consider a set of n data points $a_1, a_2, \ldots, a_n$, where each point is a real vector of size m, $a_i \in \mathbb{R}^m$. We then approximate these data points by linear combinations of r basis vectors $u_i \in \mathbb{R}^m$:

$$a_i \approx \sum_{j=1}^{r} v_{ij} u_j, \qquad v_{ij} \in \mathbb{R},\ u_j \in \mathbb{R}^m.$$

This can be rewritten in matrix form as $A \approx UV^T$, where a_i and u_i are respectively the columns of A and U and the ${v_{ij}}'s$ are the elements of V.

Optimal solutions of this approximation in terms of the Euclidean (or Frobenius) norm can be obtained by the Singular Value Decomposition (SVD) [11].

In many cases, data points are constrained to a subset of $\mathbb{R}^m$. For example, light intensities, concentrations of substances, absolute temperatures are, by their nature, nonnegative (or even positive) and lie in the nonnegative orthant $\mathbb{R}^m_+$. The input matrix A then becomes elementwise nonnegative and it is then natural to constrain the basis vectors v_i and the coefficients v_{ij} to be nonnegative as well. In order to satisfy this constraint, we need to approximate the columns of A by the following additive model:

$$a_i \approx \sum_{j=1}^{r} v_{ij} u_j, \qquad v_{ij} \in \mathbb{R}_+,\ u_j \in \mathbb{R}^m_+.$$

where the v_{ij} coefficients and u_j vectors are nonnegative, $v_{ij} \in \mathbb{R}_+,\ u_j \in \mathbb{R}^m_+$.

Many algorithms have been proposed to find such a representation, which is referred to as a Nonnegative Matrix Factorization (NMF). The earliest algorithms were introduced by Paatero [22, 23]. But the topic became quite popular with the publication of the algorithm of Lee and Seung in 1999 [17] where multiplicative rules were introduced to solve the problem. This algorithm is very simple and elegant but it lacks a complete convergence analysis. Other methods and variants can be found in [20], [18], [14].

The quality of the approximation is often measured by a distance. Two popular choices are the Euclidean (Frobenius) norm and the generalized Kullback-Leibler divergence. In this paper, we focus on the Euclidean distance and we investigate descent methods for this measure. One characteristic of descent methods is their monotonic decrease until they reach a stationary point. This point maybe located in the interior of the nonnegative orthant or on its boundary. In the second case, the constraints become active and may prohibit any further decrease of the distance measure. This is a key issue to be analyzed for any descent method.

In this paper, $\mathbb{R}^m_+$ denotes the set of nonnegative real vectors (elementwise) and $[v]_+$ the projection of the vector v on $\mathbb{R}^m_+$. We use $v \geq 0$ and $A \geq 0$ to denote nonnegative vectors and matrices and $v > 0$ and $A > 0$ to denote positive vectors and matrices. $A \circ B$ and $\frac{[A]}{[B]}$ are respectively the Hadamard (elementwise) product and quotient. $A_{:i}$ and $A_{i:}$ are the i^{th} column and i^{th} row of A.

This paper is an extension of the internal report [13], where we proposed to decouple the problem based on rank one approximations to create a new algorithm. During the revision of this report, we were informed that essentially the same algorithm was proposed in an independent contribution [7] namely Hierarchical Alternative Least Squares (HALS). But in the present

paper we give several new properties and also prove a new convergence result of this method. The paper compares different descent methods known in the literature and aims at providing a survey of such methods for nonnegative matrix factorizations. For that reason, we try to be self-contained and hence also recall some well-known results for which we provide short proofs when useful for a better understanding of the rest of the paper.

We first give a short introduction of low rank approximations, both unconstrained and constrained. In Section 3 we discuss error bounds of various approximations and in Section 4 we give a number of descent methods for Nonnegative Matrix Factorizations. In Section 6 we describe the method based on successive rank one approximations. This method is then also extended to approximate higher order tensor and to take into account other constraints than nonnegativity. In Section 7 we discuss various regularization methods and in Section 8, we present numerical experiments comparing the different methods. We end with some concluding remarks.

2 Low-rank matrix approximation

Low rank approximation is a special case of matrix nearness problem [12]. When only a rank constraint is imposed, the optimal approximation can be obtained from the singular value decomposition. We first investigate the problem without the nonnegativity constraint on the low-rank approximation. This is useful for understanding properties of the approximation when the nonnegativity constraints are imposed but inactive. We begin with the well-known Eckart-Young Theorem.

Theorem 1 (Eckart-Young)**.** *Let $A \in \mathbb{R}^{m\times n}$ $(m \geq n)$ have the singular value decomposition*

$$A = P\Sigma Q^T,\ \Sigma = \begin{pmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_n \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix}$$

where $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n$ are the singular values of A and where $P \in \mathbb{R}^{m\times m}$

and $Q \in \mathbb{R}^{n\times n}$ are orthogonal matrices. Then for $1 \leq r \leq n$, the matrix

$$A_r = P\Sigma_r Q^T,\ \Sigma_r = \begin{pmatrix} \sigma_1 & 0 & \dots & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & & \vdots \\ 0 & 0 & \dots & \sigma_r & \dots & 0 \\ \vdots & \vdots & & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & \dots & 0 \end{pmatrix}$$

is a global minimizer of the problem

$$\min_{B\in\mathbb{R}^{m\times n}\ rank(B)\leq r} \frac{1}{2}\|A-B\|_F^2 \tag{1}$$

and its error is

$$\frac{1}{2}\|A-B\|_F^2 = \frac{1}{2}\sum_{i=r+1}^{t}\sigma_i^2.$$

Moreover, if $\sigma_r > \sigma_{r+1}$ then A_r is the unique global minimizer.

The proof and other implications can be found for instance in [11].

Let us now look at the following modified problem

$$\min_{X\in\mathbb{R}^{m\times r}\ Y\in\mathbb{R}^{n\times r}} \frac{1}{2}\|A-XY^T\|_F^2, \tag{2}$$

where the rank constraint is implicit in the product XY^T since the dimensions of X and Y guarantee that $rank(XY^T) \leq r$. Conversely, every matrix of rank less than r can be trivially rewritten as a product XY^T, where $X \in \mathbb{R}^{m\times r}$ and $Y \in \mathbb{R}^{n\times r}$. Therefore the optimization problems (1) and (2) appear to be equivalent. But even when the product $A_r = XY^T$ is unique, there are infinitely many pairs (XR^T, YR^{-1}) with a R invertible that yield the same product XY^T. In order to fix this, we can always choose X and Y such that

$$X = PD^{\frac{1}{2}} \text{ and } Y = QD^{\frac{1}{2}}, \tag{3}$$

where $P^TP = I_{r\times r}$, $Q^TQ = I_{r\times r}$ and D is $r\times r$ nonnegative diagonal matrix. Doing this is equivalent to computing a compact SVD decomposition of the product $A_r = XY^T = PDQ^T$.

As usual for optimization problems, we calculate the gradient with respect to X and Y and set them equal to 0

$$\nabla_X = XY^TY - AY = 0 \qquad \nabla_Y = YX^TX - A^TX = 0. \tag{4}$$

to find the *critcal points* of Problem 2. If we then premultiply A^T and A with ∇_X and ∇_Y respectively, we obtain

$$(AA^T)X = X(Y^TYX^TX) \qquad (A^TA)Y = Y(X^TXY^TY). \tag{5}$$

Replacing (3) into (5) yields

$$(AA^T)PD^{\frac{1}{2}} = PDQ^TQDP^TPD^{\frac{1}{2}} \text{ and } (AA^T)QD^{\frac{1}{2}} = QDP^TPDQ^TQD^{\frac{1}{2}}.$$

When D is invertible, this finally yields

$$(AA^T)P = PD^2 \text{ and } (AA^T)Q = QD^2.$$

This shows that the columns of P and Q are singular vectors and the diagonal elements of D are nonzero singular values of A. Notice that if D is singular, one can throw away the corresponding columns of P and Q and reduce the problem to a smaller rank approximation with the same properties. Without loss of generality, we can therefore focus on approximations of Problem (2) which are of exact rank r. We can summarize the above arguments in the following well-known theorem.

Theorem 2. *Let $A \in \mathbb{R}^{m\times n}$ ($m > n$ and $rank(A) = t$). If A_r ($1 \leq r \leq t$) is a rank r critical point of Problem 2, then there exists two orthogonal matrices $P \in \mathbb{R}^{m\times m}$ and $Q \in \mathbb{R}^{n\times n}$ such that:*

$$A = P\hat{\Sigma}Q^T \text{ and } A_r = P\hat{\Sigma}_rQ^T$$

where

$$\hat{\Sigma} = \begin{pmatrix} \hat{\sigma}_1 & 0 & \dots & 0 \\ 0 & \hat{\sigma}_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \hat{\sigma}_n \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix}, \qquad \hat{\Sigma}_r = \begin{pmatrix} \hat{\sigma}_1 & 0 & \dots & 0 & \dots & 0 \\ 0 & \hat{\sigma}_2 & \dots & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & & \vdots \\ 0 & 0 & \dots & \hat{\sigma}_r & \dots & 0 \\ \vdots & \vdots & & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & \dots & 0 \end{pmatrix}$$

and the $\hat{\sigma}_i's$ are unsorted singular values of A. Moreover, the approximation error is:

$$\frac{1}{2}\|A - A_r\|_F^2 = \frac{1}{2}\sum_{i=r+1}^{t} \hat{\sigma}_i^2.$$

This result shows that if the singular values are all different, there are $\frac{n!}{r!(n-r)!}$ possible stationary points A_r. When there are multiple singular values, there are infinitely many stationary points A_r since there are infinitely many singular subspaces. The next result identifies the minima among all stationary points. This is also a known result but we rederive it here for the sake of completeness.

Theorem 3. *The only stable stationary points of Problem 2 are given by Theorem 1 and are global minima. All other stationary points are saddle points.*

Proof. Let us assume that A_r is a stationary point given by Theorem 2 but not by Theorem 1. Then there always exists a permutation of the columns of P and Q, and of the diagonal elements of $\hat{\Sigma}$ and $\hat{\Sigma}_r$ such that $\hat{\sigma}_{r+1} > \hat{\sigma}_r$. We then construct two points in the ϵ-neighborhood of A_r that yield an increase and a decrease, respectively, of the distance measure. They are obtained by taking:

$$\overline{\Sigma}_r(\epsilon) = \begin{pmatrix} \hat{\sigma}_1 + \epsilon & \dots & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \dots & \vdots \\ 0 & \dots & \hat{\sigma}_r & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & \dots & 0 \end{pmatrix}, \qquad \overline{A}_r(\epsilon) = P\overline{\Sigma}_r(\epsilon)Q^T$$

and

$$\underline{\Sigma}_r(\epsilon) = \begin{pmatrix} \hat{\sigma}_1 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \dots & \vdots \\ 0 & \dots & \hat{\sigma}_r & \epsilon\sqrt{\hat{\sigma}_r} & \vdots & 0 \\ 0 & \dots & \epsilon\sqrt{\hat{\sigma}_r} & \epsilon^2 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & \dots & 0 \end{pmatrix}, \qquad \underline{A}_r(\epsilon) = P\underline{\Sigma}_r(\epsilon)Q^T.$$

Clearly $\overline{A}_r(\epsilon)$ and $\underline{A}_r(\epsilon)$ are of rank r. Evaluating the distance measure yields

$$\begin{aligned} \|A - \underline{A}_r(\epsilon)\|_F^2 &= 2\hat{\sigma}_r\epsilon^2 + (\hat{\sigma}_{r+1} - \epsilon^2)^2 + \sum_{i=r+2}^{t} \hat{\sigma}_i^2 \\ &= \epsilon^2[\epsilon^2 - 2(\hat{\sigma}_{r+1} - \hat{\sigma}_r)] + \sum_{i=r+1}^{t} \hat{\sigma}_i^2 \\ &< \sum_{i=r+1}^{t} \hat{\sigma}_i^2 = \|A - A_r\|_F^2 \end{aligned}$$

for all $\epsilon \in (0, \sqrt{2(\hat{\sigma}_{r+1} - \hat{\sigma}_r)})$ and

$$\|A - \overline{A}_r(\epsilon)\|_F^2 = \epsilon^2 + \sum_{i=r+1}^{t} \hat{\sigma}_i^2 > \sum_{i=r+1}^{t} \hat{\sigma}_i^2 = \|A - A_r\|_F^2$$

for all $\epsilon > 0$. Hence, for an arbitrarily small positive ϵ, we obtain

$$\|A - \underline{A}_r(\epsilon)\|_F^2 < \|A - A_r\|_F^2 < \|A - \overline{A}_r(\epsilon)\|_F^2$$

which shows that A_r is a saddle point of the distance measure. □

When we add a nonnegativity constraint in the next section, the results of this section will help to identify stationary points at which all the nonnegativity constraints are inactive.

3 Nonnegativity constraint

In this section, we investigate the problem of Nonnegative Matrix Factorization. This problem differs Problem 2 in the previous section because of the additional nonnegativity constraints on the factors. We first discuss the effects of adding such a constraint. By doing so, the problem is no longer easy because of the existence of local minima at the boundary of the nonnegative orthant. Determining the lowest minimum among these minima is not easy. On the other hand, a minimum that coincides with a minimum of the unconstrained problem (i.e. Problem 2) may be easily reached by standard descent methods, as we will see.

We now list some of the standard methods for NMF and some general gradient schemes that can be applied to NMF. A discussion of the stopping conditions for those methods is then carried out. We first describe the NMF problem.

Problem (Nonnegative Matrix Factorization (NMF))**.** *Given a $m \times n$ nonnegative matrix A and an integer r, find*

$$\min_{U \in \mathbb{R}_+^{m \times r} \; V \in \mathbb{R}_+^{n \times r}} F(A, UV^T)$$

where $F(A, UV^T) = \frac{1}{2}\|A - UV^T\|_F^2$, and r is called the reduced rank.

The Karush-Kuhn-Tucker (KKT) conditions [6] for this NMF problem, expressed in terms of the variables U and V, are given by

$$U \geq 0 \quad , \quad V \geq 0, \tag{6}$$

$$\nabla F_U = UV^TV - AV \geq 0 \quad , \quad \nabla F_V = VU^TU - A^TU \geq 0, \tag{7}$$

$$U \circ (UV^TV - AV) = 0 \quad , \quad V \circ (VU^TU - A^TU) = 0. \tag{8}$$

where the corresponding Lagrange multipliers for U and V are also the gradient of F with respect to U and V.

Definition (NMF Solution)**.** *We call (U, V) a stationary point of the NMF Problem iff U and V satisfy the KKT conditions of the NMF optimization problem.*

There are two values of reduced rank r for which we can trivially identify the global solution which are $r = 1$ and $r = \min(m, n)$. For $r = 1$, a pair of

dominant singular vectors are a global minimizer. And for $r = \min(m, n)$, $(U = A, V = I)$ is a global minimizer. Since in this paper, we will focus on descent algorithms for the NMF problem, we should pay attention to all local minimizers. For the rank-one case, they can be easily pointed out.

3.1 Rank one approximation

The rank-one NMF problem of a nonnegative matrix A can be rewritten as

$$\min_{u \in \mathbb{R}^m_+ \; v \in \mathbb{R}^n_+} \frac{1}{2}\|A - uv^T\|_F^2 \tag{9}$$

and a complete analysis can be carried out. It is well known that any pair of nonnegative Perron vectors of AA^T and A^TA yields a global minimizer of this problem, but we can also show that the *only* stationary point of (9) is given by such vectors. The following theorem excludes the case where $u = 0$ and/or $v = 0$.

Theorem 4. *The pair (u, v) is a local minimizer of (9) if and only if u and v are nonnegative eigenvectors of AA^T and A^TA respectively of the eigenvalue $\sigma = \|u\|_2^2\|v\|_2^2$.*

Proof. The *if part* easily follows from Theorem 2. For the *only if part* we proceed as follows. Without loss of generality, we can permute the rows and columns of A such that the corresponding vectors u and v are partitioned as $(u_+ \; 0)^T$ and $(v_+ \; 0)^T$ respectively, where $u_+, \; v_+ > 0$. Partition the corresponding matrix A conformably as follows

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

then from (7) we obtain $A_{21}v_+ \leq 0$ and $A_{12}^T u_+ \leq 0$. Since A_{21} , $A_{12} \geq 0$ and $u_+, \; v_+ > 0$, we can conclude that $A_{12} = 0$ and $A_{21} = 0$. Then from (8) we have:

$$u_+ \circ (\|v_+\|_2^2 u_+ - A_{11}v_+) = 0 \text{ and } v_+ \circ (\|u_+\|_2^2 v_+ - A_{11}^+ u_+) = 0.$$

Since $u_+, \; v_+ > 0$, we have:

$$\|v_+\|_2^2 u_+ = A_{11}v_+ \text{ and } \|u_+\|_2^2 v_+ = A_{11}^T u_+$$

or

$$\|u_+\|_2^2\|v_+\|_2^2 u_+ = A_{11}A_{11}^T u_+ \text{ and } \|u_+\|_2^2\|v_+\|_2^2 v_+ = A_{11}^T A_{11} v_+.$$

Setting $\sigma = \|u_+\|_2^2\|v_+\|_2^2$ and using the block *diagonal* structure of A yields the desired result. □

Theorem 4 guarantees that all stationary points of the rank-one case are nonnegative singular vectors of a submatrix of A. These results imply that a global minimizer of the rank-one NMF can be calculated correctly based on the largest singular value and corresponding singular vectors of the matrix A.

3.2 Solution Characteristics

The KKT conditions (8) help to characterize the NMF solutions. Summing up all the elements of one of the conditions (8), we get:

$$\begin{aligned} 0 &= \sum_{ij} \big(U \circ (UV^TV - AV)\big)_{ij} \\ &= \big\langle U, UV^TV - AV \big\rangle \\ &= \big\langle UV^T, UV^T - A \big\rangle . \end{aligned} \tag{10}$$

From that, we have some simple characteristics of the NMF solutions:

Theorem 5. *Let (U,V) be a NMF solution, then $UV^T \in \mathcal{B}\left(\frac{A}{2}, \frac{1}{2}\|A\|_F\right)$, the ball centered at $\frac{A}{2}$ and with radius $= \frac{1}{2}\|A\|_F$.*

Proof. From (10) it immediately follows that

$$\left\langle \frac{A}{2} - UV^T, \frac{A}{2} - UV^T \right\rangle = \left\langle \frac{A}{2}, \frac{A}{2} \right\rangle$$

which implies

$$UV^T \in \mathcal{B}\left(\frac{A}{2}, \frac{1}{2}\|A\|_F\right).$$

□

Theorem 6. *Let (U,V) be a NMF solution, then*

$$\frac{1}{2}\|A - UV^T\|_F^2 = \frac{1}{2}(\|A\|_F^2 - \|UV^T\|_F^2)$$

.

Proof. From (10), we have $\big\langle UV^T, A \big\rangle = \big\langle UV^T, UV^T \big\rangle$. Therefore,

$$\begin{aligned} \frac{1}{2}\left\langle A - UV^T, A - UV^T \right\rangle &= \frac{1}{2}(\|A\|_F^2 - 2\left\langle UV^T, A \right\rangle + \|UV^T\|_F^2) \\ &= \frac{1}{2}(\|A\|_F^2 - \|UV^T\|_F^2). \end{aligned}$$

□

Theorem 6 also suggests that at a solution (U, V) of the NMF problem, we should have $\|A\|_F^2 \geq \|UV^T\|_F^2$. This norm inequality can be also found in [6] for less general cases where we have $\nabla F_U = 0$ and $\nabla F_V = 0$ at a stationary point. For this particular class of NMF solutions, the two factors U and V are positive. And all such stationary points are also stationary points of the unconstrained problem, characterized by Theorem 2.

We have seen in Theorem 2 that, for the unconstrained least-square problem the only stable stationary points are in fact global minima. Therefore, if the stationary points of the constrained problem are inside the nonnegative orthant (i.e. all constraints are inactive), we can then probably reach the global minimum of the NMF problem. This can be expected because the constraints may no longer prohibit the descent of the update.

Let A_r be the optimal rank-r approximation of a nonnegative matrix A, which we obtain from the singular value decomposition, as indicated in Theorem 2. Then we can easily construct its nonnegative part $[A_r]_+$, which is obtained from A_r by just setting all its negative elements equal to zero. This is in fact the closest matrix in the cone of nonnegative matrices to the matrix A_r, in the Frobenius norm (in that sense, it is its projection on that cone). We now derive some bounds for the error $\|A - [A_r]_+\|_F$.

Theorem 7. *Let A_r be the best rank r approximation of a nonnegative matrix A, and let $[A_r]_+$ be its nonnegative part, then*

$$\|A - [A_r]_+\|_F \leq \|A - A_r\|_F.$$

Proof. This follows easily from the convexity of the cone of nonnegative matrices. Since both A and $[A_r]_+$ are nonnegative and since $[A_r]_+$ is the closest matrix in that cone to A_r we immediately obtain the inequality

$$\|A - A_r\|_F^2 \geq \|A - [A_r]_+\|_F^2 + \|A_r - [A_r]_+\|_F^2 \geq \|A - [A_r]_+\|_F^2$$

from which the result readily follows. □

The approximation $[A_r]_+$ has the merit of requiring as much storage as a rank r approximation, even though its rank is larger than r whenever $A_r \neq [A_r]_+$. We will look at the quality of this approximation in Section 8. If we now compare this bound with the nonnegative approximations then we obtain the following inequalities. Let $U_* V_*^T$ be an optimal nonnegative rank r approximation of A and let UV^T be any stationary point of the KKT conditions for a nonnegative rank r approximation, then we have :

$$\|A - [A_r]_+|_F^2 \leq \|A - A_r\|_F^2 = \sum_{i=r+1}^{n} \sigma_i^2 \leq \|A - U_* V_*^T\|_F^2 \leq \|A - UV^T\|_F^2.$$

4 Existing descent algorithms

We focus on descent algorithms that guarantee a non increasing update at each iteration. Based on the search space, we have two categories: *Full-space search* and *(Block) Coordinate search*.

Algorithms in the former category try to find updates for both U and V at the same time. This requires a search for a descent direction in the $(m+n)r$-dimensional space. Note also that the NMF problem in this full space is not convex but the optimality conditions may be easier to achieve.

Algorithms in the latter category, on the other hand, find updates for each (block) coordinate in order to guarantee the descent of the objective function. Usually, search subspaces are chosen to make the objective function convex so that efficient methods can be applied. Such a simplification might lead to the loss of some convergence properties. Most of the algorithms use the following column partitioning:

$$\frac{1}{2}\|A - UV^T\|_F^2 = \frac{1}{2}\sum_{i=1}^{n}\|A_{:,i} - U(V_{i,:})^T\|_2^2, \tag{11}$$

which shows that one can minimize with respect to each of the rows of V independently. The problem thus decouples into smaller convex problems. This leads to the solution of quadratic problems of the form

$$\min_{v\geq 0}\frac{1}{2}\|a - Uv\|_2^2. \tag{12}$$

Updates for the rows of V are then alternated with updates for the rows of U in a similar manner by transposing A and UV^T.

Independent on the search space, most of algorithms use the Projected Gradient scheme for which three basic steps are carried out in each iteration:

- Calculating the gradient $\nabla F(x^k)$,
- Choosing the step size α^k,
- Projecting the update on the nonnegative orthant

$$x^{k+1} = [x^k - \alpha^k \nabla F(x^k)]_+,$$

where x^k is the variable in the selected search space. The last two steps can be merged in one iterative process and must guarantee a sufficient decrease of the objective function as well as the nonnegativity of the new point.

4.1 Multiplicative rules (Mult)

Multiplicative rules were introduced in [17]. The algorithm applies a block coordinate type search and uses the above column partition to formulate the updates. A special feature of this method is that the step size is calculated for each element of the vector. For the elementary problem (12) it is given by

$$v^{k+1} = v^k - \alpha^k \circ \nabla F(v^{k+1}) = v^k \circ \frac{\left[U^T a\right]}{\left[U^T U v^k\right]}$$

where $[\alpha^k]_i = \frac{v_i}{[U^T U v]_i}$. Applying this to all rows of V and U gives the updating rule of Algorithm 1 to compute

$$(U^*, V^*) = \underset{U \geq 0 \ V \geq 0}{\operatorname{argmin}} \|A - UV^T\|_F^2.$$

Algorithm 1 (Mult)

1: Initialize U^0, V^0 and $k = 0$
2: **repeat**
3: $\quad U^{k+1} = U^k \circ \frac{\left[AV^k\right]}{\left[U^k (V^k)^T (V^k)\right]}$
4: $\quad V^{k+1} = V^k \circ \frac{\left[A^T U^{k+1}\right]}{\left[V^k (U^{k+1})^T (U^{k+1})\right]}$
5: $\quad k = k + 1$
6: **until** Stopping condition

These updates guarantee automatically the nonnegativity of the factors but may fail to give a sufficient decrease of the objective function. It may also get stuck in a non-stationary point and hence suffer from a poor convergence. Variants can be found in [18, 21].

4.2 Line search using Armijo criterion (Line)

In order to ensure a sufficient descent, the following projected gradient scheme with Armijo criterion [20, 19] can be applied to minimize

$$x^* = \underset{x}{\operatorname{argmin}} F(x).$$

Algorithm 2 needs two parameters σ and β that may affect its convergence. It requires only the gradient information, and is applied in [20] for two different strategies : for the whole space (U, V) (Algorithm FLine) and for U and V separately in an alternating fashion (Algorithm CLine). With a good choice of parameters ($\sigma = 0.01$ and $\beta = 0.1$) and a good strategy of alternating between variables, it was reported in [20] to be the faster than the multiplicative rules.

Algorithm 2 (Line)

Initialize x^0, σ, β, $\alpha_0 = 1$ and $k = 1$
repeat
  $\alpha_k = \alpha_{k-1}$
  $y = [x^k - \alpha_k \nabla F(x^k)]_+$
  if $F(y) - F(x^k) > \sigma \left\langle \nabla F(x^k), y - x^k \right\rangle$ **then**
    repeat
      $\alpha_k = \alpha_k \cdot \beta$
      $y = [x^k - \alpha_k \nabla F(x^k)]_+$
    until $F(y) - F(x^k) \leq \sigma \left\langle \nabla F(x^k), y - x^k \right\rangle$
  else
    repeat
      $lasty = y$
      $\alpha_k = \alpha_k / \beta$
      $y = [x^k - \alpha_k \nabla F(x^k)]_+$
    until $F(y) - F(x^k) > \sigma \left\langle \nabla F(x^k), y - x^k \right\rangle$
    $y = lasty$
  end if
  $x^{k+1} = y$
  $k = k + 1$
until Stopping condition

4.3 Projected gradient with first-order approximation (FO)

In order to find the solution to

$$x^* = \operatorname*{argmin}_x F(x)$$

we can also approximate at each iteration the function $F(X)$ using:

$$\tilde{F}(x) = F(x^k) + \left\langle \nabla_x F(x^k), x - x^k \right\rangle + \frac{L}{2} \|x^k - x\|_2^2,$$

where L is a Lipshitz constant satisfying $F(x) \leq \tilde{F}(x),\ \forall x$. Because of this inequality, the solution of the following problem

$$x_{k+1} = \operatorname*{argmin}_{x \geq 0} \tilde{F}(x)$$

also is a point of descent for the function $F(x)$ since

$$F(x_{k+1}) \leq \tilde{F}(x_{k+1}) \leq \tilde{F}(x_k) = F(x_k).$$

Since the constant L is not known a priori, an inner loop is needed. Algorithm 3 presents an iterative way to carry out this scheme. As in the previous

algorithm this also requires only the gradient information and can therefore can be applied to two different strategies: to the whole space (U, V) (Algorithm FFO) and to U and V separately in an alternating fashion (Algorithm CFO).

Algorithm 3 (FO)

Initialize x^0, L_0 and $k = 0$
repeat
  $y = [x^k - \frac{1}{L_k}\nabla F(x^k)]_+$
  while $F(y) - F(x^k) > \langle \nabla F(x^k), y - x^k \rangle + \frac{L_k}{2}\|y - x^k\|_2^2$ **do**
    $L_k = L_k/\beta$
    $Y = [x^k - \frac{1}{L_k}\nabla F(x^k)]_+$
  end while
  $x^{k+1} = y$
  $L_{k+1} = L_k \cdot \beta$
  $k = k + 1$
until Stopping condition

A main difference with the previous algorithm is its stopping criterion for the inner loop. This algorithm requires also a parameter β for which the practical choice is 2.

5 Alternative least squares methods

The first algorithm proposed for solving the nonnegative matrix factorization was the alternative least squares method [22]. It is known that, fixing either U or V, the problem becomes a least squares problem with nonnegativity constraint.

Algorithm 4 Alternative Least Square (ALS)

Initialize U and V
repeat
  Solve: $\min_{V \geq 0} \frac{1}{2}\|A - UV^T\|_F^2$
  Solve: $\min_{U \geq 0} \frac{1}{2}\|A^T - VU^T\|_F^2$
until Stopping condition

Since the least squares problems in Algorithm 4 can be perfectly decoupled into smaller problems corresponding to the columns or rows of A, we can directly apply methods for the Nonnegative Least Square problem to each of the small problem. Methods that can be applied are [16], [5], etc.

5.1 Implementation

As mentioned at the beginning of the section, the above methods can be carried out using two different strategies: full space search or coordinate search. In some cases, it is required to evaluate repeatedly the function $F(U, V)$. We mention here how to do this efficiently with the coordinate search.

Full space search: The exact evaluation of $F(x) = F(U, V) = \|A - UV^T\|_F^2$ need $O(mnr)$ operations. When there is a correction $y = (U + \Delta U, V + \Delta V)$, we have to calculate $F(y)$ which also requires $O(mnr)$ operations. Hence, it requires $O(tmnr)$ operations to determine a stepsize in t iterations.

Coordinate search: when V is fixed, the Euclidean distance is a quadratic function on U:

$$\begin{aligned} F(U) = \|A - UV^T\|_F^2 &= \langle A, A\rangle - 2\left\langle UV^T, A\right\rangle + \left\langle UV^T, UV^T\right\rangle \\ &= \|A\|_F^2 - 2\langle U, AV\rangle + \left\langle U, U(V^TV)\right\rangle. \end{aligned}$$

The most expensive operator is the computation of AV, which requires $O(mnr)$ operations. But when V is fixed, AV can be calculated once at the beginning of the inner loop. The remaining computation are $\langle U, AV\rangle$ and $\left\langle U, U(V^TV)\right\rangle$, which requires $O(nr)$ and $O(nr^2 + nr)$. Therefore, it requires $O(tnr^2)$ operations to determine a stepsize in t iterations which is much less than $O(tmnr)$ operations. This is due to the assumption $r \ll n$. Similarly, when U fixed, $O(tmr^2)$ operations are needed to determine a stepsize.

If we consider an iteration is a sweep, i.e., once all the variables are updated, the following table summarize the complexity of each sweep of described algorithms:

Algorithm	Complexity per iteration
Mult	$O(mnr)$
FLine	$O(tmnr)$
CLine	$O(t_1nr^2 + t_2mr^2)$
FFO	$O(tmnr)$
CFO	$O(t_1nr^2 + t_2mr^2)$
ALS	$O(2^rmnr)$
IALS	$O(mnr)$

where t, t_1 and t_2 are the number of iterations of inner loops, which can not be bounded in general. For algorithm ALS, if the active set method is used then it can be proved to be finite [16]. $O(2^rmnr)$ is an *theoretical* upper bound that count all the possible subsets of r variables of each subproblem.

5.2 Scaling and Stopping criterion

For descent methods, several stopping conditions are used in the literature. We now discuss some problems when implementing these conditions for NMF.

The very first condition is the decrease of the objective function. The algorithm should stop when it fails to make the objective function decrease with a certain amount :

$$F(U^{k+1},V^{k+1}) - F(U^k,V^k) < \epsilon \quad or \quad \frac{F(U^{k+1},V^{k+1}) - F(U^k,V^k)}{F(U^k,V^k)} < \epsilon.$$

This is not a good choice for all cases since the algorithm may stop at a point very far from a stationary point. Time and iteration bounds can also be imposed for very slowly converging algorithms. But here again this may not be good for the optimality conditions. A better choice is probably the norm of the projected gradient as suggested in [20]. For the NMF problem it is defined as follows :

$$[\nabla_X^P]_{ij} = \begin{cases} [\nabla_X]_{ij} & \text{if } X_{ij} > 0 \\ \min(0,[\nabla_X]_{ij}) & \text{if } X_{ij} = 0 \end{cases}$$

where X stands for U or V. The proposed condition then becomes

$$\left\| \begin{pmatrix} \nabla_{U^k}^P \\ \nabla_{V^k}^P \end{pmatrix} \right\|_F \leq \epsilon \left\| \begin{pmatrix} \nabla_{U^1} \\ \nabla_{V^1} \end{pmatrix} \right\|_F. \tag{13}$$

We should also take into account the scaling invariance between U and V. Putting $\bar{U} = \gamma U$ and $\bar{V} = \frac{1}{\gamma} V$ does not change the approximation UV^T but the above projected gradient norm is affected:

$$\begin{aligned} \left\| \begin{pmatrix} \nabla_{\bar{U}}^P \\ \nabla_{\bar{V}}^P \end{pmatrix} \right\|_F^2 &= \|\nabla_{\bar{U}}^P\|_F^2 + \|\nabla_{\bar{V}}^P\|_F^2 = \frac{1}{\gamma^2}\|\nabla_U^P\|_F^2 + \gamma^2\|\nabla_V^P\|_F^2 \qquad (14) \\ &\neq \left\| \begin{pmatrix} \nabla_U^P \\ \nabla_V^P \end{pmatrix} \right\|_F^2. \end{aligned}$$

Two approximate factorizations $UV^T = \bar{U}\bar{V}^T$ resulting in the same approximation should be considered equivalent in terms of precision. One could choose $\gamma^2 := \|\nabla_U^P\|_F / \|\nabla_V^P\|_F$, which minimizes (14) and forces $\|\nabla_{\bar{U}}^P\|_F = \|\nabla_{\bar{V}}^P\|_F$, but this may not be a good choice when only one of the gradients $\|\nabla_{\bar{U}}^P\|_F$ and $\|\nabla_{\bar{V}}^P\|_F$ is nearly zero.

In fact, the gradient $\begin{pmatrix} \nabla_U \\ \nabla_V \end{pmatrix}$ is scale dependent in the NMF problem and any stopping criterion that uses gradient information is affected by this

scaling. To limit that effect, we suggest the following scaling after each iteration:

$$\tilde{U}_k \leftarrow U_k D_k \qquad \tilde{V}_k \leftarrow V_k D_k^{-1}$$

where D_k is a positive diagonal matrix:

$$[D_k]_{ii} = \sqrt{\frac{\|V_{:i}\|_2}{\|U_{:i}\|_2}}.$$

This ensures that $\|\tilde{U}\|_F^2 = \|\tilde{V}\|_F^2$ and hopefully reduces also the difference between $\|\nabla^P_{\tilde{U}}\|_F^2$ and $\|\nabla^P_{\tilde{V}}\|_F^2$. Moreover, it may help to avoid some numerically unstable situations.

6 Rank-one Residue Iteration

In the previous section, we have seen that it is very appealing to decouple the problem into convex subproblems. But this may "converge" to solutions that are far from the global minimizers of the problem.

In this section, we analyze a different decoupling of the problem based on rank one approximations. This also allows us to formulate a very simple basic subproblem. This scheme has several advantages : the subproblems can be solved in closed form, they seem to have useful convergence results and they can be extended to more general types of factorizations such as for nonnegative tensors. Moreover, the experiments in Section 8 suggest that this method outperforms the other ones in most cases.

6.1 New partition of variables

Let u_i's and v_i's be respectively the columns of U and V. Then the NMF problem can be rewritten as follows :

Problem (Nonnegative Matrix Factorization)**.** *Given a $m \times n$ nonnegative matrix A, solve*

$$\min_{u_i \geq 0 \ v_i \geq 0} \frac{1}{2}\|A - \sum_{i=1}^{r} u_i v_i^T\|_F^2.$$

Let us fix all the variables, except for a single vector v_t and consider the following least squares problem:

$$\min_{v \geq 0} \frac{1}{2}\|R_t - u_t v^T\|_F^2, \tag{15}$$

where $R_t = A - \sum_{i \neq t} u_i v_i^T$. We have:

$$\begin{aligned} \|R_t - u_t v^T\|_F^2 &= trace\left[(R_t - u_t v^T)^T (R_t - u_t v^T)\right] \quad (16) \\ &= \|R_t\|_F^2 - 2v^T R_t^T u_t + \|u_t\|_2^2 \|v\|_2^2. \quad (17) \end{aligned}$$

From this formulation, one now derives the following lemma.

Lemma 1. *If* $[R_t^T u_t]_+ \neq 0$, *then* $v_* := \frac{[R_t^T u_t]_+}{\|u_t\|_2^2}$ *is the unique global minimizer of (15) and the function value equals* $\|R_t\|_F^2 - \frac{\|[R_t^T u_t]_+\|_2^2}{\|u_t\|_2^2}$.

Proof. Let us permute the elements of the vectors $x := R_t^T u_t$ and v such that

$$Px = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \quad Pv = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}, \quad \text{with} \quad x_1 \geq 0, \quad x_2 < 0$$

and P is the permutation matrix. Then

$$\|R_t - u_t v^T\|_F^2 = \|R_t\|_F^2 - 2v_1^T x_1 - 2v_2^T x_2 + \|u_t\|_2^2 (v_1^T v_1 + v_2^T v_2).$$

Since $x_2 < 0$ and $v_2 \geq 0$, it is obvious that $\|R_t - u_t v^T\|_F^2$ can only be minimal if $v_2 = 0$. Our assumption implies that x_1 is nonempty and $x_1 > 0$. One can then find the optimal v_1 by minimizing the remaining quadratic function

$$\|R_t\|_F^2 - 2v_1^T x_1 + \|u_t\|_2^2 v_1^T v_1$$

which yields the solution $v_1 = \frac{x_1}{\|u_t\|_2^2}$. Putting the two components together yields the result

$$v_* = \frac{[R_t^T u_t]_+}{\|u_t\|_2^2} \quad \text{and} \quad \|R_t - u_t v_*^T\|_F^2 = \|R_t\|_F^2 - \frac{\|[R_t^T u_t]_+\|_2^2}{\|u_t\|_2^2}.$$

□

Remark 1: In the case where $[R_t^T u_t]_+ = 0$, we have a trivial solution for $v = 0$ that is not covered by Lemma 1. This will result in a rank-deficient approximation. To avoid that, one can replace $u_t v_t^T$ by any rank one approximation that reduces the norm of the error matrix, such as

$$u_t = e_{i^*} \qquad v_t = [R_t^T u_t]_+$$

where $i^* = \text{argmax}_i \|[R_t^T e_i]_+\|$ and $e_i = (0\ 0\ \ldots \underbrace{1}_{i-th\ position} \ldots\ 0)^T$.

Remark 2: The above lemma has of course a dual form, where one fixes v_t but solves for the optimal u to minimize $\|R_t - u v_t^T\|_F^2$. These lemmas would yield the updating rules

$$v_t \leftarrow \frac{[R_t^T u_t]_+}{\|u_t\|_2^2} \quad \text{and} \quad u_t \leftarrow \frac{[R_t v_t]_+}{\|v_t\|_2^2} \quad (18)$$

which can be used to recursively update approximations $\sum_{i=1}^{r} u_i v_i^T$ by modifying each rank-one matrix $u_t v_t^T$ in a cyclic manner. This problem is different from the NMF, since the error matrices $R_t = A - \sum_{i \neq t} u_i v_i^T$ are no longer nonnegative. We will therefore call this method the *Rank-one Residue Iteration* (RRI).

Algorithm 5 (RRI)

1: Initialize u_i's, v_i's, for $i = 1$ to r
2: **repeat**
3: **for** $i = 1$ to r **do**
4: $R_i = A - \sum_{j \neq i} u_j v_j^T$
5: $v_i \leftarrow \frac{[R_i^T u_i]_+}{\|u_i\|_2^2}$
6: $u_i \leftarrow \frac{[R_i v_i]_+}{\|v_i\|_2^2}$
7: **end for**
8: **until** Stopping condition

Remark 3: Notice that the norm of the new rank one update is bounded since $\|u_t v_*^T\|_F = v_*^T u_t = ([u_t^T R_t]_+ u_t)/\|u_t\|_2^2 \leq \|R_t\|_F$. One can moreover scale the vector pairs (u_i, v_i) at each stage as explained in Section 4 without affecting the local optimality of Lemma 1. Since the optimality of Lemma 1 implies that R_t can not increase, it then follows that the rank one products $u_i v_i^T$ and their scaled vectors remain bounded.

6.2 Convergence

In the previous section, we have established the partial updates for each of the variable u_i or v_i. And for a NMF problem where the reduced rank is r, we have in total $2r$ vector variables (the u_i's and v_i's). The described algorithm can be also considered as a projected gradient method since the update (18) can be rewritten as:

$$\begin{aligned} u_t &\leftarrow \frac{[R_t v_t]_+}{\|v_t\|_2^2} = \frac{[(A - \sum_{i \neq t} u_i v_i^T) v_t]_+}{\|v_t\|_2^2} = \frac{[(A - \sum_i u_i v_i^T + u_t v_t^T) v_t]_+}{\|v_t\|_2^2} \\ &= \frac{[(A - \sum_i u_i v_i^T) v_t + u_t v_t^T v_t]_+}{\|v_t\|_2^2} = \left[u_t - \frac{1}{\|v_t\|_2^2} \nabla_{u_t} \right]_+ . \end{aligned}$$

Similarly, the update for v_i can be rewritten as

$$v_t \leftarrow \left[v_t - \frac{1}{\|u_t\|_2^2} \nabla_{v_t} \right]_+ .$$

Therefore, the new method follows the projected gradient scheme described in the previous section. But it produces the optimal solution in closed form.

Because at each step of the $2r$ basic steps of Algorithm 5 we compute an optimal rank-one nonnegative correction to the corresponding error matrix R_t the Frobenius norm of the error can not increase. This is a reassuring property but it does not imply convergence of the algorithm.

Each vector u_i or v_i lies in a convex set $\mathbb{U}_i \subset \mathbb{R}_+^m$ or $\mathbb{V}_i \subset \mathbb{R}_+^n$. Moreover, because of the possibility to include scaling we can set an upper bound for $\|U\|$ and $\|V\|$, in such a way that all the set $\mathbb{U}_i$'s and $\mathbb{V}_i$'s can be considered as closed convex. Then, we can use the following Theorem 8, to prove a stronger convergence result for Algorithm 5.

Theorem 8. *Every limit point generated by Algorithm 5 is a stationary point.*

Proof. Clearly $\frac{1}{2}\|A - \sum_{i=1}^r u_i v_i^T\|_F^2$ is continuously differentiable over the set $\mathbb{U}_1 \times \ldots \times \mathbb{U}_r \times \mathbb{V}_1 \times \ldots \times \mathbb{V}_r$, and the $\mathbb{U}_i$'s and $\mathbb{V}_i$'s are closed convex. Moreover, the unique global solution for each of the elementary problem is attained. Applying Proposition 2.7.1 in [4] gives the desired result. □

We can see that, in Algorithm 5, the variables are updated in the order: v_1, u_1, v_2, u_2, ... We can alternate the variables in a different order as well, for example v_1, v_2, ..., v_r u_1, u_2, ..., u_r, ... Unless this is carried out in a cyclic fashion, the Theorem 8 still holds and this does not increase the complexity of each iteration of the algorithm.

For each update of a column v_i (or u_i), the proposed algorithm requires just a matrix-vector multiplication $R_i^T u_i$ (or $R_i v_i$), wherein the residue matrix $R_i = A - \sum_{j \neq i} u_j v_j^T$ does not have to be calculated explicitly. Indeed, by calculating $R_i^T u_i$ (or $R_i v_i$) from $A^T u_i$ (or $A v_i$) and $\sum_{j \neq i} v_j (u_j^T u_i)$ (or $\sum_{j \neq i} u_j (v_j^T v_i)$), the complexity is reduced from $O(mnr + mn)$ to only $O\left(mn + (m+n)(r-1)\right)$ which is majored by $O(mn)$. This implies that the complexity of each sweep through the $2r$ variables $u_i's$ and $v_i's$ requires only $O(mnr)$ operations, which is equivalent to a sweep of the multiplicative rules and to an inner loop of any gradient methods. This is very low since the evaluation of the whole gradient requires already the same complexity.

6.3 Variants of the RRI method

We now extend the Reduced Rank-one Iteration by using a factorization of the type XDY^T where D is diagonal and nonnegative and the columns of the nonnegative matrices X and Y are normalized. The NMF formulation then becomes

$$\min_{\substack{x_i \in \mathbb{X}_i \; y_i \in \mathbb{Y}_i \\ d_i \in \mathbb{R}_+}} \frac{1}{2}\|A - \sum_{i=1}^r d_i x_i y_i^T\|_F^2,$$

where $\mathbb{X}_i$'s and $\mathbb{Y}_i$'s are sets of normed vectors. First note that our

The variants that we present here depend on the choice of $\mathbb{X}_i$'s and $\mathbb{Y}_i$'s. A generalized Reduced Rank-one Iteration method for low-rank approximation is given in Algorithm 6. This algorithm needs to solve a sequence of elementary problems of the type :

$$\max_{s \in \mathbb{S}} y^T s \tag{19}$$

where $y \in \mathbb{R}^n$ and $\mathbb{S} \subset \mathbb{R}^n$ is a set of normed vectors. We first introduce a permutation vector $I_y = (i_1\ i_2\ \ldots\ i_n)$ which reorders the elements of y in non-increasing order : $y_{i_k} \geq y_{i_{k+1}}$, $k = 1 \ldots (n-1)$. The function $p(y)$ returns the number of positive entries of y.

Algorithm 6 GRRI

1: Initialize x_i's, y_i's and d_i's
2: **repeat**
3: **for** $i = 1$ to r **do**
4: $R_i = A - \sum_{j \neq i} d_j x_j y_j^T$
5: $y_i \leftarrow \text{argmax}_{s \in \mathbb{Y}_i} \left(x_i^T R_i s \right)$
6: $x_i \leftarrow \text{argmax}_{s \in \mathbb{X}_i} \left(y_i^T R_i^T c \right)$
7: $d_i = x_i^T R_i y_i$
8: **end for**
9: **until** Stopping condition

Let us first point out that for the set of normed nonnegative vectors the solution of problem (19) is given by $s^* = \frac{y_+}{\|y_+\|_2}$. It then follows that Algorithm 6 is essentially the same as Algorithm 5 since the solutions v_i and u_i of each step of Algorithm 6, given by (18), correspond exactly to those of problem (19) via the relations $y_i = u_i/\|u_i\|_2$, $y_i = v_i/\|v_i\|_2$ and $d_i = \|u_i\|_2 \|v_i\|_2$.

Below we list the sets for which the solution s^* of (19) can be easily computed.

- *Set of normed nonnegative vectors* : $s = \frac{y_+}{\|y_+\|_2}$.
- *Set of normed binary vectors* $\{s\}$: where $s = \frac{b}{\|b\|}$ and $b \in \{0,1\}^n$. The optimal solution of (19) is given by:

$$[s^*]_{i_t} = \begin{cases} \frac{1}{\sqrt{k^*}} & \text{if } t \leq k^* \\ 0 & \text{otherwise} \end{cases} \qquad \text{where } k^* = \underset{k}{\text{argmax}} \frac{\sum_{t=1}^{k} y_{i_t}}{\sqrt{k}}.$$

- *Set of normed sparse nonnegative vectors*: all normed nonnegative vectors having at most K nonzero entries. The optimal solution for

(19) is given by norming the following vector p^*

$$[p^*]_{i_t} = \begin{cases} y_{i_t} & \text{if } t \leq \min(p(y), K) \\ 0 & \text{otherwise} \end{cases}$$

- *Set of normed fixed-sparsity nonnegative vectors*: all nonnegative vectors s a fixed sparsity ψ_0

$$sparsity(s) = \frac{\sqrt{n} - \|s\|_1/\|s\|_2}{\sqrt{n} - 1} = \psi_0.$$

  The optimal solution for (19) is given by using the projection scheme in [14].

One can also imagine other variants, for instance by combining the above ones. Depending on how data need to be approximated, one can create new algorithms provided it is relatively simple to solve problem (19). There have been some particular ideas in the literatures such as NMF with sparseness constraint [14], Semi discrete Matrix Decomposition [15] and Semi-Nonnegative Matrix Factorization [9] for which variants of the above scheme can offer an alternative choice of algorithm.

Remark: Not all the above sets are the normed version of a closed convex set, as required by Theorem 8. Therefore the algorithms might not converge to a local minimum. However, the algorithm always guarantees a non-increasing update even in those cases and can therefore be expected to return a *good* approximation.

6.4 Nonnegative Tensor Factorization

If we refer to the problem of finding the nearest nonnegative vector to a given vector a as the nonnegative approximation in one dimension, the NMF is its generalization in two dimensions and naturally, it can be extended to even higher-order tensor approximation problems. Algorithms described in the previous section use the closed form solution of the one dimensional problem to solve the two dimensional problem. We now generalize this to higher orders. Since in one dimension such an approximation is easy to construct, we continue to use this approach to build the solutions for higher order problems.

For a low-rank tensor, there are two popular kinds of factored tensors, namely those of Tucker and Kruskal [2]. We only give only an algorithm for finding approximations of Kruskal type. It is easy to extend this to tensors of Tucker type, but this is omitted here.

Given a d dimensional tensor T, we will derive an algorithm for approximating a nonnegative tensor by a rank-r nonnegative Kruskal tensor $S \in \mathbb{R}^{n_1 \times n_2 \times \ldots \times n_d}$ represented as a sum of r rank-one tensors:

$$S = \sum_{i=1}^{r} \sigma_i u_{1i} \star u_{2i} \star \ldots \star u_{di}$$

where $\sigma_i \in \mathbb{R}_+$ is a scaling factor, $u_{ti} \in \mathbb{R}_+^{n_t}$ is a normed vector (i.e. $\|u_{ti}\|_2 = 1$) and $a \star b$ stands for the outer product between two vectors or tensors a and b.

The following update rules are the generalization of the matrix case to the higher order tensor:

$$y = (\ldots((\ldots(R_k u_{1k})\ldots u_{(t-1)k})u_{(t+1)k})\ldots)u_{dk} \tag{20}$$

$$\sigma_k = \|[y]_+\|_2, \quad u_{tk} = \frac{[y]_+}{\sigma_k}, \tag{21}$$

where $R_k = T - \sum_{i \neq k} \sigma_i u_{1i} \star u_{2i} \star \ldots \star u_{di}$ is the residue tensor calculated without the k^{th} component of S and $R_k u_{ij}$ is the ordinary tensor/vector product in the corresponding dimension.

We can then produce an algorithm which updates in a cyclic fashion all vectors u_{ji}. Since each of these updates computes a unique optimal solution for u_{ij} of a constrained minimization problem over a closed convex set, Theorem 8 guarantees that each limit point of this algorithm is a local minimizer of the objective function.

Again, as we have seen in the previous section, we can extend the procedure to take into account different constraints on the vectors u_{ij} such as discreteness, sparseness, etc.

7 Regularizations

Regularization is a common method to solve multi-criterion problems. One may need to minimize two or more cost functions at the same time. To attack this problem, it is possible to minimize a combination between the functions and vary their positive weights to control the trade-off between them. We will list here the update for u_i and v_i when some simple regularizations are added to the original cost function. The proof of these updates are straight-forward and hence omitted.

- One-Norm $\|.\|_1$ regularization: the one-norm of the vector variable can be added as a heuristic for finding a sparse solution. This is an alternative to the fixed-sparsity variant presented above. The regularized

cost function with respect to the variable v_i will be

$$\frac{1}{2}\|R_i - u_i v^T\|_F^2 + \beta\|v\|_1, \quad \beta > 0$$

where the optimal update is given by

$$v^* = \frac{[R_i^T u_i - \beta \mathbf{1}_{n\times 1}]_+}{\|u_i\|_2^2}.$$

The constant $\beta > 0$ can be varied to control the trade-off between the approximation error $\frac{1}{2}\|R_i - u_i v^T\|_F^2$ and $\|v\|_1$. One then sees that this modification tends to zero out elements of $R_i^T u_i$ which are smaller than β, hence reducing the number of nonzero elements of v^*.

- Two-Norm $\|.\|_2$ regularization: We usually add the two-norm regularization to minimize the approximation error $\frac{1}{2}\|R_i - u_i v^T\|_F^2$ while keeping the norm of the solution $\|v^*\|_2$ small. The regularized cost function is

$$\frac{1}{2}\|R_i - u_i v^T\|_F^2 + \frac{\gamma}{2}\|v\|_2^2, \quad \gamma > 0$$

where the optimal regularized solution is given by

$$v^* = \frac{[R_i^T u_i]_+}{\|u_i\|_2^2 + \gamma} \leq \frac{[R_i^T u_i]_+}{\|u_i\|_2^2}.$$

The constant $\gamma \geq 0$ can be varied to control the trade-off between the approximation error $\frac{1}{2}\|R_i - u_i v^T\|_F^2$ and $\|v\|_2^2$. One can see that its effect is to scale down the optimal update v^* by a factor of $\frac{\|u_i\|_2^2}{\|u_i\|_2^2+\gamma}$.

- Smoothness regularization $\|v - B\hat{v}_i\|_F^2$: here $\hat{v}_i$ is the current value of v_i and the matrix B averages the neighboring elements of each element of v. When v comes from a one-dimensional problem, B is the following $n \times n$ matrix:

$$B = \begin{pmatrix} 0 & 1 & \dots & \dots & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & \dots & 0 & 1 & 0 \end{pmatrix}. \tag{22}$$

This matrix has to be modified appropriately when for instance $v = vec(F)$ where F is a matrix or a tensor. The regularized cost function with respect to the variable v_i will be

$$\frac{1}{2}\|R_i - u_i v^T\|_F^2 + \frac{\delta}{2}\|v - B\hat{v}_i\|_F^2, \quad \delta > 0$$

where the optimal regularized solution is given by

$$v^* = \frac{[R_i^T u_i + \delta B\hat{v}_i]_+}{\|u_i\|_2^2 + \delta}.$$

The constant $\delta \geq 0$ can be varied to control the trade-off between the approximation error $\frac{1}{2}\|R_i - u_i v^T\|_F^2$ and the smoothness of v_i at the fixed point. One can see that the regularized optimal solution v^* looks for a smooth solution in the neighborhood of $B\hat{v}_i$.

The three above regularizations can be applied independently to each of the columns of U and/or V. The trade-off factor β (or γ or δ) can be different for each column. A combination of the above regularizations can also be used to solve the multi-criterion problem

$$\frac{1}{2}\|R_i - u_i v^T\|_F^2 + \beta\|v\|_1 + \frac{\gamma}{2}\|v\|_2^2 + \frac{\delta}{2}\|v - B\hat{v}_i\|_F^2, \quad \beta, \gamma, \delta > 0$$

where the optimal update is given by

$$v^* = \frac{[R_i^T u_i - \beta \mathbf{1}_{n\times 1} + \delta B\hat{v}_i]_+}{\|u_i\|_2^2 + \gamma + \delta}.$$

The one-norm and two-norm regularizations can be found in [1] and [3]. A major difference is that in those papers the norm constraints are imposed on the rows rather than on the columns of V or U as done here. However, the effects are somehow similar. We test the smoothness regularization at the end of Section 8. We want to mention also the NMFLAB toolbox [8] that implements many variants and extentions proposed in the literature.

8 Experiments

Here we present several experiments to compare the different descent algorithms presented in this paper. For all the algorithms, the scaling scheme proposed in section 5.2 was applied.

8.1 Random matrices

We generated 100 random nonnegative matrices of different sizes. We used seven different algorithms to approximate each matrix:

- the multiplicative rule (**Mult**),
- alternative least squares using Matlab function *lsqnonneg* (**ALS**),

ϵ	Mult	ALS	FLine	CLine	FFO	CFO	RRI
(**m=30**, **n=20**, **r=2**)							
10^{-2}	0.02(96)	0.40	0.04	0.02	0.02	0.01	0.01
10^{-3}	0.08(74)	1.36	0.12	0.09	0.05	0.04	0.03
10^{-4}	0.17(71)	2.81	0.24	0.17	0.11	0.08	0.05
10^{-5}	0.36(64)	4.10	0.31	0.25	0.15	0.11	0.07
10^{-6}	0.31(76)	4.74	0.40	0.29	0.19	0.15	0.09
(**m=100**, **n=50**, **r=5**)							
10^{-2}	45∗(0)	3.48	0.10	0.09	0.09	0.04	0.02
10^{-3}	45∗(0)	24.30(96)	0.59	0.63	0.78	0.25	0.15
10^{-4}	45∗(0)	45∗(0)	2.74	2.18	3.34	0.86	0.45
10^{-5}	45∗(0)	45∗(0)	5.93	4.06	6.71	1.58	0.89
10^{-6}	45∗(0)	45∗(0)	7.23	4.75	8.98	1.93	1.30
(**m=100**, **n=50**, **r=10**)							
10^{-2}	45∗(0)	11.61	0.28	0.27	0.18	0.11	0.05
10^{-3}	45∗(0)	41.89(5)	1.90	2.11	1.50	0.74	0.35
10^{-4}	45∗(0)	45∗(0)	7.20	5.57	5.08	2.29	1.13
10^{-5}	45∗(0)	45∗(0)	12.90	9.69	10.30	4.01	1.71
10^{-6}	45∗(0)	45∗(0)	14.62(99)	11.68(99)	13.19	5.26	2.11
(**m=100**, **n=50**, **r=15**)							
10^{-2}	45∗(0)	25.98	0.66	0.59	0.40	0.20	0.09
10^{-3}	45∗(0)	45∗(0)	3.90	4.58	3.18	1.57	0.61
10^{-4}	45∗(0)	45∗(0)	16.55(98)	13.61(99)	9.74	6.12	1.87
10^{-5}	45∗(0)	45∗(0)	21.72(97)	17.31(92)	16.59(98)	7.08	2.39
10^{-6}	45∗(0)	45∗(0)	25.88(89)	19.76(98)	19.20(98)	10.34	3.66
(**m=100**, **n=100**, **r=20**)							
10^{-2}	45∗(0)	42.51(4)	1.16	0.80	0.89	0.55	0.17
10^{-3}	45∗(0)	45∗(0)	9.19	8.58	10.51	5.45	1.41
10^{-4}	45∗(0)	45∗(0)	28.59(86)	20.63(94)	29.89(69)	12.59	4.02
10^{-5}	45∗(0)	45∗(0)	32.89(42)	27.94(68)	34.59(34)	18.83(90)	6.59
10^{-6}	45∗(0)	45∗(0)	37.14(20)	30.75(60)	36.48(8)	22.80(87)	8.71
(**m=200**, **n=100**, **r=30**)							
10^{-2}	45∗(0)	45∗(0)	2.56	2.20	2.68	1.31	0.44
10^{-3}	45∗(0)	45∗(0)	22.60(99)	25.03(98)	29.67(90)	12.94	4.12
10^{-4}	45∗(0)	45∗(0)	36.49(2)	39.13(13)	45∗(0)	33.33(45)	14.03
10^{-5}	45∗(0)	45∗(0)	45∗(0)	39.84(2)	45∗(0)	37.60(6)	21.96(92)
10^{-6}	45∗(0)	45∗(0)	45∗(0)	45∗(0)	45∗(0)	45∗(0)	25.61(87)

Table 1: Comparison of average successful running time of algorithms over 100 random matrices. Time limit is 45 seconds. 0.02(96) means that a result is returned with the required precision ϵ within 45 seconds for 96 (of 100) matrices of which the average running time is 0.02 seconds. $45 * (0)$: failed in all 100 matrices.

- a full space search using line search and Armijo criterion (**FLine**),
- a coordinate search alternating on U and V, and using line search and Armijo criterion (**CLine**),
- a full space search using first-order approximation (**FFO**),
- a coordinate search alternating on U and V, and using first-order approximation (**CFO**)
- an iterative rank-one residue approximation (**RRI**).

For each matrix, the same starting point is used for every algorithm. We create a starting point by randomly generating two matrices U and V and then rescaling them to yield a first approximation of the original matrix A as proposed in Section 5.2:

$$U = UD\sqrt{\alpha}, \qquad V = VD^{-1}\sqrt{\alpha},$$

where

$$\alpha := \frac{\langle A, UV^T \rangle}{\langle UV^T, UV^T \rangle} \quad \text{and} \quad D_{ij} = \begin{cases} \sqrt{\frac{\|V_{:i}\|_2}{\|U_{:i}\|_2}} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}.$$

The algorithms are all stopped when the projected gradient norm is lower than ϵ times the gradient norm at the starting point or when it takes more than 45 seconds. The relative precisions ϵ are chosen equal to 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-6}. No limit was imposed on the number of iterations.

For alternative gradient algorithms **CLine** and **CFO**, we use different precisions ϵ_U and ϵ_V for each of the inner iteration for U and for V as suggested in [20] where ϵ_U and ϵ_V are initialized by 10^{-3}. And when the inner loop for U or V needs no iteration to reach the precision ϵ_U or ϵ_V, one more digit of precision will be added into ϵ_U or ϵ_V (i.e. $\epsilon_U = \epsilon_U/10$ or $\epsilon_V = \epsilon_V/10$).

Table 1 shows that for all sizes and ranks, Algorithm **RRI** is the fastest to reach the required precision. Even though it is widely used in practice, algorithm **Mult** fails to provide solutions to the NMF problem within the allocated time. A further investigation shows that the algorithm gets easily trapped in boundary points where some U_{ij} and/or V_{ij} is zero while $\nabla_{U_{ij}}$ and/or $\nabla_{V_{ij}}$ is negative, hence violating one of the KKT conditions (7). The multiplicative rules then fail to move and do not return to a local minimizer. A slightly modified version of this algorithm was given in [18], but it needs to wait to get sufficiently close to such points before attempting an escape, and is therefore also not efficient. The **ALS** algorithm can return a stationary point, but it takes too much time.

We selected the following five methods **FLine**, **CLine**, **FFO**, **CFO** and **RRI** for our detailed comparison. For each matrix A, we run these algorithms with 100 different starting points. Figure 1, 2, 3 and 4 show the results for different requested accuracies. One can see that, when the approximated errors are almost the same between the algorithms, **RRI** is the best overall in terms of running times.

Figure 1: Comparison of selected algorithms for $\epsilon = 10^{-7}$

8.2 Image data

The following experiments use the Cambridge ORL face database as input data. The database contains 400 images of 40 persons (10 images per person). The size of each image is 112×92 with 256 gray levels per pixel representing a front view of the face of a person. Since it was used in [17], this data has become the standard benchmark for NMF algorithms.

In the first experiment, we run six NMF algorithms described above on this

Figure 2: Comparison of selected algorithms for $\epsilon = 10^{-5}$

Figure 3: Comparison of selected algorithms for $\epsilon = 10^{-4}$

Figure 4: Comparison of selected algorithms for $\epsilon = 10^{-3}$

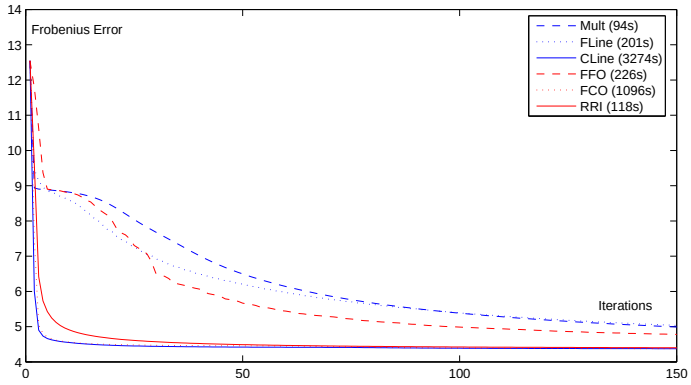

Figure 5: NMF: Error vs. Iterations

data for the reduced rank of 49. Each column of the original matrix A consists of an image transformed into a single column. Figure 5 shows for the six algorithms the evolution of the error versus the number of iterations. Because the minimization process is different in each algorithm, we will say that one iteration corresponds to all elements of both U and V being updated. Figure 6 shows the evolution of the error versus time. Since the work of one iteration varies from one algorithm to another, it is crucial to plot the error versus time to get a fair comparison between the different algorithms. In the two figures, we can see that the RRI algorithm behaves very well on this dataset. And since its computation load of each iteration is small and constant (without inner loop), this algorithm converges faster than the others.

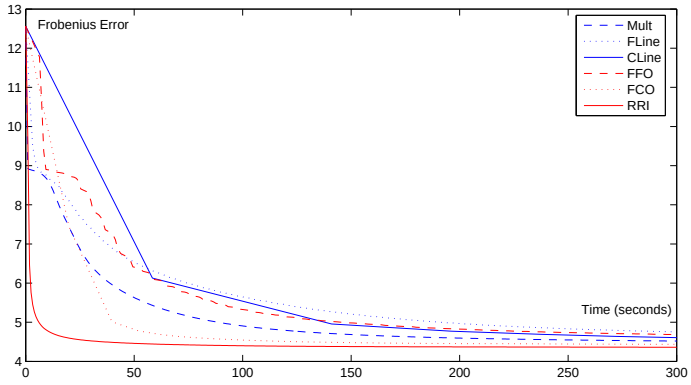

Figure 6: NMF: Error vs. Time

In the second experiment, we construct a third-order nonnegative tensor approximation. We first build a tensor by stacking all 400 images to have a $112 \times 92 \times 400$ nonnegative tensor. Using the proposed algorithm, a $rank-142$ nonnegative tensor is calculated to approximate this tensor. Figure 7

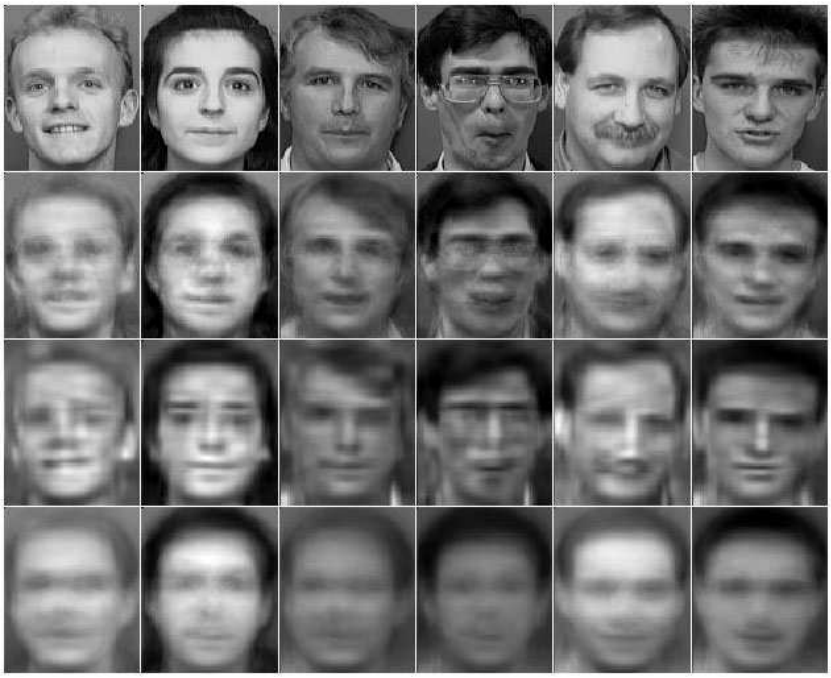

Figure 7: Tensor Factorization vs. Matrix Factorization on facial data. Six randomly chosen images from 400 of ORL dataset. From top to bottom: original images, their $rank-8$ truncated SVD approximation, their $rank-142$ nonnegative tensor approximation and their $rank-8$ nonnegative matrix approximation.

shows the result for six images chosen randomly from the 400 images. Their approximations given by the rank-142 nonnegative tensor are much better than that given by the rank-8 nonnegative matrix, even though they require similar storage space: $8 * (112 * 92 + 400) = 85632$ and $142 * (112 + 92 + 400) = 85768$. The rank-8 truncated SVD approximation (i.e. $[A_8]_+$) is also included for reference.

In the third experiment, we apply the variants of RRI algorithm mentioned in Section 6.3 to the face databases. The following settings are compared:

- **Original**: original faces from the databases.
- **49NMF**: standard factorization (nonnegative vectors), $r = 49$.
- **100Binary**: columns of U are limited to the scaled binary vectors, $r = 100$.
- **49Sparse10**: columns of U are sparse. Not more than 10% of the elements of each column of A are positive. $r = 49$.
- **49Sparse20**: columns of U are sparse. Not more than 20% of the elements of each column of A are positive. $r = 49$.
- **49HSparse60**: columns of U are sparse. The Hoyer sparsity of each column of U are 0.6. $r = 49$.
- **49HSparse70**: columns of U are sparse. The Hoyer sparsity of each column of U are 0.7. $r = 49$.
- **49HBSparse60**: columns of U are sparse. The Hoyer sparsity of each column of U are 0.6. Columns of V are scaled binary. $r = 49$.
- **49HBSparse70**: columns of U are sparse. The Hoyer sparsity of each column of U are 0.7. Columns of V are scaled binary. $r = 49$.

For each setting, we use RRI algorithm to compute the corresponding factorization. Some randomly selected faces are reconstructed by these settings as showed in Figure 8. For each setting, RRI algorithm produces a different set of bases to approximate the original faces. When the columns of V are constrained to scaled binary vectors (**100Binary**), the factorization can be rewritten as $UV^T = \hat{U}B^T$, where B is a binary matrix. This implies that each image is reconstructed by just the presence or absence of 100 bases shown in Figure 9.

Figure 10 and 11 show nonnegative bases obtained by imposing some sparsity on the columns of V. The sparsity can be easily controlled by the percentages of positive elements or by the Hoyer sparsity measure.

Figure 8: Nonnegative matrix factorization with several sparse settings

Figure 9: Bases from **100Binary** setting

(a) (b)

Figure 10: Sparse bases **49Sparse20** and **49Sparse10**. Maximal percentage of positive elements is 20% (a) and 10% (b)

.

(a) (b)

Figure 11: Hoyer sparse bases **49HSparse60** and **49HSparse70**. Sparsity of bases is 0.6 (a) and 0.7 (b)

.

Figure 12 combines the sparsity of the bases (columns of U) and the binary representation of V. The sparsity is measured by the Hoyer measure as in Figure 11. Only with the absence or presence of these 49 *features*, faces are approximated as showed in the last two rows of Figure 8.

(a) (b)

Figure 12: Hoyer sparse bases **49HBSparse60** and **49HBSparse70**. Sparsity of bases is 0.6 (a) and 0.7 (b). V is binary matrix.

.

The above examples show how to use the variants of the RRI algorithm to control the sparsity of the bases. One can see that the sparser the bases are, the less storage is needed to store the approximation. Moreover, this provides a part-based decomposition using *local* features of the faces.

8.3 Smooth approximation

Figure 13: Smooth functions

We carry out this experiment to test the new smoothness constraint introduced in the previous section:

$$\frac{1}{2}\|R_i - u_i v^T\|_F^2 + \frac{\delta}{2}\|v - B\hat{v}_i\|_F^2, \quad \delta > 0$$

where B is defined in (22).

We generate the data using four smooth nonnegative functions f_1, f_2, f_3 et f_4, described in Figure 13, where each function is represented as a nonnegative vector of size 200.

We then generate a matrix A containing 100 mixture of these functions as follows

$$A = max(FE^T + N, 0)$$

where $F = [f_1\ f_2\ f_3\ f_4]$, E is a random nonnegative matrix and N is random noise with $\|N\|_F = 0.2\|FE^T\|_F$. Four randomly selected columns of A are plotted in Figure 14.

Figure 14: Randomly selected generated data

Figure 15: Original functions vs. reconstructed functions

We run the regularized RRI algorithm to force the smoothness of columns of U. We apply, for each run, the same value of δ for all the columns of U: $\delta = 0, 10, 100$. The results obtained through these runs are presented in Figure 15. We see that, without regularization, i.e. $\delta = 0$, the noise is present in the approximation, which produces nonsmooth solutions. When increasing the regularizing terms, i.e. $\delta = 10, 100$, the reconstructed functions become smoother and the shape of the original functions are well recovered.

9 Conclusion

This paper focuses on the descend methods for Nonnegative Matrix Factorization, which are characterized by nonincreasing updates at each iteration.

We present also the Rank-one Residue Iteration algorithm for computing an approximate Nonnegative Matrix Factorization. It uses recursively nonnegative rank one approximations of a residual matrix that is not necessarily nonnegative. This algorithm requires no parameter tuning, has nice properties and typically converges quite fast. It also has many potential extensions. During the revision of this report, we were informed that essentially the same algorithm was published in an independent contribution [7] and also mentioned later in an independent personal communication [10].

Acknowledgments

This paper presents research results of the Concerted Research Action(ARC) "Large Graphs and Networks" of the French Community of Belgium and the Belgian Network DYSCO (Dynamical Systems, Control, and Optimization), funded by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office. The scientific responsibility rests with its authors. Ngoc-Diep Ho is a FRIA fellow.

References

[1] R. Albright, J. Cox, D. Duling, A.N. Langville, and C.D. Meyer. Algorithms, initializations, and convergence for the nonnegative matrix factorization. *Preprint*, 2006.

[2] B.W. Bader and T.G. Kolda. Efficient MATLAB computations with sparse and factored tensors. Technical Report SAND2006-7592, Sandia National Laboratories, Albuquerque, NM and Livermore, CA, Dec. 2006.

[3] M.W. Berry, M. Browne, A.N. Langville, V.P. Pauca, and R.J. Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational Statistics and Data Analysis*, 52(1):155–173, 2007.

[4] D.P. Bertsekas. *Nonlinear programming*. Athena Scientific Belmont, Mass, 1999.

[5] R. Bro and S. De Jong. A fast non-negativity constrained least squares algorithm. *Journal of Chemometrics*, 11(5):393–401, 1997.

[6] M. Catral, L. Han, M. Neumann, and R.J. Plemmons. On reduced rank nonnegative matrix factorization for symmetric nonnegative matrices. *Linear Algebra and Its Applications*, 393:107–126, 2004.

[7] A. Cichocki, R. Zdunek, and S. Amari. Hierarchical ALS Algorithms for Nonnegative Matrix and 3D Tensor Factorization. In *Proceedings of Independent Component Analysis, ICA 2007, London, UK, September 9-12, 2007, Lecture Notes in Computer Science, Vol. LNCS 4666, Springer*, 169–176, 2007.

[8] A. Cichocki, R. Zdunek. NMFLAB for Signal Processing, available at http://www.bsp.brain.riken.jp/ICALAB/nmflab.html.

[9] C. Ding, T. Li, and M.I. Jordan. Convex and Semi-Nonnegative Matrix Factorizations. Technical report, LBNL Tech Report 60428, 2006.

[10] N. Gillis and Franois Glineur. Nonnegative Matrix Factorization and Underapproximation. Preprint, 2007.

[11] G. Golub and C.F. Van Loan. *Matrix computations.3rd ed.* Baltimore, The Johns Hopkins Univ. Press. xxvii, 694 p. , 1996.

[12] N.J. Higham, M.J.C. Gover and S. Barnett. Matrix Nearness Problems and Applications. *Applications of Matrix Theory*, 1–27, 1989.

[13] N.-D. Ho, P. Van Dooren, and V.D. Blondel. Descent algorithms for Nonnegative Matrix Factorization. Technical Report 2007-57, Cesame. University catholique de Louvain. Belgium. 2007.

[14] P.O. Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469, 2004.

[15] T.G. Kolda and D.P. O'Leary. A semidiscrete matrix decomposition for latent semantic indexing information retrieval. *ACM Transactions on Information Systems (TOIS)*, 16(4):322–346, 1998.

[16] C.L. Lawson and R.J. Hanson. *Solving least squares problems*. Prentice-Hall Englewood Cliffs, NJ, 1974.

[17] D.D. Lee and H.S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.

[18] C.-J. Lin. On the convergence of multiplicative update algorithms for non-negative matrix factorization. *IEEE Transactions on Neural Networks*, 2007. To appear.

[19] C.-J. Lin. Projected gradient methods for non-negative matrix factorization. *Neural Computation*, 2007. To appear.

[20] C.-J.-Lin. Projected gradient methods for non-negative matrix factorization. *Technical Report Information and Support Services Technical Report ISSTECH-95-013, Department of Computer Science, National Taiwan University, 2005*

[21] M. Merritt and Y. Zhang. Interior-Point Gradient Method for Large-Scale Totally Nonnegative Least Squares Problems. *Journal of Optimization Theory and Applications*, 126(1):191–202, 2005.

[22] P. Paatero and U. Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(1):111–126, 1994.

[23] P. Paatero. A weighted non-negative least squares algorithm for three-way 'parafac' factor analysis. *Chemometrics and Intelligent Laboratory Systems*, 38(2):223–242, 1997.

[24] M. Biggs, A. Ghodsi and S. Vavasis. Nonnegative matrix factorization via rank-one downdate. *University of Waterloo, Preprint*, 2007.