Using underapproximations for sparse nonnegative matrix factorization ☆

Nicolas Gillis [a,b,*,1], François Glineur [a,b]

[a] Université catholique de Louvain, Department of Mathematical Engineering, B-1348 Louvain-la-Neuve, Belgium
[b] Université catholique de Louvain, CORE, B-1348 Louvain-la-Neuve, Belgium

ARTICLE INFO

ABSTRACT

Nonnegative matrix factorization consists in (approximately) factorizing a nonnegative data matrix by the product of two low-rank nonnegative matrices. It has been successfully applied as a data analysis technique in numerous domains, e.g., text mining, image processing, microarray data analysis, collaborative filtering, etc.

We introduce a novel approach to solve NMF problems, based on the use of an underapproximation technique, and show its effectiveness to obtain sparse solutions. This approach, based on Lagrangian relaxation, allows the resolution of NMF problems in a recursive fashion. We also prove that the underapproximation problem is NP-hard for any fixed factorization rank, using a reduction of the maximum edge biclique problem in bipartite graphs.

We test two variants of our underapproximation approach on several standard image datasets and show that they provide sparse part-based representations with low reconstruction error. Our results are comparable and sometimes superior to those obtained by two standard sparse nonnegative matrix factorization techniques.

1. Introduction

Nonnegative matrix factorization (NMF) is a recent data analysis technique with applications in image processing, text mining, spectral unmixing, air emission control, computational biology, clustering, etc. (see [1–4] and references therein). NMF can be described as follows: given a nonnegative input matrix $M \in \mathbb{R}_+^{m \times n}$ and an integer $1 \le r < \min(m, n)$, find two nonnegative matrices $V \in \mathbb{R}_+^{m \times r}$ and $W \in \mathbb{R}_+^{r \times n}$ whose product approximates the input matrix as closely as possible:

$$M \approx VW, \qquad (1)$$

so that VW is a low-rank approximation of M. Matrix factorization can be interpreted as a linear factor model: assuming that each column of the input matrix M represents an element of a dataset, decomposition (1) can be written as[2]

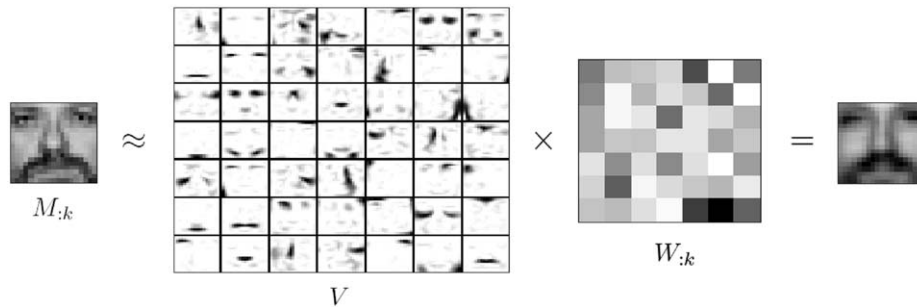$$M_{\cdot j} \approx \sum_k V_{\cdot k} W_{kj}, \qquad \forall j,$$

i.e., each input column $M_{\cdot j}$ is a linear combination of a set of r basis elements $V_{\cdot k}$ with corresponding weights W_{kj}.

In contrast with standard linear factor model techniques such as principal component analysis, NMF considers nonnegativity of the input columns to be an important feature and consequently requires the basis elements to be also nonnegative, so that they can be interpreted in the same way (e.g., these columns can correspond to images described by nonnegative pixel intensities or to texts represented by vectors of nonnegative word counts). Furthermore, NMF imposes nonnegativity of the weights, leading to an essentially additive reconstruction of the input columns by the basis elements. This representation is then *part-based*: basis elements $V_{\cdot k}$ will represent common parts of the columns of $M_{\cdot j}$. For example, if each column of M represents a face using pixel intensities, the basis elements generated by NMF can be facial features, such as eyes, noses and lips, as shown in Fig. 1.

This low-rank approximation technique with nonnegativity constraints was introduced in 1994 by Paatero and Tapper [5] and started to be extensively studied after the publication of an article by Lee and Seung [6] in 1999. Since an exact representation of the input matrix cannot be obtained in general, the quality of the approximation is measured by some criterion, typically the sum of the squares of the errors on the entries, which leads to the following minimization problem[3]:

$$\min_{V \in \mathbb{R}^{m \times r}, W \in \mathbb{R}^{r \times n}} \|M - VW\|_F^2 \quad \text{such that } V \ge 0 \text{ and } W \ge 0. \qquad \text{(NMF)}$$

---

[2] In this text, A_{ij} stands for the (i, j)-entry of a matrix A, $A_{\cdot j}$ for its jth column and $A_{i\cdot}$ for its ith row.

---

[3] $\|A\|_F = \left(\sum_{ij} A_{ij}^2 \right)^{1/2}$ denotes the Frobenius norm of matrix A.

Fig. 1. NMF applied to the CBCL face database #1, MIT Center For Biological and Computation Learning (available at http://cbcl.mit.edu/cbcl/software-datasets/FaceData2.html). It consists of the approximation of 2429 gray-level images of faces represented with 19×19 pixels (columns of M) using $r = 49$ basis elements (columns of V).

An important feature of NMF is that its nonnegativity constraints typically induce *sparse factors*, i.e., factors with relatively many zero entries. Intuitively, decomposition into parts requires the basis elements to be sparse, cf. Fig. 1. More formally, the reason for this behavior is that stationary points (V, W) of NMF will be typically located at the boundary of the feasible domain $\mathbb{R}_+^{m \times r} \times \mathbb{R}_+^{r \times n}$, hence will feature zero components. This can be explained with the first-order optimality conditions: because the set of stationary points of a problem of the type

$$\min_{x \in \mathbb{R}^n} f(x) \quad \text{such that } x \geq 0$$

is given by the following expression (where $\nabla f(x)$ is the gradient of f):

$$S_f = \{x \in \mathbb{R}^n \mid x \geq 0, \nabla f(x) \geq 0 \text{ and } \mathbf{x_i}[\nabla \mathbf{f(x)}]_\mathbf{i} = \mathbf{0}, \ \forall i\},$$

some components of the solution can be expected to be equal to zero.

Sparsity of the factors is an important consideration in practice: in addition to reducing memory requirements to store the basis elements and their weights, sparsity improves interpretation of the factors, especially when dealing with classification/clustering problems, e.g., in text mining [7] and computational biology [8,9]. By contrast, unconstrained low-rank approximations such as principal component analysis (PCA) do not naturally generate sparse factors (for that reason, low-rank approximation techniques with additional sparsity constraints have been recently introduced; this is referred to as sparse principal component analysis, sparse PCA or SPCA, see, e.g., [10] and references therein).

Although solutions of NMF typically display some level of sparsity, some applications require even sparser solutions, leading to variants of NMF called sparse nonnegative matrix factorization. They are in general developed in two different ways: some authors define a priori a desired sparsity level and adapt the main iteration of their method in order to guarantee that the factors satisfy that level of sparsity throughout the application of the algorithm, see, e.g., [11,12]. Alternatively, a penalty term can be added to the objective function to prevent the algorithm from considering dense solutions, see [13]. In particular, it is well-known that l_1- norm penalty terms induce sparser solutions (see, e.g., [14,9,15]). More details about these techniques are given at the beginning of Section 4.

Unfortunately the advantages of NMF (part-based representation and sparsity) over PCA come at a certain price. First, because of the additional nonnegativity constraints, the approximation error of the input data for a given factorization rank r will always be higher for NMF than in the unconstrained case. Second, optimization problem (NMF) is more difficult to solve than its unconstrained counterpart: while PCA problems can be solved in polynomial time (e.g., using a singular value decomposition technique [16]), NMF problems belong to the class of NP-hard

problems, as recently showed by Vavasis [17]. However, it should also be pointed out that these drawbacks (higher error, NP-hardness) are also present for competing techniques emphasizing sparsity, such as SPCA.

Because of its NP-hardness, practical algorithms cannot be expected to find provably optimal global solutions for (NMF) in a reasonable amount of time and aim instead at finding locally optimal solutions. Most methods start from some initial guess factors (V, W) and improve them iteratively using nonlinear optimization schemes such as projected gradient methods [18], Newton-like methods [19,20], (block-)coordinate descent (also called alternating nonnegative least squares—NNLS) [21,22,14], multiplicative updates [23], etc. (see also [1,2,4,24] and references therein).

In this paper, we introduce a novel approach to solve NMF problem based on the use of an underapproximation technique and show its effectiveness to obtain sparse solutions. Section 2 introduces our underapproximation problem, motivated by a recursive technique to solve NMF, studies the sparsity of its solutions and proves that it is NP-hard for any fixed factorization rank. Nevertheless, Section 3 describes an algorithm to solve it approximately using a technique based on Lagrangian relaxation. Finally, in the last section, we test this approach on several standard image datasets, and show both qualitatively and quantitatively that it provides part-based and sparse representations that are comparable and sometimes superior to those obtained with standard sparse nonnegative matrix factorization techniques.

2. Nonnegative matrix underapproximation

2.1. A recursive approach

Finding a rank-one nonnegative matrix factorization, i.e., solving (NMF) with $r = 1$ is notably easier than for higher factorization ranks: while the general problem is NP-hard, computing a globally optimal rank-one approximation can be done in polynomial time. More specifically, the first rank-one factor of the singular value decomposition (SVD) of the input matrix is an optimal solution: indeed, the Perron–Frobenius theorem implies that the dominant left and right singular vectors of a nonnegative matrix are nonnegative, while the Eckart–Young theorem states that the outer product of these dominant singular vectors is the best possible rank-one approximation in the Frobenius norm.

In principle, we might try exploit this result to find factorizations of higher ranks by applying it recursively: after identification of an optimal rank-one NMF solution (v, w), one could subtract the vw factor from M and apply the same technique to $M - vw$ to recover the next rank-one factor. Unfortunately, this idea cannot work: the difference between M and its rank-one approximation may contain negative values (typically roughly

half of them), so that the next SVD factor will no longer provide a nonnegative solution. Moreover, there is no hope of replacing SVD by another efficient technique for this step since [26] shows that it is NP-hard to find the optimal nonnegative rank-one approximation to a matrix which is not nonnegative.

If we wish to keep the principle of a recursive algorithm finding one rank-one factor at a time, we have to add a constraint ensuring that the vw factor, when subtracted from M, gives a nonnegative remainder, i.e., we need to have $vw \leq M$. Therefore we introduce a similar *upper bound constraint* $VW \leq M$ to the general (NMF) problem and obtain a new problem we call *nonnegative matrix underapproximation* (NMU): given $M \in \mathbb{R}_+^{m \times n}$ and $1 \leq r < \min(m, n)$, the NMU optimization problem is defined as

$$\min_{V \in \mathbb{R}^{m \times r}, W \in \mathbb{R}^{r \times n}} \|M - VW\|_F^2 \quad \text{such that } V \geq 0, \quad W \geq 0 \text{ and } VW \leq M.$$

(NMU)

Assuming we are able to solve it for $r = 1$, an underapproximation of any rank can then be built by following the recursive procedure outlined above. More precisely, if $(V_{:1}, W_{1:})$ is a rank-one underapproximation for M, i.e., $V_{:1}W_{1:} \approx M(= R_1)$ and $V_{:1}W_{1:} \leq M$, we have that $R_2 = M - V_{:1}W_{1:}$ is nonnegative. R_2 can then be underapproximated $V_{:2}W_{2:} \leq R_2$, leading to $R_3 = R_2 - V_{:2}W_{2:}$, and so on. After r steps, we get an underapproximation of rank r

$$M \geq V_{:1}W_{1:} + V_{:2}W_{2:} + \cdots + V_{:r}W_{r:}$$
$$= [V_{:1} V_{:2} \cdots V_{:r}][W_{1:}; W_{2:}; \cdots; W_{r:}]$$
$$= VW.$$

Besides enabling this recursive procedure, we notice that NMU leads to a more *localized* part-based decomposition, in the sense that different basis elements tend to describe disjoint parts of the input data (i.e., involving different nonzero entries). This is a consequence of the underapproximation constraints which impose the extracted parts (the basis elements $V_{:k}$) to really be common features of the columns of M since

$$M_{:j} \gtrapprox \sum_k V_{:k} W_{kj}, \quad \forall j.$$

Basis elements can only be combined to approximate a column of M if each of them represents a part of this column, i.e., none of the parts selected with a positive weight can involve a nonzero entry corresponding to a zero entry in the input column $M_{:j}$. The following example demonstrates this behavior.

Example 1 (*Swimmer database*). The swimmer image dataset consists of 256 binary images of a body with four limbs which can be each in four different positions. NMF is expected to find a part-based decomposition of these images, i.e., isolate different constitutive parts of the images (the body and the limbs) in each of its basis elements.

Fig. 2 displays a sample of such images along with the basis elements obtained with NMF and NMU. While NMF elements are rather sparse, they are mixtures of several limbs. By contrast, NMU returns an even sparser solution and is able to extract a single body part for each of its elements.

2.2. Sparsity

The fact that NMU decompositions naturally generate sparser solutions than NMF can be explained as follows: since the zero entries of M can only be underapproximated by zeros, we have

$$M_{ij} = 0 \Rightarrow (VW)_{ij} = 0 \Rightarrow V_{ik} = 0 \quad \text{or} \quad W_{kj} = 0, \quad \forall k$$

which shows that when the input matrix is sparse, many components of the NMU factors will have to be equal to zero. This observation can be made more formal: defining the sparsity $s(M)$ of a m by n matrix M as the proportion of its zero entries, i.e.,
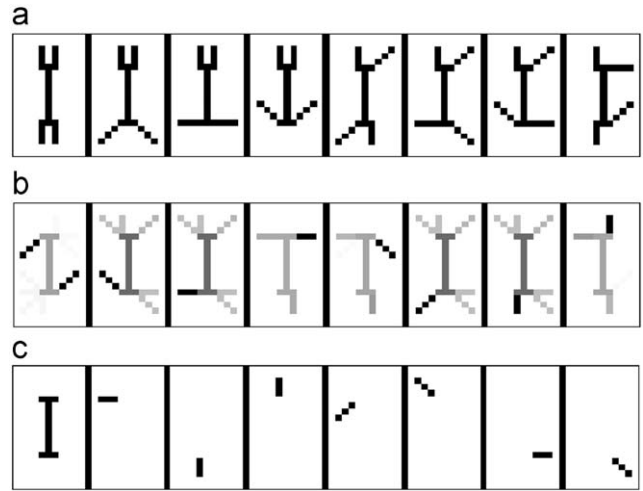

Fig. 2. Basis elements generated for the swimmer image dataset with $r = 8$: (a) sample images from the dataset, (b) NMF and (c) NMU; see Section 4 for the algorithms used to compute the factorizations.

$$s(M) = \frac{\#zeros(M)}{mn} \in [0, 1],$$

we have the following theorem relating sparsity of M and its NMU factors.

Theorem 1. *For any nonnegative rank-one underapproximation* $(v, w) \in \mathbb{R}_+^m \times \mathbb{R}_+^n$ *of* $M \in \mathbb{R}_+^{m \times n}$ *we have*

$$s(v) + s(w) \geq s(M).$$

Proof. For a rank-one matrix vw, the number of nonzeros is exactly equal to the product of the number of nonzeros in vectors v and w. Therefore we have that $(1 - s(vw)) = (1 - s(v))(1 - s(w))$ which implies $s(vw) = s(v) + s(w) - s(v)s(w) \leq s(v) + s(w)$. Since underapproximation vw satisfies $0 \leq vw \leq M$, it must have more zeros than M and we have

$$s(M) \leq s(sw) \leq s(v) + s(w),$$

proving our claim. ☐

Recall the recursive definition of the residuals $R_{k+1} = R_k - V_{:k}W_{k:}$ and $R_1 = M$. The following corollary relates their sparsity and the sparsity of the whole rank-r approximation with that of the NMU factors.

Corollary 1. *For any nonnegative underapproximation* $(V, W) \in \mathbb{R}_+^{m \times r} \times \mathbb{R}_+^{r \times n}$ *of* $M \in \mathbb{R}_+^{m \times n}$ *we have for each factor*

$$s(V_{:k}) + s(W_{k:}) \geq s(R_k) \geq s(M), \quad 1 \leq k \leq r,$$

and $s(V) + s(W) \geq s(M)$.

Proof. We have $0 \leq V_{:k}W_{k:} \leq R_k \leq M$, which implies by the previous theorem the first set of inequalities. Observing that $s(V) = (1/r)\sum_k s(V_{:k})$ and $s(W) = (1/r)\sum_k s(W_{k:})$ is sufficient to prove the second inequality. ☐

Sparsity of the residuals R_k is monotonically nondecreasing at each step, since $M = R_1 \geq R_2 \geq \cdots \geq 0$. Moreover, the following theorem can guarantee an increase in sparsity at each step.

Theorem 2. *For any locally optimal nonnegative rank-one underapproximation* $(v, w) \in \mathbb{R}_+^m \times \mathbb{R}_+^n$ *of* $M \in \mathbb{R}_+^{m \times n}$, *define sets* I *and* J *(supports of vectors* v *and* w*) by*

$$I = \{i \mid v_i > 0\}, \quad J = \{j \mid w_j > 0\},$$

and define matrix $R(I, J)$ *to be the submatrix of residual* $R = M - vw$ *whose row and column indices belong, respectively, to* I *and* J

(*corresponding to the submatrix of M that is not approximated by zeros*). *Then there is at least one zero in each row and each column of submatrix* $R(I,J)$.

Proof. Simply observe that if $R(i,J) > 0$ (resp. $R(I,j) > 0$) for some $i \in I$ (resp. $j \in J$), v_i (resp. w_j) can be increased to obtain a strictly better solution, which contradicts the local optimality assumption. □

This ability of NMU to generate sparse part-based decomposition will be experimentally confirmed in Section 4.

2.3. Related work

The problem of rank-one underapproximation has been first introduced by Levin in [27] in the case of *positive stochastic* matrices. He introduced a specific objective function different from the Frobenius norm and used a logarithmic change of variables in order to design an iterative method based on the corresponding optimality conditions.

In [28], the rank-one underapproximation problem is cast as a convex problem (hence efficiently solvable) using again different objective functions. Solutions are then used to initialize standard NMF algorithms in order to accelerate their convergence and, in general, find better final solutions as compared to those obtained with random initializations. Similar behavior was observed for other judicious initializations in [29–31].

More recently, Dong et al. [32] studied the same problem with the additional constraint that the rank of the residual must be strictly smaller than the rank of the factorized matrix. Using the Wedderburn rank reduction formula, they proposed a numerical procedure which is able to compute the maximum rank splitting of a nonnegative matrix. However, the underlying optimization problem is NP-hard [17] and their algorithm is not guaranteed to find a solution in all cases.

Biggs et al. [33] also introduced a recursive procedure to solve NMF problems: their idea is to locate and then approximate nearly rank-one submatrices of M. However, the problem of locating maximum rank-one submatrices is also shown to be NP-hard, and their algorithm is not globally optimal.

2.4. Complexity

We now prove that (NMU) is NP-hard, even in the rank-one case (unlike (NMF), which is polynomially solvable in the rank-one case). In order to do this, the rank-one version of the problem is proved to be equivalent to the biclique problem, which is NP-hard. The result is then generalized to (NMU) with arbitrary factorization rank r using a simple construction.

A *bipartite graph* G_b is a graph whose vertices can be divided into two disjoint sets such that there is no edge between two vertices in the same set. A *biclique* K_b is a complete bipartite graph, i.e., a bipartite graph where all the vertices from different sets are connected by an edge. Finally, the so-called maximum edge biclique problem (the biclique problem for short) in a bipartite graph

$$G_b = (V = V_1 \cup V_2, E \subseteq (V_1 \times V_2))$$

is the problem of finding a biclique $K_b = (V', E')$ in G_b (i.e., $V' = V_1' \cup V_2' \subseteq V$ and $E' = (V_1' \times V_2') \subseteq E$) with a maximum number of edges $|E'| = |V_1'| \cdot |V_2'|$.

Letting $M \in \{0,1\}^{m \times n}$ be the adjacency matrix of G_b with $V_1 = \{s_1, \ldots, s_m\}$ and $V_2 = \{t_1, \ldots, t_n\}$, i.e.,

$$M_{ij} = 1 \iff (s_i, t_j) \in E,$$

and introducing indicator binary variables v_i (resp. w_j) to denote whether s_i (resp. t_j) belongs to the biclique K_b, the maximum edge biclique problem (MBP) in a bipartite graph can be formulated as follows:

$$\min_{v,w} \quad \sum_{i,j} (M_{ij} - v_i w_j)^2$$
$$v_i w_j \leq M_{ij}, \quad \forall i,j,$$
$$v \in \{0,1\}^m, \quad w \in \{0,1\}^n. \qquad \text{(MBP)}$$

One can check that this objective is equivalent to $\max_{v,w} \sum_{i,j} v_i w_j$. In fact, $M_{ij} - v_i w_j = (M_{ij} - v_i w_j)^2$ since M, v and w are binary and $M_{ij} \geq v_i w_j$.

The corresponding decision problem "*Given K, does G_b contain a biclique with at least K edges?*" has been shown to be NP-complete [34]. Therefore, the corresponding optimization problem (MBP) is at least NP-hard.

For $r = 1$, (NMU) can be written as

$$\min_{v \in \mathbb{R}^m, w \in \mathbb{R}^n} \quad \sum_{i,j} (M_{ij} - v_i w_j)^2$$
$$v_i w_j \leq M_{ij}, \quad \forall i,j,$$
$$v \geq 0, \quad w \geq 0, \qquad \text{(NMU1)}$$

which is very close to (MBP): the difference is that vectors v and w are required to be *binary* for (MBP) and *nonnegative* for (NMU1). The next lemma proves that the two problems are actually equivalent.

Lemma 1. *For* $M \in \{0,1\}^{m \times n}$, *every optimal solution* (v,w) *of* (NMU1) *is such that* vw *is binary, i.e.,* $vw \in \{0,1\}^{m \times n}$, *and can then be trivially transformed into a binary optimal solution* $(v', w') \in \{0,1\}^m \times \{0,1\}^n$ *of* (MBP).

Proof. For $M = 0$, this is trivial. Otherwise, suppose (v,w) is an optimal solution of (NMU1). Let define (v', w') as

$$v_i' = \begin{cases} 1 & \text{if } v_i \neq 0 \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad w_j' = \begin{cases} 1 & \text{if } w_j \neq 0, \\ 0 & \text{otherwise}, \end{cases}$$

and analyze the different possibilities: as $v_i w_j \leq M_{ij}$, we have either

- $M_{ij} = 1$ and $0 < v_i w_j \leq 1 \Rightarrow v_i' w_j' = 1$;
- $M_{ij} = 1$ and $v_i w_j = 0 \Rightarrow v_i' w_j' = 0$;
- $M_{ij} = 0 \Rightarrow v_i w_j = 0 \Rightarrow v_i' w_j' = 0$.

Therefore, $v_i w_j \leq v_i' w_j' \leq M_{ij}$ which implies

$$\|M - v'w'\|_F \leq \|M - vw\|_F.$$

By optimality of (v,w), we must have $vw = v'w' \in \{0,1\}^{m \times n}$. Therefore, $(v', w') = (v/\max(v), w/\max(w))$ is an optimal binary solution of (NMU1) which is then also an optimal solution of (MBP) (note that we must have $\max(v) = \max(w)^{-1}$). □

Corollary 2. (NMU1) *is NP-hard.*

We now generalize Corollary 2 to the more general case of (NMU) with $r > 1$.

Theorem 3. (NMU) *is NP-hard.*

Proof. Let $M \in \{0,1\}^{m \times n}$ be the adjacency matrix of a bipartite graph G_b. We define the matrix A as

$$A = \text{diag}(M, r) = \begin{pmatrix} M & 0 & \ldots & 0 \\ 0 & M & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \ldots & & M \end{pmatrix},$$

which is the adjacency matrix of another bipartite graph G_b^r which is the graph G_b repeated r times. Let (V, W) be an

optimal solution of (NMU). Since $VW = \sum_{k=1}^{r} V_{:k} W_{k:}$, we have $VW \leq A \Rightarrow V_{:k} W_{k:} \leq A$. Therefore $(V_{:k}, W_{k:})$ is a feasible solution of (NMU1) for the matrix A, i.e., for the graph G_b^r. Hence, each $(V_{:k}, W_{k:})$ corresponds to a biclique $B_G^k = (V_1^k \cup V_2^k, E^k)$ of G_b^r with

$$V_{ik} \neq 0 \iff s_i \in V_1^k \quad \text{and} \quad W_{kj} \neq 0 \iff t_j \in V_2^k.$$

By optimality of (V, W) and since there are at least r independent maximum biclique in G_b^r, each $(V_{:k}, W_{k:})$ must coincide with a maximum biclique of G_b^r which corresponds to a maximum biclique of G_b. This is due to the fact that, because G_b^r is the graph G_b repeated r times, a biclique clearly cannot span two disjoint subgraphs of G_b^r. Therefore, (NMU) is NP-hard since any instance of (MBP) can be polynomially reduced to an instance of (NMU). □

3. An algorithm for NMU based on Lagrangian relaxation

Since (NMU), like (NMF), is a NP-hard problem, we cannot expect to solve it up to guaranteed global optimality in a reasonable (e.g., polynomial) computational time (unless $P = NP$). In this section, we propose a nonlinear optimization scheme based on Lagrangian relaxation in order to compute approximate solutions of (NMU).

Drop the $m \times n$ underapproximation constraints $VW \leq M$ of (NMU) and add them into the objective function with the corresponding Lagrange multipliers (dual variables, forming a matrix) $\Lambda \in \mathbb{R}_+^{m \times n}$, to obtain the Lagrangian function $L(V, W, \Lambda)$

$$L(V, W, \Lambda) = \frac{1}{2} \|M - VW\|_F^2 + \sum_{i=1}^{m} \sum_{j=1}^{n} \Lambda_{ij} (VW - M)_{ij},$$

where a factor of $\frac{1}{2}$ was introduced to make the presentation nicer. The Lagrangian relaxation subproblem LR_Λ consists in minimizing L for a fixed value of the Λ multipliers, leading to the corresponding Lagrangian dual function $f(\Lambda)$

$$f(\Lambda) = \min_{V, W \geq 0} L(V, W, \Lambda), \tag{LR_Λ}$$

where $f(\Lambda)$ is well-defined because the minimum of $L(V, W, \Lambda)$ is always attained, due to the fact that f is bounded below and the search space can be restricted to a compact set. Indeed, considering each rank-one factor individually $(V_{:k}, W_{k:})$ and imposing w.l.o.g. $\|V_{:k}\|_F^2 = \|W_{k:}\|_F^2 = \|V_{:k}\|_F \|W_{k:}\|_F = \|V_{:k} W_{k:}\|_F$, we have

$$\|V_{:k}\|_F^2 = \|W_{k:}\|_F^2$$
$$\leq \|VW\|_F \leq \|M - \Lambda\|_F + \|M - \Lambda - VW\|_F \leq 2\|M - \Lambda\|_F, \quad \forall k,$$

where we have used the trivial solution $(V, W) = (0, 0)$ to bound $\|M - \Lambda - VW\|_F$ (cf. derivations of Section 3.1).

Standard application of Lagrangian duality tells us that

$$(NMU) \equiv \min_{V, W \geq 0} \sup_{\Lambda \geq 0} L(V, W, \Lambda) \geq \sup_{\Lambda \geq 0} \min_{V, W \geq 0} L(V, W, \Lambda) = \sup_{\Lambda \geq 0} f(\Lambda),$$

where the problem on the left of the inequality is equivalent to our original NMU formulation and the problem on the right is its Lagrangian dual, whose solution will provide a (hopefully tight) lower bound on the optimal (NMU). This new problem is a nondifferentiable optimization problem with the nice property that its objective $f(\Lambda) = \min_{V, W \geq 0} L(V, W, \Lambda)$ is concave and its maximization (over a convex set) is then a convex problem (see [35] and references therein).

We describe in the next section a general solution technique, which consists in repeatedly applying the following two steps:

1. Given multipliers Λ, compute (V, W) to (approximately) minimize $L(V, W, \Lambda)$, i.e., solve (LR_Λ); this is discussed in Section 3.1;
2. Given solution (V, W), update multipliers Λ; this is described in Section 3.2.

3.1. Solving the Lagrangian relaxation problem

The following derivations:

$$\begin{aligned} L(V, W, \Lambda) &= \sum_{i,j} \frac{1}{2} (M - VW)_{ij}^2 + \sum_{i,j} \Lambda_{ij} (VW - M)_{ij} \\ &= \frac{1}{2} \sum_{i,j} M_{ij}^2 - \sum_{i,j} M_{ij} (VW)_{ij} + \frac{1}{2} \sum_{i,j} (VW)_{ij}^2 \\ &\quad + \sum_{i,j} \Lambda_{ij} (VW)_{ij} - \sum_{i,j} \Lambda_{ij} M_{ij} \\ &= \frac{1}{2} \|(M - \Lambda) - VW\|_F^2 - \frac{1}{2} \|\Lambda\|_F^2 \end{aligned}$$

show that minimizing $L(V, W, \Lambda)$ for a fixed Λ is equivalent to minimizing $\|(M - \Lambda) - VW\|_F^2$. Matrix $N = M - \Lambda$ is not necessarily nonnegative, therefore finding $V \geq 0$ and $W \geq 0$ such that $N \approx VW$ is a more general problem than NMF. It is actually studied in detail in [26] (see also [36]), where it is called nonnegative factorization (NF), is formulated as

$$\min_{V \in \mathbb{R}^{m \times r}, W \in \mathbb{R}^{r \times n}} \|N - VW\|_F^2 \quad \text{such that } V \geq 0 \text{ and } W \geq 0, \tag{NF}$$

with $N \in \mathbb{R}^{m \times n}$ and $1 \leq r < \min(m, n)$ and is shown to be NP-hard for any factorization rank (including $r = 1$).

Some standard algorithms for NMF can easily adapted to handle an input matrix that is not nonnegative, i.e., solve a NF problem. For this work, we decided to use a recent technique called hierarchical alternating least squares (HALS), proposed in [22], which alternatively updates each column of V and each row of W with the following optimal closed-form solutions:

$$\begin{aligned} V_{:k}^* &= \operatorname*{argmin}_{V_{:k} \geq 0} \|(M - \Lambda) - VW\|_F^2 \\ &= \max\left(0, \frac{A_{:k} - \sum_{l=1, l \neq k}^{r} V_{:l} B_{lk}}{B_{kk}}\right), \end{aligned} \tag{2}$$

with $A = (M - \Lambda) W^T$ and $B = WW^T$, and

$$W_{k:}^* = \operatorname*{argmin}_{W_{k:} \geq 0} \|(M - \Lambda) - VW\|_F^2 = \max\left(\mathbf{0}, \frac{C_{k:} - \sum_{l=1, l \neq k}^{r} D_{kl} W_{l:}}{D_{kk}}\right), \tag{3}$$

with $C = V^T (M - \Lambda)$ and $D = V^T V$. This can be viewed as a simple method of (block-)coordinate descent (also called alternating variables), which has been shown to perform strikingly well in practice, and much better than the popular multiplicative updates of Lee and Seung (see [4,15,26]). Under some mild assumptions, every limit point of the above alternating scheme is a stationary point [4,26].

The main computational cost of one HALS iteration is the evaluation of A and C: they each require $2mnr$ (floating point) operations. One can check that the resulting total number of operations is $4mnr + O((m + n)r^2)$.

Remark 1. HALS is sensitive to the scaling of the initial matrices. For example, if the initial matrices V and W are chosen such that $VW \gg M$, optimal columns of V and optimal rows of W computed by formulas (2) and (3) at the first step will most likely be equal to zero. This will lead to rank deficient approximations ($V_{:k} W_{k:} = 0$ for some k) and numerical problems (for $V_{:k} = 0$, update of $W_{k:}$ is not well defined and vice versa). If the initial matrices (V, W) are scaled [26], i.e., by ensuring that

$$1 = \operatorname*{argmin}_{\alpha} \|M - \alpha VW\|_F = \frac{\langle M, VW \rangle}{\langle VW, VW \rangle}, \tag{4}$$

where $\langle A, B \rangle = \sum_{i,j} A_{ij} B_{ij} = \text{trace}(AB^T)$, this behavior is in general avoided. All initial matrices used in the following have been scaled.

3.2. Update of the multipliers Λ

The second step of our algorithm consists in updating Λ in order to find better (i.e., higher) solutions to the Lagrangian dual problem. Using the knowledge that any optimal solution (Λ^*, V^*, W^*) of the Lagrangian dual must satisfy the following complementarity slackness conditions

$\Lambda_{ij}^* (M - V^* W^*)_{ij} = 0, \ \forall i,j$ as well as feasibility conditions

$\quad \Lambda_{ij}^* \geq 0$ and $(M - V^* W^*)_{ij} \geq 0,$

we see that the update rule for the multipliers Λ should satisfy the following:

- if $(M - VW)_{ij} > 0$, Λ_{ij} should be decreased and eventually reach zero if $(M - V^* W^*)_{ij} > 0$,
- if $(M - VW)_{ij} < 0$, Λ_{ij} should be increased to give more importance to $(M - VW)_{ij}$ in the cost function, hopefully in order to get a feasible solution such that $(M - V^* W^*)_{ij} \geq 0$.

In the sequel, we use the following rule to update Λ, which satisfies the above requirements:

$\Lambda \leftarrow \max(0, \Lambda - \mu_k (M - VW)), \quad \mu_k \to 0,$

where μ_k is a predefined sequence of step lengths decreasing to zero; Λ can be initialized to zero. This update is inspired from the concept of subgradient methods [37]; in fact, one can easily check that the quantity $(VW - M)$ is a subgradient of

$f(\Lambda) = \min_{V, W \geq 0} L(V, W, \Lambda)$

with respect to Λ, i.e., if $(\overline{V}, \overline{W}) = \text{argmin}_{V, W \geq 0} L(V, W, \overline{\Lambda})$, we have

$f(\Lambda) \leq f(\overline{\Lambda}) + \langle \overline{V}\,\overline{W} - M, \Lambda - \overline{\Lambda} \rangle, \quad \forall \Lambda.$

Two questions now arise:

- Since an iterative algorithm is used to solve (approximately) the Lagrangian relaxation problem (cf. Section 3.1), after how many of these HALS iterations do we stop and proceed to update the multipliers Λ?
- How do we choose the sequence of step lengths μ_k?

Subgradient methods usually assume that the Lagrangian relaxation problem (LR_Λ) can be solved exactly and can guarantee their convergence to an optimal solution provided an appropriate sequence of step sizes is selected (see, e.g., [35]), for example $\{\mu_k\}$ satisfying the conditions

$0 \leq \mu_k \to 0 \quad \text{such that } \sum_{k=0}^{\infty} \mu_k^2 < +\infty \text{ while } \sum_{k=0}^{\infty} \mu_k = +\infty.$

In the sequel, we choose to use $\mu_k = 1/k$, which is such a suitable sequence. However, in our case, we cannot expect to solve (LR_Λ) in a reasonable amount of time since the problem is NP-hard. It would even probably be too expensive to wait for the stabilization of (V, W) (e.g., getting close to a stationary but not necessarily optimal point). We therefore suggest to update (V, W) only a constant number of times T between each update of Λ, which leads to Algorithm L-NMU. Note that because we do not solve (LR_Λ) exactly, Algorithm L-NMU is not guaranteed to converge to an optimal solution of the Lagrangian dual but, as we will see, it produces satisfactory solutions in practice.

Algorithm 1. Lagrangian NMU (L-NMU)

> **Require:** $M \in \mathbb{R}_+^{m \times n}$, $r > 0$, $V \in \mathbb{R}_+^{m \times r}$, $W \in \mathbb{R}_+^{r \times n}$, maxiter, T.
> **Ensure:** (V, W) s.t. $VW \lesssim M$.
> 1: $\quad \Lambda = 0;$
> 2: $\quad$ **for** $k = 1$: maxiter **do**
> 3: $\quad\quad$ Update (V, W) using T iterations of HALS (2)-(3);
> 4: $\quad\quad$ Update $\Lambda \leftarrow \max(0, \Lambda - \frac{1}{k}(M - VW));$
> 5: $\quad$ **end for**

The additional computational cost of one iteration of algorithm L-NMU when compared with one iteration of HALS for NMF consists in the computation of $M - \Lambda$ (needed in step 3) and the update of Λ (at step 4), which require $2mnr + O(mn)$ operations (and, in the special case $r = 1$, $5mn$ operations).

Remark 2. Because convergence is not theoretically guaranteed, Algorithm L-NMU may end up with solutions that do not completely satisfy the underapproximation constraint. Although our numerical experiments show that this has no detrimental influence on the quality of the obtained sparse part-based representations (see Section 4), we give here a simple technique to transform such a solution into a feasible solution. Indeed, it is enough to consider the following QP problem (convex quadratic objective function, linear inequality constraints) which only involves the V factor

$$V^* = \text{argmin}_{V \geq 0, VW \leq M} \|M - VW\|_F^2. \tag{5}$$

Because of its convexity, this problem can be solved up to global optimality in a very efficient manner, and replacing the original V factor by the optimal solution V^* leads to a feasible solution (V^*, W) to (NMU).

Remark 3. Because update rule (5) is exact and computable in practice, it would be natural to consider a simpler algorithm based on its alternative application to the V and W factors, without using the Lagrangian relaxation technique, hoping to converge to a solution of (NMU). Unfortunately, we observed that this is quite inefficient in practice. In fact,

- it is relatively computationally expensive to solve these linearly constrained quadratic programs (with $mn + mr$ and $mn + nr$ inequalities), at least compared to the HALS closed-form update rules (2)–(3);
- since the underapproximation constraint is imposed at each step, this algorithm has much less freedom to converge to good solutions: iterates rapidly get stuck on the boundary of the feasible domain, typically with (too) many zeros and a lower rank. For example, assuming M has one zero in each column, we have that for any positive matrix V the corresponding optimal W is equal to 0:

$\forall j, \ \exists i \text{ s.t. } M_{ij} = 0$
$\quad \Rightarrow \forall j, \exists i \text{ s.t. } \sum_k V_{ik} W_{kj} = 0 \Rightarrow W_{kj} = 0, \quad \forall k,j.$

Therefore, such an algorithm can only work if we decide a priori which values in V and W should be equal to zero, i.e., if we find a good sparsity pattern for the solution, which is precisely where the difficulty of the problem lies. Note that the same behavior is observed if a HALS-type algorithm is used instead of (5) (i.e., updating columns of V and rows of W alternatively): after the update of one column of V, the residual will have one zero in each row (cf. Theorem 2) which will

prevent the other columns of V to be nonzero (except if the sparsity pattern is chosen a priori).

Remark 4. The L-NMU algorithm described above is not particularly well-suited to deal with very sparse input matrices. In fact, one has to store a potentially dense $m \times n$ matrix with the Lagrangian variables Λ. Nevertheless, Berry et al. [38] have obtained encouraging results when applying NMU to sparse anomaly detection problems in text mining. Moreover, it is possible to take advantage of the input sparsity pattern and design a computationally cheaper method. First note that the Lagrangian variables associated with a zero of M will be nondecreasing in the course of the algorithm, since

$$0 \leq (V^{(k)}W^{(k)})_{ij} \text{ and } (M)_{ij} = 0 \Rightarrow \Lambda_{ij}^{(k)} \leq \Lambda_{ij}^{(k+1)},$$

where superscript $^{(k)}$ denotes the solution at step k. Therefore one can significantly reduce the computational cost by defining

$$\Lambda_{ij}^{(k)} = -g(k) \quad \text{for all } i,j \text{ s.t. } M_{ij} = 0,$$

where $g(k)$ is an arbitrary positive nondecreasing function, e.g., $g(k) = \rho^k$ with $\rho > 1$, as is implicitly done in [26].

4. Numerical tests on image datasets

We have argued in Section 2 that NMU is potentially able to extract a better part-based representation of the data and that its factors should be sparser than those of the standard NMF, at the detriment of the approximation error. In this section, we support these claims by reporting results of computational experiments involving two variants of Algorithm L-NMU on several image datasets.

A direct comparison between NMU and NMF is not very informative in itself: while the former will provide a sparser part-based representation, the latter will feature a lower approximation error. This does not really tell us whether the improvements in the part-based representation and sparsity are worth the increase in approximation error. For that reason, we chose to compare NMU with two other sparse nonnegative matrix factorizations techniques, described below, in order to better assess whether the increase in sparsity achieved by NMU is worth the loss in reconstruction accuracy.

4.1. Sparse NMF

We selected and tested the following two sparse nonnegative matrix factorization techniques that are frequently used in the literature.

1. Hoyer describes in [11] an algorithm relying on additional explicit sparsity constraints on the factors, enforced at each iteration by means of a projection. The approximation error is reduced via a combination of projected gradient and multiplicative updates. For our experiments, we use the MATLAB® code provided by the author.[4] It should be pointed out that Hoyer is using a different definition of sparsity: for any nonzero n dimensional vector x, his measure of sparsity $sh(x)$ is defined as

$$sh(x) = \frac{\sqrt{n} - \|x\|_1 / \|x\|_2}{\sqrt{n} - 1} \in [0, 1]. \tag{6}$$

Hence, a vector with a single nonzero entry is perfectly sparse

$$sh([0 \ldots 0 \ k \ 0 \ldots 0]) = 1, \quad \forall k \neq 0,$$

while a vector with all entries equal to each other is completely dense

$$sh([k \ldots k]) = 0, \quad \forall k \neq 0.$$

In our experiments, we report sparsity using both the standard $s(\cdot)$ indicator and Hoyer's $sh(\cdot)$ measure.

2. Instead of enforcing sparsity at every iteration, a sparsity-inducing penalty term can be introduced in the objective function [13]. In particular, it is well-known that adding l_1- norm penalty terms induce sparser solutions (see, e.g., [14,9,15]), and we therefore solve the following problem:

$$\min_{V,W \geq 0} \|M - VW\|_F^2 + \mu_V \|V\|_1 + \mu_W \|W\|_1, \tag{sNMF}$$

where $\|A\|_1 = \sum_{ik} |A_{ik}|$ and μ_V and μ_W are two positive parameters controlling the sparsity of V and W. In order to solve (sNMF), we use the HALS algorithm which can easily be adapted to handle the additional l_1−norm penalty terms (see, e.g., [4,15]). This algorithm will be referred to as sNMF.

Technical details for the first technique are more complicated, but it allows the sparsity of the factors to be chosen a priori. The second technique is conceptually simpler but requires the determination of appropriate penalizing parameters by other means.

4.2. Tested algorithms

Algorithm L-NMU proposed in Section 3 can be used to compute underapproximations for any given factorization rank r. This opens the possibility of building a rank-r underapproximation in several different ways: one simple option consists in applying algorithm L-NMU directly to the rank-r problem—we call this method *global NMU* (G-NMU). Another option consists in applying the recursive technique outlined in the Introduction, used to motivate the introduction of underapproximations. More specifically, this means running algorithm L-NMU successively r times to compute r rank-one approximations, subtracting each approximation from the input matrix before computing the next one—we call this method *recursive NMU* (R-NMU). Note that many other variants are possible (e.g., computing two rank-$r/2$ approximations, computing $r/2$ successive rank-two approximations, etc.) but we only tested the two above-mentioned variants, which represent two extreme cases (no recursion and maximum recursion).

In both cases, our implementation of algorithm L-NMU computes two HALS steps between each update of the multipliers Λ (i.e., we fixed $T = 2$). Most of the computational work done in one iteration of L-NMU consists in computing $M - \Lambda$, performing the two HALS steps and updating Λ; more specifically, one can estimate the computational cost of one iteration of G-NMU to $10mnr + O((m+n)r^2)$ operations, while an R-NMU iteration takes $13mn + O((m+n)r)$ operations (repeated r times in the recursive procedure).

For each dataset, we test five nonnegative factorization algorithms: NMF based on HALS updates (NMF), global NMU (G-NMU), recursive NMU (R-NMU), sparse NMF with l_1−penalty terms (sNMF) and the algorithm of Hoyer. We also report the results of a standard principal component analysis (PCA) to serve as a reference (recall that the approximation error of this unconstrained low-rank approximation, computed here with a singular value decomposition, is globally minimal for the given rank, but that its factors are neither nonnegative nor sparse).

---

[4] This code was downloaded from http://www.cs.helsinki.fi/u/phoyer/software.html.
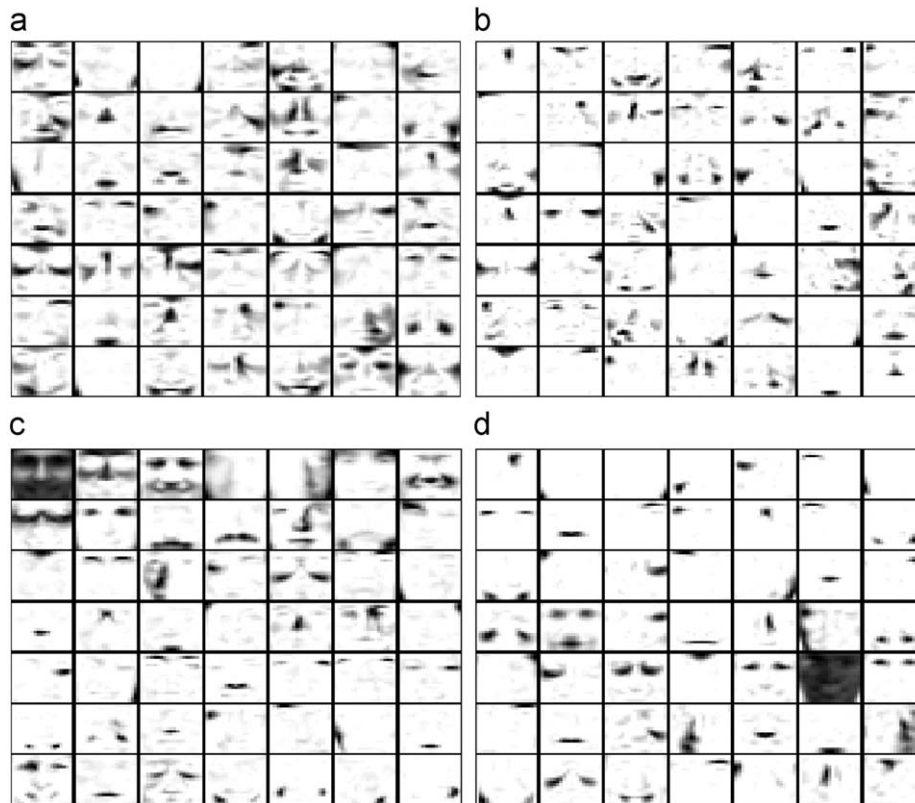
Fig. 3. Basis elements ($V_{:k}$) generated for the CBCL image dataset: (a) NMF, (b) G-NMU, (c) R-NMU and (d) sNMF with sparsity of G-NMU.

4.3. Iteration limits and CPU time

Each of the five iterative algorithms described above requires a limit on the number of its iterations; these limits were chosen in order to roughly allocate the same CPU time to each algorithm. More specifically, the standard NMF was given a 600-iterations limit, which corresponds to the computation of 600 HALS updates. The sparse sNMF, based on a slightly modified HALS update, was also allowed 600 iterations. Because a HALS update involves $4mnr + O((m+n)r^2)$ operations, we can deduce the following iteration budgets for G-NMU and R-NMU from the leading terms in their corresponding operation counts: G-NMU is allowed $600 \times \frac{4}{10} = 240$ L-NMU iterations while R-NMU can take $600 \times \frac{4}{13} \approx 180$ iterations.

An exception to the equal CPU time rule was made for the algorithm of Hoyer. Results obtained after an amount of CPU time similar to that of the other algorithms were too poor to be compared in a meaningful way. Indeed, because this method is based on a projected gradient method and multiplicative updates (both $O(mnr)$ operations per iteration), which are known to converge at a typically much slower rate, a relatively high limit of 1000 iterations had to be fixed, although the resulting CPU time is then much larger than for the other methods (for example, on the CBCL dataset, 600 iterations of HALS took $\sim 80\,\mathrm{s}$ while 1000 iterations of the algorithm of Hoyer needed $\sim 260\,\mathrm{s}$).

4.4. Testing methodology

Recall we decided to test algorithms sNMF and Hoyer to assess the quality of the sparsity-accuracy compromise proposed by our NMU approaches. To achieve this, we decided to pit each NMU variant against a solution of sNMF/Hoyer featuring the same level sparsity, and compare the resulting approximation errors.

Table 1
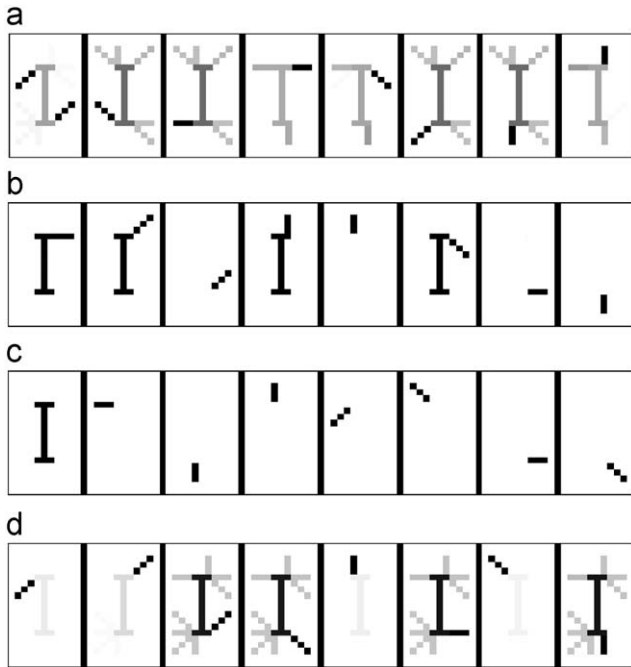Comparison of the relative approximation error and sparsity for the CBCL image dataset.

	Plain	Improved	$s(V)$	$s(W)$	$sh(V)$	$sh(W)$
PCA	7.43	7.43	0	0	22	22
NMF	8.12	8.11	56	11	66	22
G-NMU	12.45	8.76	74	14	74	21
sNMF{G-NMU}	8.68	8.44	74	14	74	30
Hoyer{G-NMU}	9.33	8.78	69	6	73	16
R-NMU	16.42	10.89	53	52	63	64
sNMF{R-NMU}	10.23	9.49	50	50	56	57
Hoyer{R-NMU}	8.83	8.56	54	12	64	22

We therefore report results for eight algorithms on each dataset: PCA, NMF, G-NMU, sNMF with the same sparsity as G-NMU, which we denote by sNMF{G-NMU}, Hoyer{G-NMU}, R-NMU, sNMF{R-NMU} and Hoyer{R-NMU}.

In order to enforce a sparsity similar to the NMU solution in Hoyer's code, we compute the *sh* measure of the NMU factors and input it as a parameter of the method (see description in Section 4.1); note, however, that we could only enforce this for the sparsest of the two NMU factors.[5] In the case of sNMF, sparsity cannot be directly controlled, and penalty parameters are found using the following adaptive procedure, which proved to work well in practice: μ_V and μ_W are initialized to 0.1 and, after each iteration, μ_V (resp. μ_W) is increased by 5% if $s(V)$ (resp. $s(W)$) is below the target sparsity, and is decreased by 5% otherwise.

---

[5] Ideally, we would have imposed sparsity for both factors, but the implementation we used seemed to return poor results in that situation.

Fig. 4. Basis elements for the swimmer image dataset: (a) NMF, (b) G-NMU, (c) R-NMU and (d) sNMF with sparsity of R-NMU.

Table 2
Comparison of the relative approximation error and sparsity for the swimmer image dataset.

Error	Plain	Improved	$s(V)$	$s(W)$	$sh(V)$	$sh(W)$
PCA	37.98	37.98	77[a]	0	67	17
NMF	40.41	40.41	84	45	73	67
G-NMU	47.70	46.85	94	75	85	78
sNMF{G-NMU}	50.52	42.04	89	66	84	73
Hoyer{G-NMU}	42.04	41.91	90	45	80	63
R-NMU	50.92	50.71	98	66	93	65
sNMF{R-NMU}	41.66	41.17	85	66	80	79
Hoyer{R-NMU}[b]	/	/	/	/	/	/

[a] This value is very close to the percentage of zero rows in the matrix M (corresponding to pixels that are equal to zero in all images): in general, PCA factors feature a zero component when all the entries of either one row or one column of the input matrix are equal to zero.

[b] When imposing the sparsity level of R-NMU ($sh(V) = 0.93$), Hoyer's algorithm was not able to converge, probably because it is not well adapted to handle high sparsity constraints. Note that sNMF is also sensitive to high sparsity requirements: high penalty terms sometimes lead to optimal zero factors ($V_{\cdot k} = 0$ for some k), which had to be reinitialized.

All algorithms were run 10 times with the same initial random matrices and only the best solution with respect to the Frobenius norm of the error is reported. When testing with gray-level images, the input matrices M where normalized to have their entries varying between 0 and 1, with 0 representing white and 1 representing black (when trying to decompose M as a sum of parts, this make more sense than the opposite convention, since the dark regions are the constitutive parts of the objects in the image datasets we analyze). Finally, before computing reported sparsity measures of the factors, any sufficiently small[6] entry is rounded to zero (indeed, because algorithms are stopped by the iteration limit before convergence, true zeros are typically not all

---

[6] We declare an entry of a factor to be sufficiently small if it is less than 0.1% of the largest entry in its column.
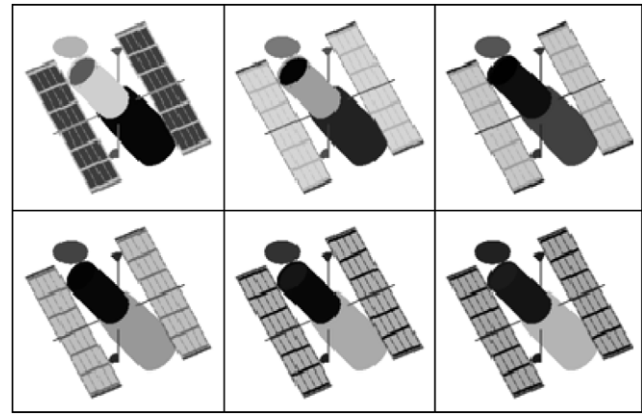
Fig. 5. Sample of Hubble space telescope spectral images.

reached). All tests were run within the MATLAB® 7.1 (R14) version, on a 3 GHz Intel® Core™2 Dual CPU PC.

4.5. CBCL face database

The CBCL face image dataset was used for the illustrative example of Fig. 1 and is made of 2429 gray-level images of faces represented with 19×19 pixels. We look for an approximation of rank $r = 49$.

Fig. 3 displays the basis elements for NMF, G-NMU, R-NMU and sNMF{G-NMU} (which was the best solution obtained in term of sparsity vs. error among all four sNMF and Hoyer variants). Both G-NMU, R-NMU and sNMF achieve a better part-based representation than NMF, generating sparser solutions. An interesting feature of R-NMU is that it extracts parts successively in order of "importance": the first basis element is a "mean" face (which is dense) while the next ones describe different complementary parts (which become sparser as the recursion moves on, cf. Corollary 1 and Theorem 2).

A more quantitative assessment is provided in Table 1, reporting for the eight algorithms tested the relative error (in percent) of their solutions

$$\text{relative error } = \frac{\|M - VW\|_F}{\|M\|_F}$$

in the second column ("Plain") and the corresponding sparsity measures (in percent) of factors V and W in the last four columns.

As expected, PCA returns the smallest error, albeit with very dense factors. NMF already features much sparser factors (slightly half of the entries in V are equal to zero), at the cost of a relatively modest increase in the approximation error ($7.43 \to 8.12$). G-NMU provides an even sparser solution (three quarters of zero entries), increasing again the approximation error ($8.12 \to 12.45$). The factors recursively computed by R-NMU are in comparison not as sparse: as explained above, this is because R-NMU focuses on obtained representative parts, including relatively dense ones for the first few steps of the recursion. However, it features a much sparse weight vectors, giving again more credit to the hypothesis that better parts are extracted. The corresponding approximation error is higher than for other methods, because the intrinsically greedy approach taken by R-NMU is not as efficient as a method that optimizes all the factors simultaneously.

Is the increased sparsity provided by G-NMU worth the increase in approximation error? Looking at the corresponding results for sNMU{G-NMU} and Hoyer{G-NMU}, i.e., for sparse NMF and Hoyer's algorithms with a similar target sparsity, it might seem at first that the answer is negative: the other methods return solutions with similar number of nonzeros (slightly higher
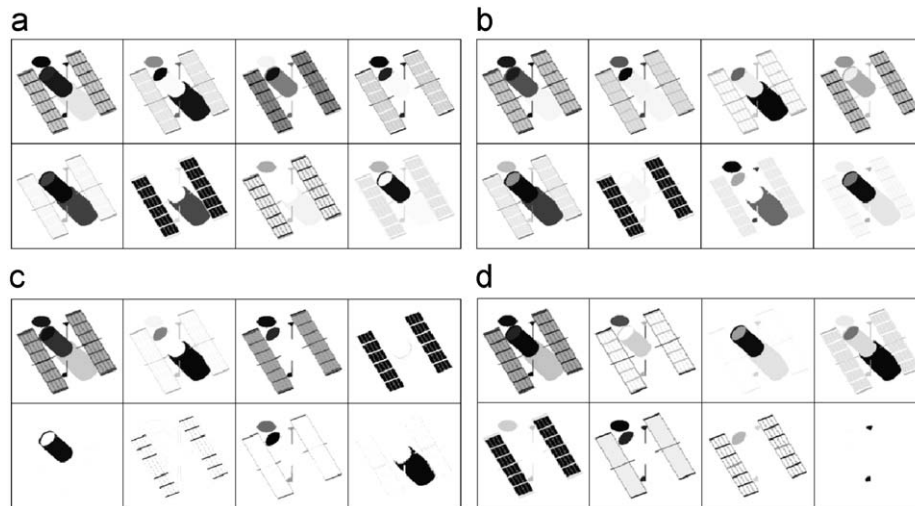
Fig. 6. Basis for the Hubble telescope: (a) NMF, (b) G-NMU, (c) R-NMU and (d) sNMF with sparsity of R-NMU.

Table 3
Comparison of the relative approximation error and sparsity for the Hubble telescope image dataset.

Error	Plain	Improved	$s(V)$	$s(W)$	$sh(V)$	$sh(W)$
PCA	0.01	0.01	57	0	62	25
NMF	0.29	0.29	64	5	57	35
G-NMU	0.52	0.11	64	4	60	31
R-NMU	3.74	1.37	79	30	71	62
sNMF{R-NMU}	0.48	0.37	73	28	66	64
Hoyer{R-NMU}	0.77	0.68	75	0	71	12

Table 4
Comparison of the relative approximation error and sparsity for the Kuls image dataset.

Error	Plain	Improved	$s(V)$	$s(W)$	$sh(V)$	$sh(W)$
PCA	4.36	4.36	0	0	23	15
NMF	4.38	4.38	1	7	9	38
G-NMU	6.27	4.49	3	20	8	48
sNMF{G-NMU}	4.42	4.41	2	20	8	47
Hoyer{G-NMU}	4.71	4.60	2	25	8	53
R-NMU	8.13	5.73	29	31	38	67
sNMF{R-NMU}	5.24	5.01	29	31	32	59
Hoyer{R-NMU}	6.82	6.54	0	71	6	92

for Hoyer) and a lower approximation error (8.68 and 9.33 instead of 12.45). Actually, this was expected: because it tries to return an underapproximation, i.e., factors such that $VW \lesssim M$, the entries in the error term $M - VM$ are mostly nonnegative, while the other techniques, with no underapproximation constraint, obtain a smaller norm of the error by choosing the entries of $M - VW$ to be roughly half negative, half positive. It is therefore not completely fair to compare directly the error of the NMU approach to the other techniques.

In order to compensate for this, a simple rescaling could be used, i.e., multiplying VW by a scalar since $VW \lesssim M$ (cf. Eq. (4)). However, we chose a different procedure that has the advantage of benefiting all algorithms, including those whose error was not suffering from the underapproximation constraint. Once a solution is computed by one of the eight algorithms, we fix the zero entries of V and W and optimize the approximation error, i.e., $\min_{V,W \geq 0} \|M - VW\|_F^2$, on the remaining (nonzero) entries (again, HALS can easily be adapted to handle this situation). In essence, this allows us to compare the sparsity patterns of the different solutions. We perform 100 additional HALS steps on each solution, and report the new relative approximation error in the third column of Table 1 ("Improved"). Note that sNMF and Hoyer's errors are also improved by this procedure; this can be explained by the fact that they were also not directly trying to minimize the approximation error (Hoyer had to take into account its sparsity constraint, and sNMF was influenced by the penalty terms added to the approximation error).

Looking now at the NMU solutions in a fairer comparison, we observe that their approximation error becomes very close to that of sNMF and Hoyer, in particular for G-NMU, and not

very far from the denser NMF, so that we can conclude that the sparsity-approximation error compromise it offers is worth while.

4.6. Swimmer database

For the swimmer image dataset described in Example 1 (256 images with 20×11 pixels), the eight basis elements obtained with the different algorithms are displayed in Fig. 4 and the corresponding approximation errors and sparsity measures are reported in Table 2.

As mentioned earlier, our two NMU algorithms are the only methods able to extract truly independent parts, while NMF and sNMF generate a combination of them. Note, however, that the solution generated by sNMF bears some similarity to the one of G-NMU.

4.7. Hubble space telescope spectral images

The next image dataset consists of 100 spectral images (128×128 pixels) of the Hubble telescope at different frequencies [39,40], see Fig. 5. With the choice $r = 8$, NMF generates a nearly exact factorization (relative error 0.29%), because the spectral reflectance of the Hubble telescope results from the additive linear combination of the reflectance of eight constitutive materials. Fig. 6 and Table 3 provide the visual and computational results for this dataset.

Because NMF is already a nearly exact reconstruction (Table 3), the NMU constraints are somehow redundant: NMF and G-NMU

Fig. 7. Basis for the Kuls image dataset, from top to bottom: sample of images, NMF, G-NMU, R-NMU, sNMF with sparsity of R-NMU.

are basically equivalent and return solutions with very similar sparsity measures (albeit with a slightly lower error for G-NMU). For that reason, sNMF{G-NMU} and Hoyer{G-NMU} return results nearly identical to NMF and are omitted from the table.

Recursive R-NMU extracts parts in order of importance: first, a global picture of the telescope and then its different constitutive parts. This allows it to generate the sparsest solution, with several basis elements representing well-delimited constitutive parts of the telescope not identified by the other methods.

4.8. Kuls illuminated faces

A static scene was illuminated from many directions with a moving light source to produce the Kuls image dataset.[7] It consists of 20 images (64×64 pixels) of a face. Because the images are very similar, most of the information (more than 70%) can be expressed with only one factor. The remaining information resides in the different orientations of the lighting. Computational and visual results for a rank-5 factorization are given by Table 4 and Fig. 7. We observe that NMF and G-NMU obtain similar results: even though they are both able to extract several faces with different lighting orientations, they do not extract a sparse and part-based representation.

R-NMU first extracts a face illuminated from all directions, and then complementary parts representing different orientations of the lighting (successively on the fourth row of Fig. 7: global then light from the right, left, bottom and top). This nice recursive extraction of the information is a direct consequence of the underapproximation constraints. Although sNMF (with the same sparsity requirement as R-NMU) is also able to extract a part-based representation with a slightly better approximation error, only two components are well-identified (left and right lighting mixed with top and bottom lighting).

5. Conclusion

In order to solve the NMF problem in a recursive way, we have introduced a new problem, namely nonnegative matrix under-approximation (NMU), which was shown to be NP-hard using its equivalence with the maximum-edge biclique problem. The additional constraints of NMU are shown to induce sparser factors and to lead naturally to a better part-based representation of the data, while keeping a fairly good reconstruction. We proposed an algorithm based on Lagrangian relaxation to find approximate solutions to NMU.

We tested two factorization methods based on this algorithm, one with full recursion (R-NMU), the other without recursion (G-NMU), on several standard image datasets. After suitable post-processing, we observed that the factors computed by these methods indeed offer a

---

[7] Available at http://www.robots.ox.ac.uk/~amb/.

good compromise between their achieved sparsity and the resulting approximation error, comparable or sometimes superior to that of two standard sparse matrix factorization techniques.

These two variants can be contrasted in the following way: where G-NMU mainly focuses on finding sparse factors with small reconstruction error, in the same spirit as sNMF and Hoyer, R-NMU typically computes an even sparser factorization corresponding to a better part-based representation, albeit with a moderate increase in the reconstruction error (due to the greedy approach). Moreover, this second variant is useful in situations where the factorization rank is not fixed a priori: the fact that it is recursive allows the user to stop the procedure as soon as the reconstruction error becomes satisfactory, without having to recompute a completely different solution from scratch every time a higher-rank factorization needs to be considered.

Acknowledgments

References

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

[2] I. Dhillon, S. Sra, Nonnegative matrix approximations: algorithms and applications, Technical Report, University of Texas (Austin), Department of Computer Sciences, 2006.

[3] K. Devarajan, Nonnegative matrix factorization: an analytical and interpretive tool in computational biology, PLoS Computational Biology 4 (7) (2008) e1000029.

[4] N.-D. Ho, Nonnegative matrix factorization—algorithms and applications, Ph.D. Thesis, Université catholique de Louvain, 2008.

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

[6] D. Lee, H. Seung, Learning the parts of objects by nonnegative matrix factorization, Nature 401 (1999) 788–791.

[7] F. Shahnaz, M. Berry, A.V. Pauca, R. Plemmons, Document clustering using nonnegative matrix factorization, Information Processing and Management 42 (2006) 373–386.

[8] Y. Gao, G. Church, Improving molecular cancer class discovery through sparse non-negative matrix factorization, Bioinformatics 21 (21) (2005) 3970–3975.

[9] H. Kim, H. Park, Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis, Bioinformatics 23 (12) (2007) 1495–1502.

[10] A. d'Aspremont, L. El Ghaoui, M. Jordan, G. Lanckriet, A direct formulation for sparse PCA using semidefinite programming, SIAM Review 49 (3) (2007) 434–448.

[11] P. Hoyer, Nonnegative matrix factorization with sparseness constraints, Journal of Machine Learning Research 5 (2004) 1457–1469.

[12] M. Heiler, C. Schnörr, Learning sparse representations by non-negative matrix factorization and sequential cone programming, Journal of Machine Learning Research 7 (2006) 1385–1407.

[13] S. Li, X. Hou, H. Zhang, Q. Cheng, Learning spatially localized parts-based representation, in: Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition, 2001, pp. 207–212.

[14] H. Kim, H. Park, Non-negative matrix factorization based on alternating non-negativity constrained least squares and active set method, SIAM Journal on Matrix Analysis and Applications 30 (2) (2008) 713–730.

[15] C. Cichocki, A.-H. Phan, Fast local algorithms for large scale nonnegative matrix and tensor factorizations, IEICE Transactions on Fundamentals of Electronics E92-A (3) (2009) 708–721.

[16] G. Golub, C. Van Loan, Matrix Computation, third ed., The Johns Hopkins University Press, Baltimore, 1996.

[17] S.A. Vavasis, On the complexity of nonnegative matrix factorization, SIAM Journal on Optimization 20 (3) (2009) 1364–1377.

[18] C.-J. Lin, Projected gradient methods for nonnegative matrix factorization, Neural Computation 19 (2007) 2756–2779.

[19] I. Dhillon, D. Kim, S. Sra, Fast Newton-type methods for the least squares nonnegative matrix approximation problem, in: Proceedings of SIAM Conference on Data Mining, 2007.

[20] C. Cichocki, R. Zdunek, S. Amari, Non-negative matrix factorization with quasi-Newton optimization, Lecture Notes in Artificial Intelligence, vol. 4029, Springer, Berlin, 2006, pp. 870–879.

[21] D. Chen, R. Plemmons, Nonnegativity constraints in numerical analysis, in: A. Bultheel, R. Cools (Eds.), Symposium on the Birth of Numerical Analysis, World Scientific Press, 2009.

[22] C. Cichocki, R. Zdunek, S. Amari, Hierarchical ALS algorithms for nonnegative matrix and 3D tensor factorization, Lecture Notes in Computer Science, vol. 4666, Springer, Berlin, 2007, pp. 169–176.

[23] D. Lee, H. Seung, Algorithms for non-negative matrix factorization, Advances in Neural Information Processing 13 (2001) 556–562.

[24] C. Cichocki, R. Zdunek, S. Amari, Nonnegative matrix and tensor factorization, IEEE Signal Processing Magazine (2008) 142–145.

[26] N. Gillis, F. Glineur, Nonnegative factorization and the maximum edge biclique problem, CORE Discussion Paper 2008/64, 2008.

[27] B. Levin, On calculating maximum rank one underapproximations for positive arrays, Technical Report, Columbia University, Division of Biostatistics, 1985.

[28] N. Gillis, Approximation et sous-approximation de matrices par factorisation positive: algorithmes, complexité et applications, Master's Thesis, Université catholique de Louvain, 2007 (in French).

[29] R. Albright, J. Cox, D. Duling, A. Langville, C. Meyer, Initializations and convergence for the nonnegative matrix factorization, in: 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2006.

[30] C. Boutsidis, E. Gallopoulos, SVD based initialization: a head start for nonnegative matrix factorization, Journal of Pattern Recognition 41 (2008) 1350–1362.

[31] J. Curry, A. Dougherty, S. Wild, Improving non-negative matrix factorizations through structured initialization, Journal of Pattern Recognition 37 (11) (2004) 2217–2232.

[32] B. Dong, M. Lin, M. Chu, Nonnegative rank factorization via rank reduction, preprint, 2008.

[33] M. Biggs, A. Ghodsi, S. Vavasis, Nonnegative matrix factorization via rank-one downdate, in: 25th International Conference on Machine Learning (ICML), 2008.

[34] R. Peeters, The maximum edge biclique problem is NP-complete, Discrete Applied Mathematics 131 (3) (2003) 651–654.

[35] K.M. Anstreicher, L.A. Wolsey, Two "well-known" properties of subgradient optimization, Mathematical Programming 120 (1) (2009) 213–220.

[36] C. Ding, T. Li, M. Jordan, Convex and semi-nonnegative matrix factorizations, IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (1) (2010) 45–55.

[37] N. Shor, Minimization Methods for Non-differentiable Functions, in: Springer Series in Computational Mathematics, 1985.

[38] M. Berry, N. Gillis, F. Glineur, Document classification using nonnegative matrix factorization and underapproximation, in: Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), 2009, pp. 2782–2785, ISBN: 978-1-4244-3828-0.

[39] P. Pauca, J. Piper, R. Plemmons, Nonnegative matrix factorization for spectral data analysis, Linear Algebra and its Applications 406 (1) (2006) 29–47.

[40] Q. Zhang, H. Wang, R. Plemmons, P. Pauca, Tensor methods for hyperspectral data analysis: a space object material identification study, Journal of the Optical Society of America A 25 (12) (2008) 3001–3012.

About the Author—NICOLAS GILLIS was born in Brussels, Belgium, in 1984. He received a master degree in applied mathematics from Université catholique de Louvain (UCL) in 2007. Since then, he is a Ph.D. student in applied mathematics associated with the Center for Operations Research and Econometrics and the Department of Mathematical Engineering at UCL.

About the Author—FRANÇOIS GLINEUR received engineering degrees from Faculté Polytechnique de Mons (Belgium) and École supérieure d'électricité (France) in 1997, and a Ph.D. in applied sciences from Faculté Polytechnique de Mons in 2001. Since 2002, he is an associate professor at Université catholique de Louvain, member of both the Mathematical Engineering Department and the Center for Operations Research and Econometrics. His research interests revolve mainly around convex optimization, including models, algorithms and applications.