Review article

A survey on deep matrix factorizations

Pierre De Handschutter *, Nicolas Gillis, Xavier Siebert

Department of Mathematics and Operational Research, Faculté Polytechnique, Université de Mons, Rue de Houdain, 9, Mons, 7000, Belgium

ARTICLE INFO

ABSTRACT

Constrained low-rank matrix approximations have been known for decades as powerful linear dimensionality reduction techniques able to extract the information contained in large data sets in a relevant way. However, such low-rank approaches are unable to mine complex, interleaved features that underlie hierarchical semantics. Recently, deep matrix factorization (deep MF) was introduced to deal with the extraction of several layers of features and has been shown to reach outstanding performances on unsupervised tasks. Deep MF was motivated by the success of deep learning, as it is conceptually close to some neural networks paradigms. In this survey paper, we present the main models, algorithms, and applications of deep MF through a comprehensive literature review. We also discuss theoretical questions and perspectives of research as deep MF is likely to become an important paradigm in unsupervised learning in the next few years.

Contents

* Corresponding author.
  E-mail addresses: pierre.dehandschutter@umons.ac.be (P. De Handschutter), nicolas.gillis@umons.ac.be (N. Gillis), xavier.siebert@umons.ac.be (X. Siebert).

1. Introduction

In the current era of data deluge, the automatic extraction of interpretable features in unlabeled data sets is a key challenge. For many years, linear algebra tools have been used to deal with such tasks. Among these techniques, the constrained low-rank matrix approximations (CLRMA) [1] mine relevant information from large data sets and have therefore been drawing attention of numerous researchers. In practice, many data sets appear to be well approximated by low-rank matrices [2], and hence CLRMA are particularly appropriate to extract pertinent information. Within this general framework, some well-known techniques such as principal component analysis (PCA) [3], singular value decomposition (SVD) [4], sparse coding [5], sparse component analysis (SCA) [6], and non-negative matrix factorization (NMF) [7], to name only a few, have been used in many applications for the last decades. These variants mostly differ by the function chosen to measure the quality of the approximation and by the additional constraints considered. Given a set of n data points lying in an m-dimensional space, a matrix $X \in \mathbb{R}^{m \times n}$ is built such that each data point corresponds to a column of X. The goal of a low-rank matrix approximation is to express each data point as a linear combination of a few basis vectors. In other words, one has to find a matrix $W \in \mathbb{R}^{m \times r}$ and a matrix $H \in \mathbb{R}^{r \times n}$ such that each data point can be approximated as

$$X(:, j) \approx \sum_{k=1}^{r} W(:, k)H(k, j) \text{ for } j = 1, \ldots, n,$$

where $W(:, k)$ denotes the kth column of W and corresponds to the kth basis element, and $H(k, j)$ is the weight with which the kth basis element appears in the jth data point. More precisely, $H(:, j)$ is the representation of data point $X(:, j)$ in an r-dimensional linear subspace spanned by W. In matrix form, this approximation, also sometimes called factorization, is written as $X \approx WH$.

On the other hand, deep neural networks have been widely used as deep learning gained success in many supervised classification tasks [8,9] and even in generative models [10]. Their main advantage lies in their ability to combine features in a highly non-linear way but their theoretical foundations remain quite elusive.

At midway between the linear algebra models and the deep neural networks lies deep matrix factorization (deep MF), the core of this paper. The main motivation of deep MF is to combine both interpretability, as in classical matrix factorizations, of which it is an extension, and the extraction of multiple hierarchical features, as allowed by multilayer architectures. Linearity offers impressive interpretability power and easily allows to understand how the features are combined.

One layer matrix approximations are not able to extract multi-level features in complex data sets. The goal of deep MF is to decompose a data matrix $X \in \mathbb{R}^{m \times n}$ as

$$X \approx W_1 W_2 \ldots W_L H_L, \quad (1)$$

where L is the number of layers, $W_l \in \mathbb{R}^{d_{l-1} \times d_l}$ for $l = 1, \ldots, L$ with $d_0 = m$, and $H_L \in \mathbb{R}_+^{d_L \times n}$. The approximation in (1) corresponds to successive factorizations of X:

$$\begin{aligned}
X &\approx W_1 H_1, \\
H_1 &\approx W_2 H_2, \\
&\vdots \\
H_{L-1} &\approx W_L H_L,
\end{aligned} \quad (2)$$

where $H_l \in \mathbb{R}_+^{d_l \times n}$ for all l. Each matrix W_l ($l = 1, \ldots, L$) can be interpreted as the feature matrix of layer l and each H_l can be interpreted as the representation matrix of layer l. In other words, successive factorizations of rank d_l ($1 \leq l \leq L$) are performed such that various recombinations of the features of the first layers would appear in the following ones allowing numerous interpretations of the semantics hidden in the data set.

One of the first applications for which deep MF has proven to be useful is the extraction of facial features, as described in the seminal paper of Trigeorgis et al. [11]. Given a set of n gray-scale facial images, each one described by m pixel values, deep MF extracts several layers of features, each one corresponding to a specific interpretation ranging from low-level features at the first layer to high-level features at the last layer.

Fig. 1 displays the features extracted at each layer by deep NMF (that is, deep MF with nonnegativity constraints on the factors W_l, $l = 1, 2, \ldots, L$, and H_L) on the CBCL faces data set, originally used for standard NMF by Lee & Seung [7]. This data set contains 2429 gray-scale images of 19×19 pixels, with $L = 3$ layers where $d_1 = 100$, $d_2 = 50$ and $d_3 = 25$. At the first layer, 100 low-level features corresponding to very specific and localized facial characteristics are extracted. At the second layer, deep MF extracts 50 meaningful parts of the faces, such as eyebrows, noses, eyes, and mouths. Finally, the last layer exhibits faces made of several facial features, and is more closely related to the identities of the persons. This illustrates the fact that, at each layer, deep MF extracts higher-level features obtained by combining lower-level features from the previous layer.

To compute the solution displayed on Fig. 1, we used a simple multi-layer approach; see Section 3.1. At the first layer, we used the separable NMF algorithm of [12] which provides a sparse solution, and at the next layers, we used the fast NMF algorithm of [13].

In summary, deep MF is able to extract hierarchical facial features corresponding to the columns of $\prod_{l=1}^{i} W_l$ at the ith layer that become less localized as the deep MF unfolds.

Without any constraint on the factors of deep MF, (1) merely degenerates into classical matrix factorization. In this case, the product of the matrices W_l's could be replaced by a single equivalent (without additional particular property) matrix whose rank is less than or equal to the minimum of the d_l's and the factorization is highly non-unique. For example, one could simply replace any W_l by $W_l Q$ and W_{l+1} by $Q^{-1} W_{l+1}$ for any l and any invertible matrix $Q \in \mathbb{R}^{d_l \times d_l}$, and obtain another decomposition of X with the same approximation error but most likely with a rather different interpretation. Therefore, constraints on the factors such
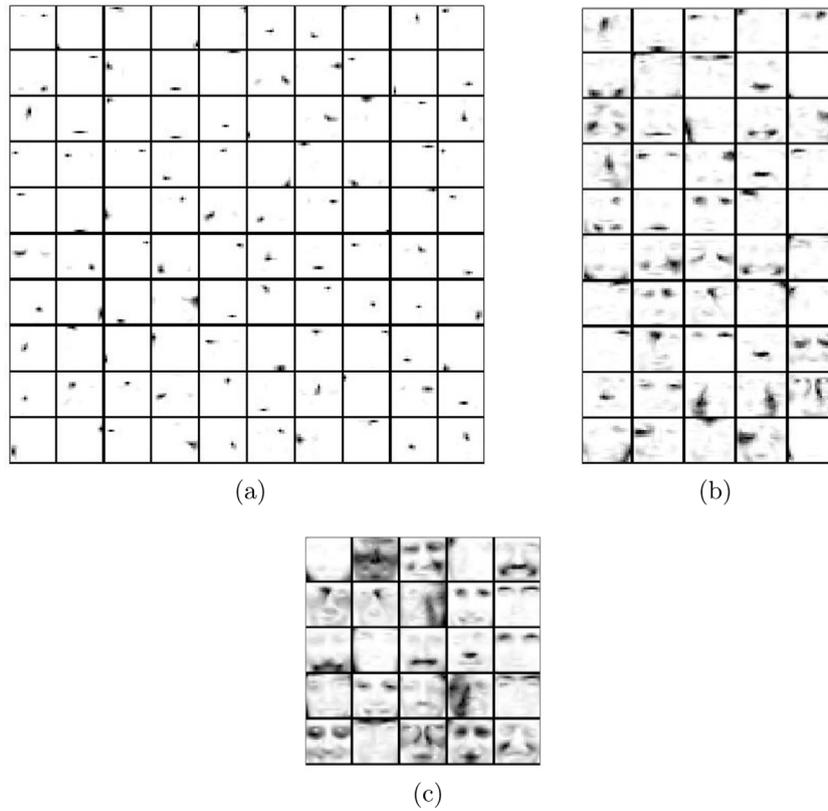
Fig. 1. Application of deep NMF on the CBCL face data set, with $L = 3$, $d_1 = 100$, $d_2 = 50$, and $d_3 = 25$. Each image contains the features extracted a layer: (a) first layer W_1, (b) second layer W_1W_2, and (c) third layer $W_1W_2W_3$.

as non-negativity and sparsity, and/or regularizations should be considered, which results in various deep MF models. Most deep MF models assume the non-negativity of several factors of the decomposition and therefore extend some NMF ideas.

This paper serves as a survey on the recent literature on deep MF. It is organized as follows. We first briefly summarize the main ideas behind CLRMA in Section 2. In Section 3, we present the early multi-level models, namely multilayer MF up to recent deep MF, their regularizations, and the different algorithmic approaches to tackle them. In Section 4, we present the performances of deep MF on two illustrative examples (namely, recommender systems and hyperspectral imaging), and review the main applications. Connections with deep learning are initiated in Section 5 while Section 6 highlights the lack of theoretical guarantees that have come with the models so far. However, contributions from deep linear networks might open new directions of research. Finally, in Section 7, we summarize the identified perspectives of future research and conclude.

2. A brief summary on matrix factorizations

In this section, we recall the basics of matrix factorizations, which will be key to understand deep MF.

Low-rank matrix approximations consist in finding two matrices $W \in \mathbb{R}^{m \times r}$ and $H \in \mathbb{R}^{r \times n}$ such that the product WH approximates as well as possible a data matrix $X \in \mathbb{R}^{m \times n}$ made of n points in dimension m where r, called the rank of the approximation, is generally a small value compared to m and n and is fixed in advance in many practical applications.

A critical aspect of matrix approximations is the choice of the loss metric between X and WH, that is, the way to evaluate how good the approximation is. Most models aim at minimizing a

divergence between the original data matrix and its low-rank reconstruction. More precisely, the β-divergences are usually considered to quantify the fidelity between the original data matrix and its low-rank approximation [14], and a common choice in the community is to minimize the Frobenius norm of the difference between these two matrices, which corresponds to $\beta = 2$. The reason for this choice is twofold: (1) the squared Frobenius norm of a matrix is equal to the sum of its squared elements, and hence render the computation of the derivatives straightforward in the optimization process, and (2) it corresponds to the maximum likelihood estimator in the presence of additive Gaussian noise which is a reasonable model in many applications. Therefore, the standard matrix factorization optimization problem is formulated as

$$\min_{\substack{W \in \mathbb{R}^{m \times r} \\ H \in \mathbb{R}^{r \times n}}} \|X - WH\|_F^2, \tag{3}$$

where $\|A\|_F^2 = \sum_{i,j} A(i,j)^2$ is the squared Frobenius norm of matrix A. This essentially corresponds to PCA (although PCA typically performs mean centering before computing the principal components), and can be computed via the SVD.

A usual feature of the data matrix is that it is entry-wise non-negative, that is, $X(i,j) \geq 0$ for all i, j, which is denoted $X \geq 0$. Many real-world applications record such non-negative measurements, which has led to the development of the so-called non-negative matrix factorization (NMF) model [15]. In NMF, the input data matrix X is element-wise non-negative and in turn, entry-wise non-negativity is required for both factors W and H. NMF has been widely studied, in terms of theoretical guarantees, models and applications [16–20] and is formulated as

$$\min_{\substack{W \in \mathbb{R}^{m \times r} \\ H \in \mathbb{R}^{r \times n}}} \|X - WH\|_F^2 \text{ such that } W \geq 0 \text{ and } H \geq 0, \tag{4}$$

which corresponds to (3) with the additional non-negativity constraints.

A strong advantage of NMF is the interpretability of the factors [18]. The matrix W is often considered as the matrix of features, with each column of W corresponding to a basis vector, while the matrix H corresponds to the activations of each basis vector in each original data point. Especially, if it is also required that the sum of the elements of any column of H is equal to 1, that is, H is column stochastic with $\sum_{j=1}^{r} H(j, k) = 1$ for all k, then the entries of the kth column of H can be interpreted as the proportions in which each feature vector appears in the kth data point. In this sense, NMF can be seen as a soft clustering technique as for all j and k, $H(k, j)$ is the membership indicator of the jth data point in the kth cluster. This model is sometimes referred to as simplex-structured matrix factorization; see [21] and the references therein.

Most of the time, additional properties are enforced for the two factors W and H. This can be translated by hard-coded constraints or through a penalty term called regularizer added to the data fitting term in the objective function. Several models and algorithms have been designed, exploiting geometric or algebraic properties [19]. Among the most widely used techniques, minimum-volume NMF (MinVolNMF) [22–24], sparse NMF [25], and variants of archetypal analysis (AA) [26–28] have led to the best performances. For example, minimum-volume NMF aims at minimizing the volume delimited by the basis vectors, while sparse NMF imposes that the factors only contain a reduced number of non-zero entries. Moreover, these methods start to be supported by theoretical advances, such as identifiability results which provide conditions under which the ground-truth matrices W and H are unique (up to trivial ambiguities such as permutation and scaling); see [24] and the references therein. Some of these variants will be detailed in Section 3.2 as they have been extended to the multi-layer case.

NMF is an NP-hard non-convex problem [29] which is generally solved through an alternated scheme known as block-coordinate descent (BCD), as described in Algorithm 1. This consists in alternatively optimizing one of the two factors of (4) while keeping the other fixed. Note that the corresponding subproblems, that is, *update_W* and *update_H* in Algorithm 1, are convex nonnegative least squares problems which are efficiently solvable.[1]

Algorithm 1 Two-block coordinate descent to solve NMF

Input: Nonnegative matrix X, rank r of the factorization
Output: Matrices W and H minimizing Eq. (4)
 1: Compute initial matrices $W^{(0)}$ and $H^{(0)}$, $t = 1$
 2: **while** *stopping criterion* not met **do**
 3:     $W^{(t)} = \text{update\_W}(X, W^{(t-1)}, H^{(t-1)})$
 4:     $H^{(t)} = \text{update\_H}(X, W^{(t)}, H^{(t-1)})$
 5:     $t = t + 1$
 6: **end while**

---

3. Deep MF models and algorithms

Although CLRMA such as NMF with negativity constraints and SCA with sparsity constraints lead to a compact and meaningful representation of the input data, they are limited by the shallowness of the representation. Such techniques are only able to extract a single layer of features, preventing to reveal hierarchical features. While the standard matrix factorization decomposes the

data matrix in only two factors, deep MF, inspired by the success of deep learning, is able to extract several layers of features in a hierarchical way, giving new insights in a broad range of applications.

Deep MF considers a product of matrices W_l's ($l = 1, \ldots, L$) in place of a single matrix W in the approximation; see (1). As constraints on the factors of this decomposition are necessary to make the model meaningful (see Section 1), the next sections present various models and algorithms of deep MF. We first present the evolution from the early multi-layers models to the recent deep models in Section 3.1. Then, in Section 3.2, we describe the main variants, which are inspired from those of classical matrix factorizations. Section 3.3 describes the possible algorithmic choices and briefly discusses the computational cost.

3.1. From multilayer MF to deep MF

The first model extending CLRMA to several levels is multilayer NMF proposed by Cichocki et al. in 2006 [30,31]. Based on the hierarchical factorizations of a non-negative data matrix $X \in \mathbb{R}_+^{m \times n}$ as described by (2), multilayer MF decomposes X in a sequential manner. At the first layer, a low-rank factorization of X is computed such that $X \approx W_1 H_1$. At the second layer, the matrix H_1 is factorized as $H_1 \approx W_2 H_2$, and so on until H_{L-1} is decomposed as $W_L H_L$; see Algorithm 2. Moreover, all the factors of the decomposition are constrained to be non-negative.

Algorithm 2 Early multilayer NMF [30]

Input: Non-negative data matrix X, number of layers L, inner ranks d_l
Output: Matrices $W_1, \ldots, W_L$ and $H_1, \ldots, H_L$
 1: $Y = X$
 2: **for** $l = 1, \ldots, L$ **do**
 3:     (W_l, H_l) = Algorithm 1 (Y, d_l)
 4:     $Y = H_l$
 5: **end for**

However, the multilayer NMF of [31] does not investigate much the hierarchical power of deep schemes as the decomposition is purely sequential, that is, multilayer NMF is equivalent to a succession of successive single layer NMFs, with a single forward pass. More precisely, Algorithm 2 consists in sequentially minimizing the reconstruction errors $\|H_{i-1} - W_i H_i\|_F^2$ for all $i = 1, \ldots, L$ with $H_0 = X$, but it does not involve a global cost function. In other words, the error is minimized layer by layer as in (4), but there is no retroaction of the last layers on the first ones.

A key improvement was achieved by the papers of Trigeorgis et al. who introduced deep MF [11,32]. The data matrix X still undergoes successive factorizations as in (2), but the breakthrough lies in the way the optimization of the factors W_l's and H_l's is performed. The main algorithmic novelty of deep MF (Algorithm 3) compared to multilayer MF (Algorithm 2) is the iterative updates of the factors: As opposed to multilayer MF that uses a purely sequential approach, deep MF not only propagates the information from the first, more abstract layer, to the last, more refined layer, but also propagates the information in the reverse direction.

The following error function involving the factors of all layers is considered

$$\mathcal{L}(W_1, W_2, \ldots, W_L; H_L) = \|X - W_1 W_2 \ldots W_L H_L\|_F^2, \quad (5)$$

and a block-coordinate descent strategy is used to iteratively update all the factors. The deep MF algorithm [11] is described

---

[1] They can be solved for example in Matlab using the function `lsqnonneg`.
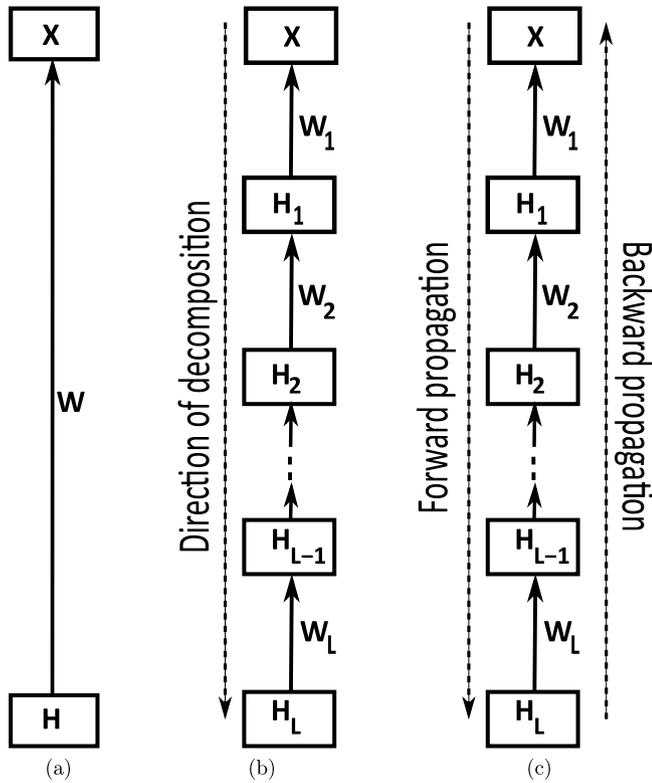
Fig. 2. (a) MF, (b) multilayer MF [30], (c) deep MF [11]. An arrow means a matrix product is performed: $\boxed{H} \xrightarrow{W} \boxed{X}$ means that H is multiplied by W to approximate X.

in Algorithm 3, and illustrated on Fig. 2(c). In Algorithm 3, *arg reduce* means that the factor is updated through some algorithm (see Section 3.3) that (typically) decreases the objective function for several iterations.

Algorithm 3 Deep semi-NMF [11]

Input: Data matrix X, number of layers L, inner ranks d_l
Output: Matrices $W_1, \ldots, W_L$ and $H_1, \ldots, H_L$
1: Compute initial matrices $W_l^{(0)}$ and $H_l^{(0)}$ for all l through a sequential decomposition of X (for example Algorithm 2)
2: **for** $k = 1, \ldots$ **do**
3:   **for** $l = 1, \ldots, L$ **do**
4:     $A_l^{(k)} = \prod_{j<l} W_j^{(k)}$
5:     $B_l^{(k)} = \begin{cases} H_L^{(k-1)} & \text{if } l = L \\ W_{l+1}^{(k-1)} H_{l+1}^{(k-1)} & \text{otherwise} \end{cases}$
6:     $W_l^{(k)} = \underset{W}{\arg\,\mathrm{reduce}} \|X - A_l^{(k)} W B_l^{(k)}\|_F^2$
7:     $H_l^{(k)} = \underset{H \geq 0}{\arg\,\mathrm{reduce}} \|X - A_l^{(k)} W_l^{(k)} H\|_F^2$
8:   **end for**
9: **end for**

Several comments can be formulated:

- First, the work of Trigeorgis et al. was inspired by semi non-negative matrix factorization (semi-NMF) [33], a variant of NMF where only one factor, typically H, must contain non-negative entries while the other W is allowed to contain mixed-sign elements. Therefore, this model should rather be called deep semi-NMF as the W_l's are not directly constrained to be non-negative, and the matrix X is not required

to have non-negative entries. However, one should keep in mind that each factor $H_l \approx W_{l+1} H_{l+1}$ is required to be non-negative, which implies an implicit constraint on the W_l's.

In practice, the requirement for non-negativity constraints on the basis vectors W_l's depends on the application: as most physical systems record non-negative data, it often makes sense to impose the non-negativity of the basis vectors. Therefore, if non-negativity of the basis vectors is meaningful, one can easily modify the model by adding nonnegativity constraints on the W_l's, and modifying line 6 of Algorithm 3.

- Second, to initialize all factors, a forward decomposition of the input matrix is employed, as done in Algorithm 2. Once all the factors are initialized, the updates of all matrices as in Algorithm 3 are performed until some stopping criterion is met.

- Third, the attentive reader may have noticed that Algorithm 3 does not correspond to applying a BCD method on (5) by optimizing the factors $(W_1, \ldots, W_L, H_L)$ alternatively. In fact, the nonnegative matrices H_l $(l = 1, \ldots, L-1)$ are intermediate variables that do not appear in (5). However, one needs to remember the underlying sequential decomposition of (2): as $H_l \approx W_{l+1} H_{l+1}$ $(l = 1, \ldots, L-1)$ are constrained to be non-negative, they have a dedicated update rule. However, this raises important research questions that have not been investigated much. In particular, other possibilities in the expression of $B_l^{(k)}$ are possible; for example, [34] considers $B_l^{(k)} = (\prod_{j>l} W_j^{(k-1)}) H_L^{(k-1)}$ while simply setting $B_l^{(k)} = H_l^{(k-1)}$ also makes sense, without a clear motivation why one should be preferred over the other. Moreover, how does replacing the function to minimize at line 7 by $\|H_{l-1}^{(k)} - W_l^{(k)} H\|_F^2$ change the iterates H_l's? If non-negativity constraints are imposed on the W_l's, a classical BCD can be applied to optimize alternatively the factors of (5), as described in Algorithm 4, where only the variables $(W_1, \ldots, W_L, H_L)$ are alternatively updated. Note that nonnegativity constraints can be replaced with other constraints such as sparsity.

Algorithm 4 BCD to minimize Eq. (5)

Input: Data matrix X, number of layers L, inner ranks d_l
Output: Matrices $W_1, \ldots, W_L$ and H_L minimizing Eq. (5)
1: Compute initial matrices $W_l^{(0)}$ for all l and $H_L^{(0)}$
2: **for** $k = 1, \ldots$ **do**
3:   **for** $l = 1, \ldots, L$ **do**
4:     $A_l^{(k)} = \prod_{j<l} W_j^{(k)}$
5:     $B_l^{(k)} = (\prod_{j>l} W_j^{(k-1)}) H_L^{(k-1)}$
6:     $W_l^{(k)} = \underset{W \geq 0}{\arg\,\mathrm{reduce}} \|X - A_l^{(k)} W B_l^{(k)}\|_F^2$
7:   **end for**
8:   $H_L^{(k)} = \underset{H \geq 0}{\arg\,\mathrm{reduce}} \|X - W_1^{(k)} \cdots W_L^{(k)} H\|_F^2$
9: **end for**

- Finally, the choice of the loss function itself is not obvious. In CLRMA, this issue has been investigated thoroughly, and several strategies exist, based on the statistic of the noise, or cross validation, among others [35]. In deep MF, the question of the structure of the loss function also arises. Is a loss function of the type

$$D(X, W_1 W_2 \ldots W_L H_L),$$

where $D(A, B)$ is a similarity measure between two matrices A and B, a good choice? Or would a loss function that balances the contribution of each layer, such as

$$\mathcal{L} = D(X, W_1 H_1) + \lambda_1 D(H_1, W_2 H_2) + \cdots$$
$$+ \lambda_{L-1} D(H_{L-1}, W_L H_L),$$

be more appropriate? This question has not been addressed yet, to the best of our knowledge. Moreover, most works in the deep MF literature have only considered the Frobenius norm. It would be worth to investigate other similarity measures such as the Kullback–Leibler and Itakura–Saito divergences, which have been shown to be particularly appropriate for specific applications in the case of standard NMF [36,37].

To sum up, a comparison of one-layer matrix factorization, multilayer MF [30] and deep MF [11] is illustrated on Fig. 2. Multilayer MF on Fig. 2(b) and deep MF on Fig. 2(c) both perform several levels of decomposition but the key difference is the iterative nature of the update rules in deep MF, while the decomposition is only sequential, that is, unidirectional, in multilayer MF.

3.2. Variants and regularizations of deep MF

Beside the standard models presented in the previous section, some variants have been studied in the recent literature. These variants consist in adding constraints on the factors or enforcing some properties, and are mostly inspired from CLRMA. As highlighted in Section 1, without any additional constraints on the factors, deep MF admits highly non-unique decompositions. The uniqueness of the solution is critical to ensure reproducibility and interpretability of the results. Depending on the application at hand, various constraints and regularizations can be used. In this section, we briefly review some of these models. In many of them, non-negativity is assumed on the factors, and the variants are therefore closely related to various NMF models.

3.2.1. Deep orthogonal NMF

Orthogonal NMF (ONMF) [38] is a variant of NMF which imposes that the matrix H is nonnegative and row-wise orthogonal, that is, $H \geq 0$ and $HH^T = I_r$ where I_r is the identity matrix of dimension r. In other words, all rows of H are orthogonal to each other, and their l_2 norm is equal to 1. It is easy to see that these constraints imply that there is at most one non-zero value in each column of H. Hence each data point is only associated to one basis vector (one column of W), and ONMF is equivalent to a weighted variant of spherical k-means [39], which is a hard clustering problem. ONMF therefore imposes that each data point belongs to a single cluster which is represented by a single basis vector. This allows a very straightforward interpretation of ONMF factors: the columns of W are cluster centroids, while the columns of H assign each data point to its closest centroid (up to a scaling factor).

A relaxation of the orthogonality constraint consists in adding a penalty term of the form $\sum_{j<k} H(j, :)H(k, :)^T$ to the objective function. This is referred to as approximately orthogonal NMF (AONMF) [40].

The deep version of ONMF was introduced in [41] and enriched in [42]. The decomposition is slightly different than in multi-layer and deep MF because rather than having the activation matrices H_l's successively decomposed, they decompose the features matrices W_l's:

$$X \approx W_1 H_1,$$
$$W_1 \approx W_2 H_2,$$
$$\vdots$$
$$W_{L-1} \approx W_L H_L,$$

leading to $X \approx W_L H_L \cdots H_1$, with each H_l constrained to be nonnegative and row-wise orthogonal, that is, $H_l \geq 0$ and $H_l H_l^T = I_r$ for all l. Similarly, deep AONMF adds a penalty to the objective that minimizes the inner products $H_l(j, :)H_l(k, :)^T$, for all $j \neq k$ in each layer l. Applying the successive decompositions over the basis matrices W_l's rather than the activation matrices H_l's as in [11] seems more natural: it allows to directly interpret the basis vectors of a given layer as combinations of the basis vectors of the next layer. For example, if the ranks d_l's are chosen such that $d_L = d_{L-1} - 1$, $d_{L-1} = d_{L-2} - 1, \ldots, d_2 = d_1 - 1$, each layer will merge two clusters of the previous layer while keeping the others unchanged, and hence deep ONMF performs a hierarchical clustering. This will be illustrated on two showcase examples in Section 4.1.

3.2.2. Deep sparse MF

A very common constraint considered in CLRMA is the sparsity of some factors, referred to as SCA and closely related to dictionary learning (see Section 3.2.7). It consists in limiting the number of non-zero elements of W and/or H. Many papers have studied the case of one-layer sparse MF, see for example [25,43–46] among others. The goal of sparse MF is to render the factors more interpretable. In particular, the fact that each column of H contains only a few non-zero entries means that each data point is associated with a few basis vectors.

The extension of sparse NMF to the deep setting was proposed in [47]. Based on (2), an ℓ_1 norm penalty is considered either on each column of the matrices W_l's and/or on each column of the matrices H_l's. Similarly to shallow sparse MF, the goal of sparse deep MF is to obtain sparse and easily interpretable factors at each layer. The subproblems w.r.t. the regularized factor can be efficiently solved for example through a proximal gradient descent method, such as the (fast) iterative shrinkage thresholding algorithm ((F)ISTA) [48]. Note that a normalization of the other factor should be used to avoid a pathological case where the entries of the factor for which sparsity is promoted tend to zero while those of the other factor tend to infinity, because of the scaling degree of freedom in such decompositions ($W_l H_l = (\alpha W_l)(H_l/\alpha)$ for any $\alpha > 0$). Furthermore, using the same regularization parameter for all columns of a factor at a given layer is discouraged. In practice, several regularization parameters can be tuned automatically to ensure balanced levels of sparsity [49]. Another sparse framework, inspired by multilayer NMF [30], consists in adding a regularizer based on the Dirichlet distribution on the columns of the factors [50].

There exists many ways to impose sparsity on the factors, such as ℓ_0 norm [51] or $\ell_{1/2}$ norm [52] regularizations, among others. Inspired by deep learning, dropout could also enforce sparsity. Dropout [53] consists in randomly "dropping" some activations during the learning process to improve generalization. It has recently been employed for one layer NMF [54], and was shown to be equivalent to a deterministic low-rank regularizer [55]. It would be interesting to see to what extent dropout might regularize deep MF networks as well. Therefore, deep sparse MF has not yet been explored to its full extent.

3.2.3. Deep non-smooth NMF

Non-smooth NMF (nsNMF) [56] consists in using a so-called smoothing matrix S between W and H in NMF which has the form

$$S = (1 - \theta)I_r + \frac{\theta}{r} e e^T,$$

where e is the vector of all ones of appropriate dimension. Note that nsNMF reduces to NMF when $\theta = 0$. The parameter $\theta \in [0, 1)$ promotes the sparseness of both W and H. Let us briefly explain why. We have $WS = (1 - \theta)W + \theta \bar{w} e^T$ where $\bar{w}$ is the

average of the columns of W, that is, $\bar{w} = \frac{1}{r}We$, and $\bar{w}$ is denser than any column of W. When $\theta > 0$, WS therefore moves the columns of W towards $\bar{w}$, and W is sparser than $\tilde{W} = WS$.

Deep nsNMF (dnsNMF) [34] introduces a smoothing matrix at each layer of (2), with a common fixed parameter θ, that is, $X \approx W_1 S_1 H_1$ and $H_{l-1} \approx W_l S_l H_l$ for all $l = 2, \ldots, L$ with $S_l = (1 - \theta)I_{d_l} + \frac{\theta}{d_l}ee^T$. Note that multilayer non-smooth NMF was already proposed in 2013 in [57] to empirically show how a multilayer architecture, very similar to the one proposed by Cichocki et al. [31], is able to extract features in a hierarchical way in the context of text mining, but they did not use any backpropagation strategy.

3.2.4. Deep archetypal analysis

Archetypal analysis (AA) [26], also known as convex NMF [33] is a variant of NMF in which the basis vectors are constrained to be convex combinations of the data points. In other words, in addition to the constraints stated in (4), one should have $W = XA$ where $A \geq 0$ and $A^T e = e$. Intuitively, the basis vectors are restricted to lie in the convex hull of the data points, and can be interpreted as extremal points of the data set. Although the fitting error might be higher than in standard NMF, the closeness of the archetypes to the convex hull of the data confers them a better interpretability.

The first proposal of deep AA was made in [58], to the best of our knowledge, for the acoustic scene classification task. Given a data matrix X composed of n temporal frames characterized by an m-dimensional features vector, a discriminative representation is learnt through successive AA decompositions performed in a greedy forward way:

$$
\begin{aligned}
X &\approx XA_1H_1, \\
H_1 &\approx H_1A_2H_2, \\
&\vdots \\
H_{L-1} &\approx H_{L-1}A_LH_L.
\end{aligned}
$$

However, schemes including a 'backpropagation' stage do not seem to have been tested yet for deep AA.

Finally, the non-deterministic deep AA of Keller et al. [59] approximates the data points by samples drawn from a Gaussian distribution whose parameters are learnt through a deep encoding phase and is based on the deep variational information bottleneck framework [60]. Since the model is probabilistic and non-linear, the spirit is quite different from the one of deep MF models presented previously.

Closely related to AA, concept factorization (CF) consists in approximating the basis elements as linear combinations of the data points, the difference with AA lies in the absence of the sum-to-one constraints on both A and H. Again, the first model containing several levels of decomposition was purely sequential [61], and was outperformed by more recent approaches based on the deep MF algorithm; see for example [62]. We refer the reader to [63] for a comprehensive review of shallow and deep CF.

3.2.5. Semi-supervised settings

While the models presented so far were all unsupervised, some deep MF models are able to cope with available prior information in a semi-supervised fashion, such as deep weakly-supervised semi-NMF [11]. To handle side information, a weighted graph is built at each layer, where the nodes are the data points and two nodes are connected by an edge if they share the same label. In the simplest case, the graph weights, denoted by the $n \times n$ symmetric matrix G_l for the lth layer, are binary, that is, $G_l(i, j) = 1$ if $X(:, i)$ and $X(:, j)$ share the same label w.r.t. the

features extracted at layer l. A smoothness regularization term is added to the loss function of (5) with the form:

$$
\sum_{l=1}^{L} \lambda_l \sum_{\substack{j,k=1 \\ j \neq k}}^{n} \|H_l(:, j) - H_l(:, k)\|_2^2 G_l(j, k) = \sum_{l=1}^{L} \lambda_l \mathrm{Tr}(H_l L_l H_l^T) \quad (6)
$$

where $\mathrm{Tr}(.)$ denotes the trace of a matrix, that is, the sum of its diagonal elements, and $L_l = D_l - G_l$ is the Laplacian matrix at layer l with D_l a diagonal matrix such that $D_l(j, j) = \sum_{k=1}^{n} G_l(j, k)$ for $j = 1, \ldots, n$. Intuitively, (6) enforces the hidden representations $H_l(:, j)$ and $H_l(:, k)$ of data points j and k that share the same label at layer l to be as close as possible.

When the available information is such that each data point might be associated with several labels, a dual-hypergraph Laplacian is built to grasp richer underlying information and is such that an edge can connect any number of vertices [64].

3.2.6. Deep tensor decomposition

The extension of deep MF to tensors, that is, arrays of more than two dimensions, has not yet been much investigated. The analog of shallow MF in the tensor world is the canonical polyadic decomposition (CPD), which decomposes a tensor as the sum of rank-one tensors; see [65] and the references therein. Given a tensor $\mathcal{T} \in \mathbb{R}^{l_1 \times l_2 \times \cdots \times l_K}$ where K is the dimension of the tensor, the CPD of rank R aims at finding the vectors $a_j^{(i)} \in \mathbb{R}^{l_i}$ ($i = 1, \ldots, K, j = 1, \ldots, R$) such that

$$
\mathcal{T} \approx \sum_{j=1}^{R} a_j^{(1)} \circ a_j^{(2)} \ldots \circ a_j^{(K)}, \quad (7)
$$

where the $\circ$ operator denotes the outer product. The principle of a CPD is illustrated on Fig. 3 for a 3-dimensional tensor.

An attempt of multilayer tensor decomposition has been proposed in [66] for contextual-aware recommender systems (CARS), and in [67] for audio sources separation in a mixed scene, as an extension of the matrix model of [68]. Deep tensor decompositions have also been introduced for action recognition in [69] and for the optimization of convolutional neural nets in [70].

However, all these models are quite different from each other and do not resemble deep MF as defined in this paper. No general framework has been introduced for deep tensor decompositions, and it would be worth investigating deep CPD. A particularly interesting feature of CPD is that it has weak identifiability conditions (see for example [71] and the references therein), as opposed to standard MF, which could be leveraged in the deep setting.

3.2.7. Related models

In this section, we briefly introduce two models closely related to MF that have also been extended to a deep setting, namely transform learning and dictionary learning. There are not the main focus of the survey so we encourage the interested reader to look at the references for more details.

Given an input data matrix $X \in \mathbb{R}^{m \times n}$, the goal of dictionary learning (DL) is to find a dictionary $D \in \mathbb{R}^{m \times r}$, whose columns are referred to as atoms, and a representation matrix $H \in \mathbb{R}^{r \times n}$ such that each column of H is sparse, typically k-sparse (that is, with at most k non-zero entries, where k is a parameter). When the number r of atoms is smaller than m, the dictionary is said to be undercomplete and DL is equivalent to SCA. However, DL typically looks for overcomplete dictionaries with $m \ll r$. A particularly interesting variant of DL is the convolutional sparse coding (CSC) where the atoms are convoluted with the representation matrix H. In [5], Papyan et al. proposed deep CSC where successive factorizations are performed exactly as in (2), with sparsity constraints on the H_l's; the columns of H_l are required
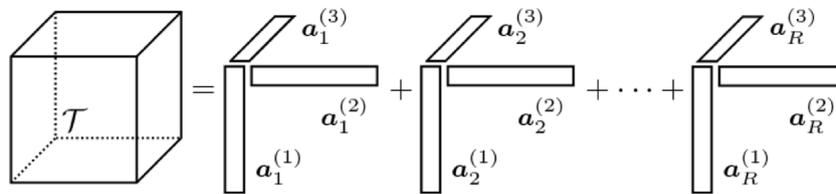
Fig. 3. Canonical polyadic decomposition of a 3-D tensor.

to be k_l-sparse, where k_l is the desired level of sparsity at layer l. Similarly to deep MF, this decomposition allows a hierarchical interpretation of the dictionaries extracted. When a sequential thresholding algorithm is applied to enforce sparsity, deep CSC is equivalent to the forward pass of a convolutional neural network (CNN).

Transform learning [72] also shares similarity with MF. Given a matrix X, it consists in premultiplying X with W to obtain H, such that $WX \approx H$. The matrix H is promoted to be sparse through an ℓ_1 norm regularizer while W is constrained to be full rank. Recently, a sequential multilayer transform learning framework was proposed by Maggu and Majumdar [73], similar to the multilayer idea of Cichocki et al. [30]. At the first layer, the approximation $W_1 X \approx H_1$ is considered. Then, H_1 is premultiplied such that $W_2 H_1 \approx H_2$, and the process continues until W_L and H_L are found, with all H_l's sparse and all W_l's full-rank. Similarly to deep MF, the matrices H_l's are hierarchical representations of the original data points that can be used for further clustering.

3.3. How to solve deep MF?

In this part, we briefly describe the initialization techniques as well as the algorithms that can be used to solve the sub-problems w.r.t. either W_l or H_l at lines 6 and 7 of Algorithm 3. As these algorithms are standard optimization techniques, we refer the reader to previous surveys [18,20] for more details and references on their applications in CLRMA problems. We also discuss the choice of several parameters such as the number of layers L and the inner ranks d_l's.

3.3.1. Initializations

The most commonly used initialization of deep MF consists in applying a sequential decomposition of the data matrix X, but there is no guarantee about the quality of this initialization. For the initialization of each factor W_l and H_l of each layer l, several routines have been presented in the literature; for example random initializations, initializations based on the SVD of X [11], or column subset selection methods that initialize W with columns of X [74]. The study of initialization techniques dedicated to deep MF is still an open direction of research.

3.3.2. Algorithms

Similarly to standard NMF, most algorithms for deep MF consist in alternatively updating each factor while keeping the others fixed as in Algorithm 3. The stopping criterion can either be a maximum number of iterations, a sufficient decrease of the loss function, or a sufficient modification of the factors between two consecutive iterations.

The subproblems w.r.t. either W_l or H_l for any l are typically solved using standard first-order optimization algorithms. Trigeorgis et al. [11] used a closed-form expression for the W_l's and a multiplicative update (MU) for the H_l's. MU is a well-known algorithm to solve NMF [75], and was also proposed to solve a sequential multilayer MF in [76]. Other techniques such as projected gradient descent (PGD) method [77], possibly combined with an acceleration scheme, such as Nesterov's one [78]

are widely used; see for example [34,47]. PGD, a well-known first-order method to solve constrained optimization problems, is an extension of gradient descent (GD) where the iterates are projected on the feasible set at each iteration. The acceleration consists in adding a momentum term to the gradient step to allow faster convergence. Another standard optimization scheme is the alternating direction method of multipliers (ADMM) which consists in reformulating the problem by decoupling the variables, and minimizing the augmented Lagrangian. It is standard in the CLRMA literature [79], and it has also been used for constrained deep MF in [80].

Since most deep MF algorithms are based on first-order methods, their computational cost is linear w.r.t. the size of the input data and the ranks, and hence these methods are scalable. In other words, the computational cost grows linearly with the number of data points, the dimension of the data, and the rank of the factorizations. For example, the computational cost of the algorithm of Trigeorgis et al. [11] requires $\mathcal{O}(Lt(mnd + (m+n)d^2))$ operations, where t is the number of iterations and $d = \max_{l=1,\dots,L} d_l$.

3.3.3. Parameters

The choice of the parameters of deep MF model, and in particular the number of layers and the inner ranks, mainly depends on the application. In most cases, the number of layers used for the results reported in the literature does not exceed three. Moreover, the ranks tend to be chosen in decreasing order as the first layers of the model are expected to capture attributes with a larger variance, thus requiring a larger capacity to encode them, while the last layers capture attributes with a lower variance [11]. This observation is also derived from the analogy with autoencoders: the inner layer of an autoencoder is generally the one that contains fewer units as the goal is to obtain a compact representation of the input data; see Section 5 for more details. On the other hand, when the decomposition is performed on the features matrices such as in 3.2.1, the ranks should also be chosen in decreasing order [42]: given $W_1 \in \mathbb{R}^{m \times d_1}$ with d_1 columns in dimension m, it only makes sense to approximate the columns of $W_1 \approx W_2 H_2$ as linear combinations of $d_2 \leq d_1$ columns of W_2 otherwise there exists an infinity of solutions; see Section 4.1 for two numerical examples. Therefore, there is no compromise over the choice of these parameters and their values mainly depend on the application at hand. Of course, there is an implicit link between the number of layers and the choice of the ranks since, in general, each new layer has a rank equal to at most the rank of the previous layer minus 1.

4. Applications

We now describe several applications for which deep MF is useful.

CLRMA has already been used successfully for countless real-world applications, and deep MF models have contributed to improve the performances. However, a clear definition of deep MF is absent within the community, and very diverse uses of this terminology have been used. Besides, [81] mentions that some researchers call their model "deep MF" but use it for supervised

tasks, or introduce a high degree of non-linearity inside it. This is for example the case of [82] that performs deep non-linear matrix completion, and [83] where the inner representations are obtained through a deep highly non-linear MF architecture. The term "deep MF" was also given to an iterative procedure to solve classical MF through a deep unfolding of the iterations over time in [84], though this has almost nothing in common with the deep MF as we have defined it in this paper.

Therefore, in this part, we mainly focus on works in the same spirit as Trigeorgis [11], aiming to extract hierarchical features in a non-supervised context, which has led to breakthrough results in several applications. This section is organized as follows. In Section 4.1, we present two simple showcase examples showing the ability of deep ONMF to extract hierarchical features. We choose deep ONMF because, as explained in Section 3.2.1, its factors are easily interpretable. A Matlab implementation is available[2] to allow the interested reader to explore these deep MF examples, and play with the different parameters. Then, in Section 4.2, we present several applications for which deep MF has been successfully used in the recent literature.

4.1. Two showcase examples

In this section, we detail two showcase examples on which deep MF reveals its inner workings. The first one is recommender systems (Section 4.1.1), and the second one is hyperspectral unmixing (HU) (Section 4.1.2). For both applications, we describe the results obtained with deep ONMF (see Section 3.2.1), which is a variant of deep MF particularly easy to interpret. Indeed, as each representation matrix H_l contains only one non-zero entry per column, each data point is associated with a single cluster.

4.1.1. Recommender systems

Recommender systems consist in predicting the ratings of users over unseen items based on historical ratings on seen items. In other words, given an incomplete rating matrix $X \in \mathbb{R}^{m \times n}$ of n users over m items (such as movies), the goal is to predict the missing entries. A standard approach to perform this task is by factorizing X as the product of two matrices $W \in \mathbb{R}^{m \times r}$ and $H \in \mathbb{R}^{r \times n}$ where W contains the ratings of r basis users over the m items and H represents the proportions in which each user behaves as the r basis users [85]. In the following, we describe how deep MF is able to extract hierarchical levels of basis users on a simple example.

Let us consider a matrix $X \in \mathbb{R}^{9 \times 15}$ such that $X(i, j)$ contains the rating of user j for movie i, and is between 1 (highly dislike) and 10 (highly like). Our goal is to apply deep MF on X and show the hierarchy of basis users extracted. The matrix X that will be considered throughout this synthetic example is the following:

$$X = \begin{pmatrix} 7 & 8 & 10 & 8 & 9 & 4 & 1 & 2 & 3 & 2 & 4 & 1 & 3 & 5 & 2 \\ 8 & 8 & 7 & 8 & 10 & 5 & 1 & 2 & 6 & 1 & 4 & 2 & 2 & 1 & 2 \\ 9 & 9 & 9 & 9 & 10 & 4 & 1 & 4 & 2 & 1 & 4 & 3 & 1 & 1 & 1 \\ 3 & 1 & 2 & 1 & 3 & 8 & 8 & 7 & 8 & 10 & 3 & 4 & 1 & 2 & 2 \\ 2 & 1 & 3 & 2 & 3 & 9 & 10 & 9 & 9 & 8 & 2 & 2 & 2 & 2 & 2 \\ 1 & 2 & 1 & 1 & 2 & 8 & 8 & 9 & 9 & 8 & 3 & 2 & 3 & 3 & 1 \\ 4 & 1 & 1 & 2 & 2 & 2 & 5 & 1 & 1 & 5 & 9 & 7 & 8 & 9 & 7 \\ 2 & 2 & 2 & 1 & 2 & 2 & 6 & 2 & 1 & 3 & 8 & 8 & 8 & 8 & 8 \\ 1 & 2 & 2 & 1 & 3 & 2 & 4 & 1 & 1 & 4 & 6 & 9 & 8 & 10 & 7 \end{pmatrix}.$$

Let us suppose, for example, that the first three movies (first three rows of X) are horror movies, the next three are comedy, and the last three are biopics. We observe that the first five users mostly enjoy horror movies, the next five ones comedies, and the last five biopics. Note that we do not consider missing data in this example, as our goal is to interpret the deep MF decomposition,
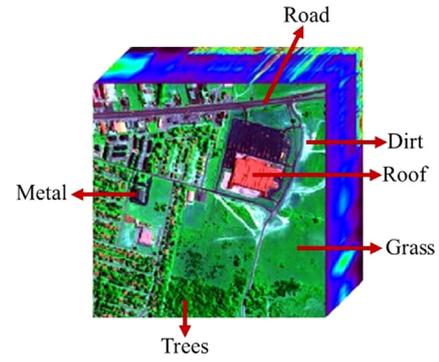

Fig. 4. The HYDICE Urban hyperspectral image.

rather than predicting missing entries. We apply deep ONMF on X with $L = 2$, $d_1 = 4$, $d_2 = 3$, that is, by computing the following decomposition:

$$\begin{aligned} X &\approx W_1 H_1, & H_1 H_1^T = I_4, & (W_1, H_1) \geq 0, \\ W_1 &\approx W_2 H_2, & H_2 H_2^T = I_3, & (W_2, H_2) \geq 0. \end{aligned} \tag{8}$$

To render the interpretation of the factors easier, we relax the orthogonality constraints by only imposing that $H_l H_l^T$ is diagonal, which does not change the hard clustering interpretation but simply allows each row of H_l's to have a norm different from 1. In counterpart, we normalize W_l's such that all the elements are between 0 and 10. This allows an easier comparison between the features extracted at each layer.

Let us interpret such a decomposition, layer by layer. At the first layer, we have $X \approx W_1 H_1$, and the matrices W_1 and H_1 are as follows (the values are rounded to one digit of accuracy):

$$W_1 = \begin{pmatrix} 9.1 & 1.7 & 3.4 & 3.8 \\ 8.9 & 1.1 & 4.9 & 2.7 \\ 10.0 & 1.1 & 3.7 & 2.5 \\ 2.2 & 10.0 & 8.6 & 3.0 \\ 2.4 & 10.0 & 10.0 & 2.5 \\ 1.5 & 8.9 & 9.6 & 3.0 \\ 2.2 & 5.5 & 1.5 & 10.0 \\ 2.0 & 5.0 & 1.8 & 9.9 \\ 2.0 & 4.4 & 1.5 & 10.0 \end{pmatrix},$$

$$H_1 = \begin{pmatrix} 0.88 & 0 & 0 & 0 \\ 0.87 & 0 & 0 & 0 \\ 0.92 & 0 & 0 & 0 \\ 0.87 & 0 & 0 & 0 \\ 1.05 & 0 & 0 & 0 \\ 0 & 0 & 0.92 & 0 \\ 0 & 0.92 & 0 & 0 \\ 0 & 0 & 0.85 & 0 \\ 0 & 0 & 0.92 & 0 \\ 0 & 0.88 & 0 & 0 \\ 0 & 0 & 0 & 0.82 \\ 0 & 0 & 0 & 0.80 \\ 0 & 0 & 0 & 0.79 \\ 0 & 0 & 0 & 0.89 \\ 0 & 0 & 0 & 0.71 \end{pmatrix}^T.$$

The columns of W_1 are themselves combinations of those of W_2, as $W_1 = W_2 H_2$ with H_2:

$$H_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1.02 & 0.98 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

2 https://bit.ly/3q4wRW8.

At the second layer, we have $X \approx W_2 \hat{H}_2$, with $\hat{H}_2 = H_2 H_1$. As $d_2 = 3$, we expect that each column of W_2 corresponds to the profile of a basis user liking only one category of movies, which is indeed the case as we obtain

$$W_2 = \begin{pmatrix} 9.1 & 2.7 & 3.8 \\ 8.9 & 3.3 & 2.7 \\ 10.0 & 2.6 & 2.5 \\ 2.2 & 9.1 & 3.0 \\ 2.4 & 10.0 & 2.5 \\ 1.5 & 9.3 & 3.0 \\ 2.2 & 3.2 & 10.0 \\ 2.0 & 3.1 & 9.9 \\ 2.0 & 2.7 & 10.0 \end{pmatrix}, \quad \hat{H}_2^T = \begin{pmatrix} 0.88 & 0 & 0 \\ 0.87 & 0 & 0 \\ 0.92 & 0 & 0 \\ 0.87 & 0 & 0 \\ 1.05 & 0 & 0 \\ 0 & 0.91 & 0 \\ 0 & 0.94 & 0 \\ 0 & 0.84 & 0 \\ 0 & 0.91 & 0 \\ 0 & 0.90 & 0 \\ 0 & 0 & 0.82 \\ 0 & 0 & 0.80 \\ 0 & 0 & 0.79 \\ 0 & 0 & 0.89 \\ 0 & 0 & 0.71 \end{pmatrix}.$$

The first column of W_2 corresponds to a basis user liking horror movies, the second comedies, and the last biopics.

The first and fourth columns of W_1 are identical to the first and third columns of W_2 respectively while the second and third column of W_1 bring more refined information. While the second column of W_2 only exhibits strong ratings for items 4 to 6 and low ones for the other items, the second and third columns of W_1 correspond to two different patterns such that the second basis user of layer 2 can be seen as the merging of two more informative basis users at layer 1. Both the second and third column of W_1 have high ratings over items 4 to 6, as for the second column of W_2 but the other ratings are different. Indeed, the second column of W_1 has intermediate ratings for biopics (items 7 to 9) but very poor ones over horror movies (items 1 to 3), and conversely for the third column. This level of granularity is not caught by the second layer of decomposition, which grasps the more general three main patterns that appear at first sight when looking at X. This justifies the benefit of using two layers of factorizations. To be more precise, as $d_1 > d_2$ in this example, deep MF first extracts 4 refined basis users and then gather two of them at layer 2 to model the more global structure of the data.

Let us mention that a single layer ONMF with $r = 4$ recovers matrices similar to W_1 and H_1, and similarly when $r = 3$ at the other layer. However, single layer factorizations do not exhibit any hierarchical relation between basis users.

4.1.2. Hyperspectral unmixing

Hyperspectral unmixing (HU) is a classical application of NMF, and many models taking into account various priors have been developed [86,87]. A hyperspectral image is composed of n pixels, each one characterized by the reflectance value (fraction of the light reflected) in m wavelengths, which is referred to as its spectral signature. Representing this image as a matrix $X \in \mathbb{R}^{m \times n}$ where each column is the spectral signature of a pixel, the purpose of HU is to identify the spectral signature of the r materials present in the image, that is, the columns of W, as well as their respective abundances in every pixel, that is, the columns of H. However, the precise number of materials is not always easy to determine as some materials have similar spectral signatures or are highly mixed. In this section, we consider the HYDICE Urban hyperspectral image which is an airborne image of a Walmart in Copperas Cove (Texas); see Fig. 4. It is made of $n = 307 \times 307$ pixels with $m = 162$ spectral bands. There are several versions of the ground truth depending on the number of materials considered [88].

Deep ONMF is able to extract the materials in a hierarchical manner, as illustrated on Fig. 5 which shows the abundance maps

H_l's, representing the proportions of a given material in the pixels. This solution was obtained by applying deep ONMF on the Urban image with $L = 4$, $d_1 = 7$, $d_2 = 6$, $d_3 = 4$, $d_4 = 2$. The first layer extracts several materials, namely two types of grass, trees, road, dirt, metal and roof. At the next layers, the materials are successively merged by two within a single cluster. At layer 2, the two clusters corresponding to road and metal, which have similar spectral signatures, are merged in a single cluster. At layer 3, the road/metal and dirt are merged to create a single cluster while the two kinds of grass are also merged in a single cluster. At layer 4, the road and roof are merged, while trees and grass are also merged in a cluster made of vegetation. Clearly, this example illustrates the ability of deep MF to extract materials in a hierarchical manner in hyperspectral images. Compared to traditional shallow extraction methods, deep MF brings an undeniable value in terms of interpretability.

Fig. 6 provides a comparison between the extracted spectral signatures at the third ($d_3 = 4$) layer with a ground truth from [89].

The signatures retrieved by deep MF are similar to the ground truth, which indicates that deep MF is able to extract meaningful features through several layers.

4.2. Real-world applications of deep MF

In this section, we review several applications of deep MF presented in the literature.

4.2.1. Recommender systems

We have already shown that deep MF extracts hierarchical information in the context of recommender systems in Section 4.1.1. Several models based on deep MF have been proposed in the recent literature.

Mongia et al. [90] use a projected gradient descent method to tackle deep MF with missing entries in the data matrix X. They refer to their model as deep latent factor model, and use it to infer missing entries while extracting several layers of explanatory factors, with a similar interpretation as in our showcase example in Section 4.1.

Xue et al. [91] derive a latent representation of both users and items through a so-called "deep factorization" though the model is different from the deep MF as defined in this paper. More precisely, based on a rating matrix $Y \in \mathbb{R}^{m \times n}$ containing both explicit ratings and non-preference implicit feedback (corresponding to a 0 value) of m users over n items, each row $Y(i, :)$ is mapped to a vector p_i such that $p_i = Y(i, :)W_1 \ldots W_L$ and similarly (with another set of matrices) for each column $Y(:, j)$, which is mapped into r_j. Then, a matrix $\hat{Y}$ is built, such that $\hat{Y}(i, j) = \frac{p_i^T r_j}{\|p_i\|\|r_j\|}$, and a cross-entropy based loss function between this similarity matrix and the original rating matrix Y is minimized on a training subset of the data. This deep approach reaches higher performance than state-of-the-art MF methods in terms of the ranking of suggested items, based on their predicted ratings.

Deep MF was also used in the context of recommender systems with implicit feedback when the ratings are not given as a numerical value but as a binary feedback (such as "like" or "dislike") [92]. For each user and each item, a vector containing both a representation of this implicit feedback and side information is provided. Then, deep MF is applied separately on the users and items to derive meaningful representations $H_L^{(u_i)}$'s and $H_L^{(v_j)}$'s for all users u_i's and items v_j's respectively, where the inner dimension d_L is the same for both factorizations. The rating r_{ij} of user i over item j is predicted as $r_{ij} = H_L^{(u_i)^T} H_L^{(v_j)} + G_{u_i} + G_{v_j}$ where G_{u_i} and G_{v_j} are obtained through maximum likelihood estimation and describe the specific influence of user and item, respectively. Using deep MF improves the root mean square error between the predicted and actual ratings compared to standard MF models on several benchmark data sets.
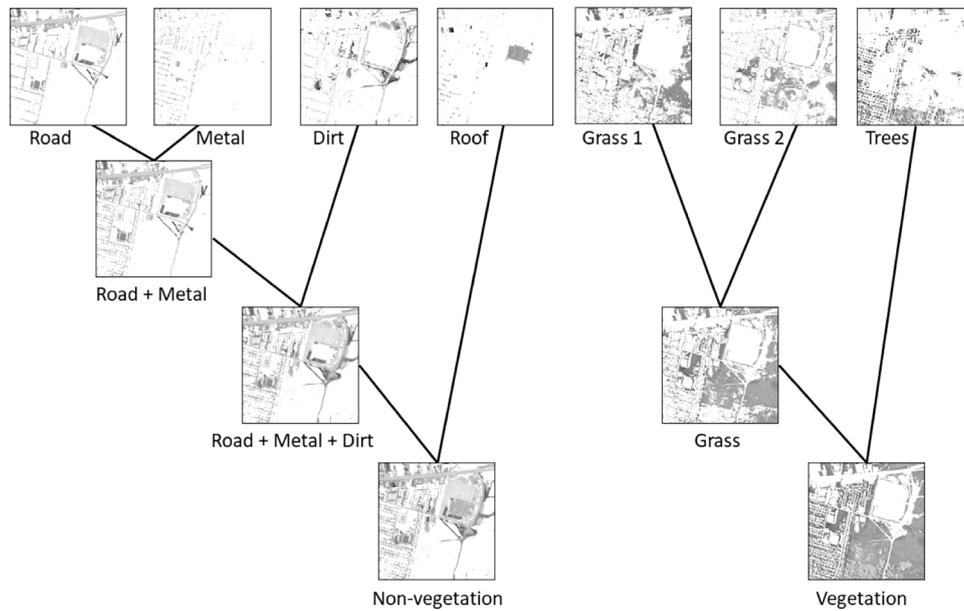
Fig. 5. Abundance maps hierarchically extracted by deep ONMF on the Urban data set. From top to bottom: first, second, third and fourth layer.
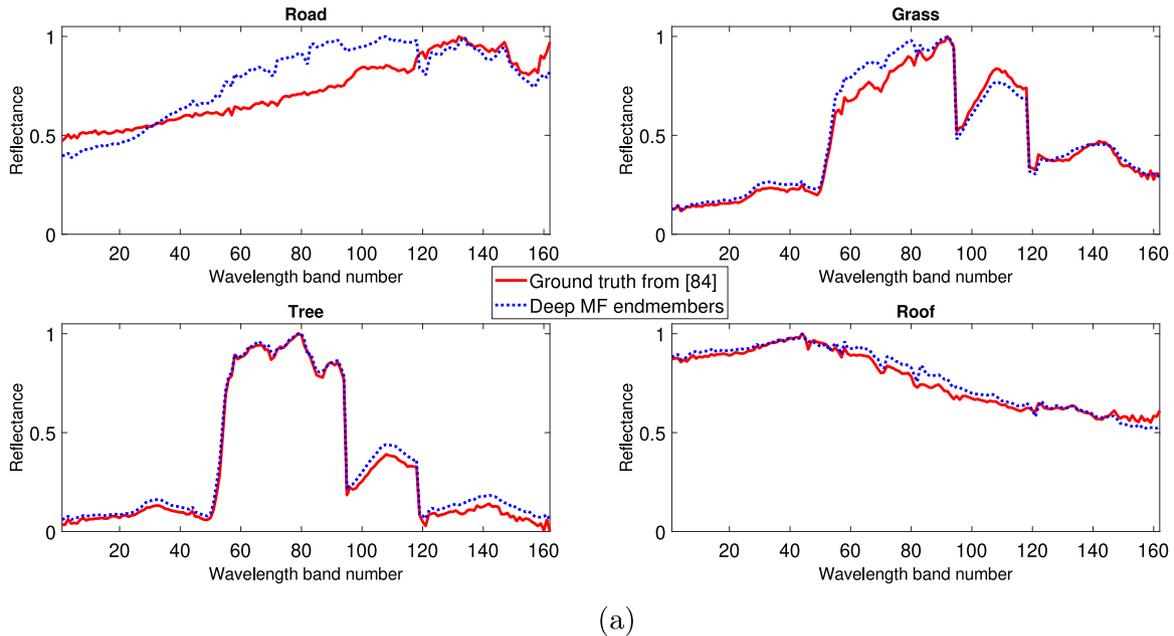

Fig. 6. Comparison between the endmembers extracted by deep ONMF at the third ($d_3 = 4$) layer and the "ground-truth" endmembers for the HYDICE Urban hyperspectral image.

4.2.2. Multi-view clustering

Multi-view clustering consists in clustering items for which the data are described by several views; for example, images described both by their pixels and textual tags, see [93] for a survey. In [94] and [95], each data matrix $X^{(v)}$ of each of the V views is deeply factorized and the last hidden representations $H_L^{(v)}$'s are constrained to be the same for all views. Cui et al. design the objective function as a weighted sum of the squared Frobenius norm of the residuals of each view, whose weights are also learned in [95]. A further refinement is proposed by Wei et al. who add a penalty term measuring the redundancy of the clusterings H_i and H_j of different layers i and j to the objective function [96]. More precisely, matrices $C^{(l)} = H_l^T H_l \in \mathbb{R}^{n \times n}$ for all l indicate if two data points are clustered identically or not at layer l. A penalty aiming at minimizing $\|C^{(i)} \odot C^{(j)}\|_1$ is added

to the objective for each pair of layers, to avoid redundancy, where $\odot$ denotes an element-wise multiplication.

A semi-supervised variant is considered by Xu et al. [97] with a graph Laplacian penalty aiming to both minimize the gap between the inner representation H_L of instances sharing the same label and maximize the gap between the inner representation H_L of instances belonging to different classes. Huang et al. [98] constrain the entries of H_L to be either 0 or 1. Finally, when the data are given through several views, such as images and documents, Xiong et al. [99] show that binary hashing codes derived through deep MF are able to find meaningful items with a binary code close to the one of a given query. In addition to the data fitting error, the loss function contains terms that aim at finding a unified latent representation H, and at minimizing the classification error of a linear classifier based on H. Moreover,

each entry of the unified code matrix H is constrained to be either $+1$ or -1.

4.2.3. Community detection

Community detection consists in identifying communities, that is, subsets of nodes that are highly connected, inside a given graph. While NMF is able to extract overlapping communities [100], deep MF allows to interpret the dynamics along which the nodes are progressively grouped. More precisely, taking as input of deep MF the adjacency matrix leads to the extraction of the membership coefficients of all nodes to d_l communities $H_l \in \mathbb{R}^{d_l \times n}$ at layer l with $d_L \leq d_{L-1} \leq \cdots \leq d_0 = n$ [101]. The interest of the deep architecture lies in the fact that nodes belonging to the same community gather closer to each other in terms of inner representations as the layers go deeper. In other words, deep MF allows to extract communities at different scales, smaller communities at the first layers are merged together in larger communities in the deeper layers as deep MF unfolds.

4.2.4. Hyperspectral unmixing

As illustrated in Section 4.1, deep MF can be used meaningfully for HU, extracting several layers of materials.

The early sequential multilayer NMF of Cichocki et al. [30] (2) was used, together with sparsity regularization, by Rajabi and Ghassemian [102]. Though the endmembers are estimated more accurately, no additional insight is given on the interpretability power of the model. Later, Tong et al. [103] use the deep model of Trigeorgis et al. [11] and show that it is efficient for the extraction of endmembers, though the interpretation of the successive inner representations is not emphasized. A similar approach takes into account an additional regularization [104]: on the one hand a sparsity constraint is considered on the abundance matrix H_L while on the other hand, a spatial regularization is applied through the total variation minimization (TVM). In a nutshell, TVM [105] is a well-known regularization which consists in computing the differences between the abundances of each pair of adjacent pixels and minimizing their sum to reduce the noise and get a smooth abundance map. A deep purely sequential model of archetypal analysis (deep AA), similar to the one developed by [58] (see Section 3.2), was also used for HU in [106].

4.2.5. Synthetic aperture radar (SAR)

SAR consists in analyzing the changes that appear on the surface of the Earth through high-resolution images. Given two images of the same location at different times, the goal of SAR change detection is to produce a binary map indicating the changed and unchanged pixels over the considered period.

Gao et al. [107] use deep semi-NMF to cluster the pixels in three categories: "unchanged", "changed" and "intermediate". Then, a more refined classification step accurately determines which pixels of the landscape have changed or not. Similarly, Li et al. propose to solve the SAR change detection problem with non-smooth deep MF [108]. The framework is again semi-supervised since a classification stage aims at reconstructing the label matrix based on the inner representation matrix H_L.

4.2.6. Audio processing

The audio source separation problem consists in extracting the frequential spectra of the sources contained in a sound recording as well as their respective activations over time. NMF has been shown to be efficient to solve this problem, when the matrix X is a time–frequency representation of the input data, for example the spectrogram obtained with a short-time Fourier transform (STFT); see [36] and the references therein.

Sharma et al. [109] use deep MF for speech recognition: given a matrix X of n frames, each one corresponding to a sequence of successive words, described by an m-dimensional vector corresponding to the well-known cepstral coefficients [110], a deep MF alternating sparse and dense layers factorizes X. The authors empirically notice that alternating sparse (l odd) and dense (l even) layers leads to more discriminative features, and the features used for the classification of the frames are obtained by applying a PCA on the concatenation of the inner representations H_l's corresponding to sparse layers.

Hsu et al. [111] apply deep MF on the spectrogram matrix of a set of spoken sentences to extract several layers of frequential basis features, and is better able to separate the speakers in a mixture than a simple one-layer NMF. Thakur et al. [112] used deep AA to extract sources based on the spectrograms of bioacoustics signals, with the dictionaries learnt at the first layers corresponding to archetypes on the convex hull of the data while deeper atoms being more in the center of the data. The classification accuracy obtained with a SVM based on the inner representations H_L's is higher than other state-of-the-art classification methods. An extension was proposed in [113] with a more sophisticated classification approach.

4.2.7. Perspectives

Deep MF does not seem to have been tested yet on several applications in which it has important potentialities. For example, in text mining tasks, it seems logical that hierarchical structures appear. For example, for NMF, given a word-by-document matrix X where the entry $X(i, j)$ is the number of times the word i appears in the document j, NMF allows to automatically extract topics as the columns of the basis matrix W, while H indicates which document discusses which topic [7]. In this context, deep NMF would be able to extract hierarchies of topics, from coarser to finer topics. For example, the first layers would extract general topics such as politics, geography and sports, while the deeper layers would refine these topics in sub-topics. For example, sports would be divided into tennis, soccer and golf, while soccer would contain results from different competitions. Note that NMF is known to be a simple topic model equivalent to latent semantic analysis/indexing (PLSA/PLSI) [114], and designing refined deep models would be of particular interest, similarly as done for NMF [115].

Though this survey focuses on linear deep MF, some applications would benefit from the introduction of non-linearities (see Section 5 for more details). For example, in hyperspectral unmixing, scattering and various interactions may justify the use of non-linear models [116]. Therefore, it would be interesting to investigate non-linear deep MF in view of the specific requirement of the applications.

5. Connections with neural networks

Several connections can be made between deep MF and deep learning. However, we restrict ourselves as much as possible to models aiming at extracting features from data in an unsupervised and interpretable way. Works embedding MF ideas in a neural network architecture, such as [117–120] are interesting but are further away from the focus of this survey.

Deep artificial neural networks [8] have been known for several years as one of the best classification paradigms. On Fig. 7, we have represented a standard neural network made of a succession of P fully-connected layers.[3] Each layer k, $k = 0, \ldots, P - 1$, is made of s_k units. Let us consider a data matrix $X \in \mathbb{R}^{m \times n}$ of n points in dimension $m = s_0$ and a binary label matrix $Y \in \mathbb{R}^{c \times n}$ indicating the membership of each data point $X(:, j)$ to each of

---

[3] We use an unusual naming of the parameters to avoid the confusion with the notation introduced for deep MF.
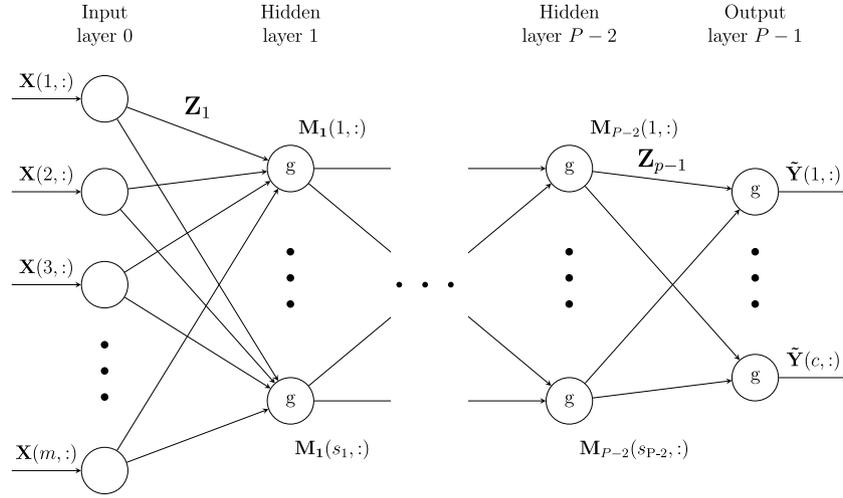
Fig. 7. Illustration of an artificial neural network.

the $c = s_{P-1}$ classes, that is, $Y(i, j) = 1$ if $X(:, j)$ belongs to the ith class. Given $X(:, j)$ for any j as input, the network produces as output a c-dimensional vector $\hat{Y}(:, j)$. Calling $Z_k \in \mathbb{R}^{s_{k-1} \times s_k}$, $k = 1, \ldots, P - 1$, the weights matrix between layer $k - 1$ and layer k, the first layer computes a vector $M_1(:, j) = g(Z_1 X(:, j))$ where g is a non-linear activation function applied element-wise. Then, any layer k, $k = 2, \ldots, P - 1$, computes $M_k(:, j) = Z_k M_{k-1}(:, j)$, with $M_{P-1}(:, j) = \hat{Y}(:, j)$. The goal of the neural network is to classify the data points $X(:, j)$'s at best, that is, optimize the Z_k's such that the prediction $\hat{Y}(:, j)$ is as close as possible to the ground-truth $Y(:, j)$ for all j. Overall, considering all the data points, the prediction matrix is given by $\hat{Y} = g(Z_{P-1}g(Z_{P-2} \ldots g(Z_1 X)))$.

Autoencoders [121] are particular neural networks where the output matrix does not correspond to a membership matrix but is identical to the input, that is, $Y = X$. Assuming that the number of layers P is odd, the purpose of an autoencoder is to extract a compressed representation M_Q of the input data at the central layer $Q = \frac{P-1}{2}$ through the encoder, and approximate as well as possible the initial data back after the decoder layers. Fig. 8(a) provides an illustration when the encoder and decoder are symmetric, that is, $s_k = s_{P-1-k}$ for all $k = 0, \ldots, P - 1$ and $Z_k = Z_{P-k}^T$ for all $k = 1, \ldots, P - 1$. This leads to the following approximation

$$X \approx \tilde{Y} = g(Z_1^T g(Z_2^T \ldots g(Z_Q^T M_Q))).$$

Let us number the layers in the reverse sense, that is, let us consider $l = P - 1 - k$ and $d_l = s_l$ for $l = 1, \ldots, L$. Let us also denote $W_l = Z_{P-l} = Z_l^T$ and $H_l = M_{P-1-l}$, with $L = Q$ such that the decoder performs the following decomposition:

$$X \approx \tilde{Y} = g(W_1 g(W_2 \cdots g(W_L H_L))). \tag{9}$$

When the activation function g is the identity, (9) becomes

$$X \approx \tilde{Y} = W_1 \ldots W_L H_L, \tag{10}$$

which corresponds to a so-called linear network. The decomposition performed by (10) is the same as deep MF but deep MF usually requires additional constraints, such as the non-negativity of some factors, to render the solution meaningful and interpretable (see Section 1).

A widely used activation function is the rectified linear unit, that is,

$$g(x) = ReLu(x) = \max(x, 0).$$

In this setting, each inner representation matrix $H_{l-1} = g(W_l H_l)$ for $l = 2, \ldots, L$ is imposed to be non-negative, as in the original deep MF of [11]. Though such a network is very similar

to deep MF, as shown on Fig. 8(b), the two models are not exactly equivalent since the representation matrix H_L of an autoencoder is learnt in a supervised way and is given by $H_L = g(W_L^T g(W_{L-1}^T \ldots g(W_1^T X)))$, which makes it close to deep archetypal analysis. In fact, autoencoders are mainly used in semi-supervised settings, for example to pre-train the networks for classification tasks, while deep MF mines unknown hierarchical features hidden in the data set. This connection suggests that the ranks of the factorization d_l's in deep MF should be chosen in a decreasing order, as for the central layer of an autoencoder, corresponding to H_L, is usually the smaller one.

Interestingly, the use of non-negativity constraints on the representation matrices within an autoencoder with a single layer in the encoder has shown to produce parts-based representations, as for NMF, while the overfitting was reduced [122]. Also, improvements were achieved in [123] using a deep structure promoting sparse activations H_l's, which is similar to deep sparse MF. However, only the pretraining stage is unsupervised and is similar to deep MF (though non-linear activations are used), while a supervised classification stage follows. This connection between neural networks and deep MF was also highlighted in [124] where the discriminative power of such a hierarchical model was observed on topic mining and audio source separation tasks.

Also inspired by deep learning non-linearities, Trigeorgis et al. [11] proposed to introduce a non-linear function, such as the sigmoid, at each layer of the model (2), that is, use $H_{l-1} = g(W_l H_l)$ for all l where $g(x) = \frac{1}{1+e^{-x}}$, which still provides a parts-based decomposition but at the cost of a possible weaker interpretability [125].

Similarly, deep AA is closely related to neural networks. An archetypal regularization based on an autoencoder was proposed by van Dijk et al. [126]. The latent representation H is learnt through a deep encoder performing a non-linear transformation of the input data and the addition of Gaussian noise to H enforces the basis vectors to be close to the data at the decoding layer. More precisely, the noise pushes the columns of H outside the unit simplex, which in turn enforces the columns of W to shrink in order to maintain a low reconstruction error. The strong connection between autoencoders and deep AA in the process of learning hierarchical features in image patches was also highlighted in [127].

6. Theoretical aspects of deep MF

Although numerous formulations, algorithms and application have been developed for deep MF, proper theoretical studies
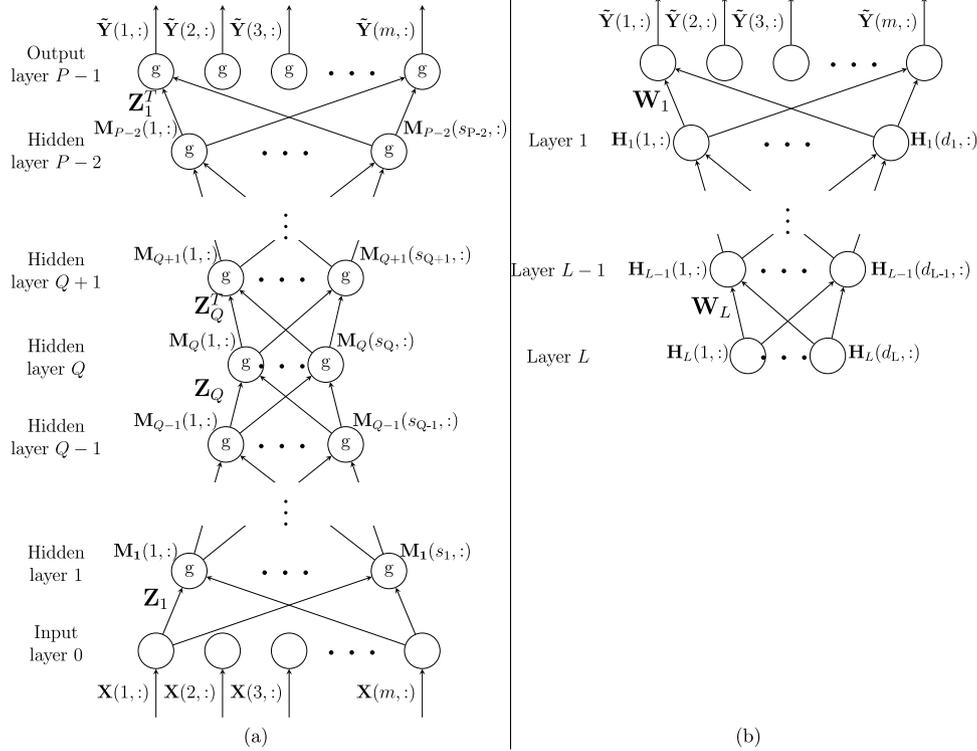
Fig. 8. Illustration of the similarity between (a) deep autoencoders and (b) deep MF.

remain scarce, apart from insights from the deep learning community working on linear networks. To the best of our knowledge, the main theoretical contributions so far are mostly the convergence of algorithms, and to a lesser extent identifiability.

6.1. Convergence issues

When the factors of deep MF are updated through a BCD (see Algorithm 4), the subproblems w.r.t. a single factor are convex, as for most CLRMA. Standard convergence results give conditions under which the iterates tend to stationary points, depending on whether the subproblems are solved through an exact algorithm or through an approximate framework, such as majorization minimization (MM). This encompasses most gradient descents used in practice. For example, when the global objective function is the least squares (5), using alternating projected gradient descent is guaranteed to converge to stationary points, because the subproblems are convex and Lipschitz smooth [128].

6.1.1. Convergence of first-order methods

The effect of the number of layers on the convergence of first-order methods applied on the problem (5) has not been much studied, to the best of our knowledge, but recent results have been obtained on deep linear networks (see Section 5).

Some of the theoretical results presented in the following are not directly related to the deep MF models described so far but rather concern networks aimed at supervised learning. However, we strongly believe that these insights coming from the deep linear networks community might be helpful to better understand deep MF and possibly open directions of future research. Let us mention a few important results. We refer the interested reader to the recent survey [129] for more details.

When the thinnest layer of a deep linear network is either the input or the output one, Laurent et al. [130] showed that deep linear networks with arbitrary convex differentiable loss produce local minima that are all global. In addition, when the input data is whitened (that is, the covariance matrix is the identity) and a proper initialization of all layers is chosen, Arora et al. [131] proved the linear convergence of gradient descent to a global minimum on such a network. This generalizes the results of [132] in which linear residual networks, where the weights of each layer are initialized to be the identity matrix and the inner ranks $d_0 = m, \ldots, d_l$ are the same, are considered. Indeed, the network architecture is more general and softer restrictions on the initialization are required. When the loss function is the squared error between Y and $\hat{Y}$, Arora et al. [133] provide an interesting result: If the weights W_l's are updated with gradient descent and if the initialization $W_{l+1}^{(0)^T} W_{l+1}^{(0)} = W_l^{(0)^T} W_l^{(0)}$ holds for all l, then there exists an equivalent update rule for the end-to-end matrix $W = W_1 \cdots W_L$ which can be seen as an acceleration of the gradient descent update as long as the learning step is sufficiently small. As the depth L grows, the effect is intensified which shows that over-parametrization, that is, considering several hidden layers, might accelerate the optimization process.

In the same spirit, when the network is restricted to be such that each hidden layer contains the same number d of units, but without considering specific assumptions on the input data nor the initialization, Du et al. [134] prove the linear convergence of gradient descent to a global optimum if the width of each layer is sufficient.

Finally, the convergence of gradient descent on a function of a product of matrices, especially the loss $\|Y - \prod_l W_l\|_F^2$, where $Y = -I_d$ and each W_l is a square matrix of size d was studied by Shamir et al. in [135]. Independent initialization of each layer is considered, that is either Xavier initialization (the entries of the W_l's are sampled from a zero-mean Gaussian distribution) or near-identity initialization (each $W_l = I + M$ with I the identity and M a matrix whose elements are sampled from a zero-mean Gaussian distribution). The smaller the variance of the initialization distribution is, the more likely it is that gradient

descent has exponential runtime w.r.t. the depth of the network, therefore advocating for shallow nets in this case. However, this result is rather empirical and the architecture of the network is particular, as each layer has the same number of units.

6.1.2. Low-rank structure

Arora et al. [81] demonstrate some advantages of unconstrained deep MF compared to standard shallow MF [136] in terms of regularization properties. Indeed, deep MF enhances an implicit tendency towards low-rank solutions. The problem considered is matrix completion, that is, impute missing entries in a given matrix $X \in \mathbb{R}^{m \times n}$. When the number of known entries is sufficiently large (this depends on the rank of X), factorizations of any depth admit solutions that tend to minimize the nuclear norm of the end-to-end matrix $W = W_1 \cdots W_L$, that is, to minimize the sum of the singular values of W. However, when there are fewer observed entries, the approximation tends to have a lower effective rank at the expense of a higher nuclear norm, especially when the depth increases. More interestingly, the evolution of the singular values of W obtained with gradient flow, that is, gradient descent with infinitesimally small learning rate, reveals that the solutions tend to have a few large singular values and many small ones, with a gap that intensifies with the depth of the factorization. This can be seen as an implicit regularization promoting low-rank solutions.

In summary, the recent literature gives evidence of the advantages of using a deep factorization, in terms of both speed of convergence of gradient descent to a global minimum and low-rank-*ness* of the factors. However, the settings described do not assume constraints on the factors of the decomposition, unlike most deep MF models. Extending these observations to the constrained case is an important direction of research.

6.2. Identifiability

Identifiability of exact deep MF is an important theoretical research question. It consists in establishing the conditions under which the factors $W_1, \ldots, W_L$ and H_L can be uniquely retrieved, up to trivial permutation and scaling. For various CLRMA problems, thorough conditions have been proposed; see for example [17] for NMF, [45] for SCA and DL, and [21] for simplex-structured MF, and the references therein.

Of course, any result for CLRMA can be extended to the corresponding deep MF model. Let us illustrate this with NMF. A necessary condition for an exact NMF $X = W_1 H_1$ with $W_1 \geq 0$ and $H_1 \geq 0$ to be unique, up to permutation and scaling, is that W_1^T and H_1 satisfy the so-called sufficiently scattered condition (SSC) [137]. Intuitively, the SSC requires that the rows of W_1 and the columns of H_1 are sufficiently well spread within the nonnegative orthant and have some degree of sparsity. Then, the two-layer $X = W_1 W_2 H_2$ is also unique, up to permutation and scaling, if $H_1 = W_2 H_2$ is unique, which is guaranteed if W_2^T and H_2 satisfy the SSC. Similar observations would apply for SCA, among others.

However, there are very few results tackling the identifiability of deep MF directly. As far as we know, the only attempt is by Malgouyres and Landsberg [138], in a very particular setting. The factorization $X \approx M_1(q_1)M_2(q_2) \ldots M_L(q_L)$ is considered where each matrix M_l is described through a small number S of parameters with $q_l \in \mathbb{R}^S$ for all l. A necessary and sufficient condition for identifiability in the noiseless case is provided as well as stability guarantees in the noisy case. However, these are quite abstract conditions involving advanced concepts such as the tensorial lifting property and the Segre embedding, and these conditions are difficult to check in practice. These results are further discussed in [139] where the conditions are extended to the case of convolutional linear networks.

Needless to say that the robustness to noise (also sometimes referred to as the stability) of deep MF models is also an important issue that has not been investigated yet. In fact, even in the matrix case, most known results apply to the unconstrained case or under orthogonality constraints [140]. For most other CLRMA problems, such results are rather scarce and difficult to derive.

7. Perspectives and conclusion

Deep MF is an emerging research topic, at the intersection of low-rank matrix approximations and deep learning. In this literature review, we presented multilayer and deep MF variants, which are being used successfully in an increasing number of applications, from recommender systems and hyperspectral unmixing to multi-view clustering and community detection.

Although many models and algorithms have been introduced for deep MF, the theoretical insights remain weak. In our opinion, this is a main direction of research that should be tackled by deep MF researchers. Interestingly, a similar trend was observed for neural networks: the theory started to be investigated thoroughly (and it is still a very active area of research) only after many models and algorithms were shown to perform well in practice.

Many perspectives have been presented throughout this survey, concerning the various aspects of deep MF. Let us summarize these perspectives:

- **Choice of the parameters**: The choice of the parameters, namely the inner ranks and the number of layers, has not been discussed much as it is mostly application dependent (see Section 3.3). Establishing proper guidelines to choose these parameters is a crucial issue.
- **Identifiability**: Identifiability of deep MF has not been investigated much, see Section 6.2. However, deriving conditions for deep MF to be unique could be particularly meaningful in some applications. In particular, it seems crucial to ensure that the features extracted at each layer are sufficiently different from each other and are meaningful. Most properties of deep MF have only been observed empirically, and the few theoretical results are only valid under specific assumptions. Hence diving into the theoretical aspects of the uniqueness of deep MF is promising direction of research.
- **Loss function**: Very few works have carefully investigated the choice of the loss function, see the discussion in Section 3.1. Besides, this influences the way the algorithms are designed and the way the quality of the respective extracted features is assessed.
- **Design new models and algorithms**: As evoked in Section 4.2, many MF models have not been extended to a deep setting yet. Moreover, efficient algorithms and initializations dedicated to deep MF are still lacking (see Section 3.3).
- **Links between deep MF and deep learning**: Though there exist obvious connections between deep neural networks and deep MF as described in Section 5, they do not seem to have been fully exploited yet. It is not clear whether it is possible to integrate advanced deep learning frameworks, such as convolutions, inside deep MF. Convolutional networks are known to extract several levels of visual features in image patches through highly non-linear operations [141]: is deep MF able of such performance in a linear and more interpretable way ? Convolutions were already integrated in shallow NMF under the so-called convolutive NMF framework, which looks for temporal patterns inside the data [142].
- **Applications**: Deep MF has not been applied yet on several important applications such as text mining. A recent work of Wang & Zhang [143] claims to apply deep MF for topic

modeling but the deepness only results from a non-linear transformation of the input data, while the factorization in itself remains shallow.

Moreover, in the applications described in Section 4.2, the interpretation of the features obtained at each layer is not always clear. This is also an important research issue. In particular, the original data points are usually clustered by applying k-means on the last inner representation matrix H_L. However, more robust techniques taking into account the information of the previous layers have not been used yet, to the best of our knowledge.

Let us also emphasize the lack of "ground-truth" baselines for deep MF: since deep MF is an unsupervised model in its essence, it is hard to guess precisely which features are expected at each layer and how to compare their quality in a fair way. As done by Trigeorgis et al. [11], it is possible to incorporate side information in the model, leading to semi-supervised models.

We believe deep MF could be a particularly useful framework as it combines the ability to extract hierarchical features, as deep learning models, with a high interpretability power, as low-rank matrix approximations. These advantages justify the necessity to maintain research efforts in deep MF, especially to improve the explainability of AI techniques.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

References

[1] M. Udell, C. Horn, R. Zadeh, S. Boyd, Generalized low rank models, Found. Trends Mach. Learn. 9 (1) (2016) 1–118.

[2] M. Udell, A. Townsend, Why are big data matrices approximately low rank? SIAM J. Math. Data Sci. 1 (1) (2019) 144–160.

[3] S. Wold, K. Esbensen, P. Geladi, Principal component analysis, Chemometr. Intell. Lab. Syst. 2 (1–3) (1987) 37–52.

[4] G.H. Golub, C. Reinsch, Singular value decomposition and least squares solutions, in: Linear Algebra, Springer, 1971, pp. 134–151.

[5] V. Papyan, Y. Romano, J. Sulam, M. Elad, Theoretical foundations of deep learning via sparse representations: A multilayer sparse model and its connection to convolutional neural networks, IEEE Signal Process. Mag. 35 (4) (2018) 72–89.

[6] P. Georgiev, F. Theis, A. Cichocki, Sparse component analysis and blind source separation of underdetermined mixtures, IEEE Trans. Neural Netw. 16 (4) (2005) 992–996.

[7] D.D. Lee, H.S. Seung, Learning the parts of objects by non-negative matrix factorization, Nature 401 (6755) (1999) 788–791.

[8] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436.

[9] G. Marcus, Deep learning: A critical appraisal, 2018, arXiv preprint arXiv:1801.00631.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Advances in Neural Information Processing Systems, 2014, pp. 2672–2680.

[11] G. Trigeorgis, K. Bousmalis, S. Zafeiriou, B. Schuller, A deep matrix factorization method for learning attribute representations, IEEE Trans. Pattern Anal. Mach. Intell. 39 (3) (2016) 417–429.

[12] N. Gillis, S.A. Vavasis, Fast and robust recursive algorithms for separable nonnegative matrix factorization, IEEE Trans. Pattern Anal. Mach. Intell. 36 (4) (2013) 698–714.

[13] A.M.S. Ang, N. Gillis, Accelerating nonnegative matrix factorization algorithms using extrapolation, Neural Comput. 31 (2) (2019) 417–439.

[14] C. Févotte, J. Idier, Algorithms for nonnegative matrix factorization with the β-divergence, Neural Comput. 23 (9) (2011) 2421–2456.

[15] Y.-X. Wang, Y.-J. Zhang, Nonnegative matrix factorization: A comprehensive review, IEEE Trans. Knowl. Data Eng. 25 (6) (2012) 1336–1353.

[16] J. Kim, Y. He, H. Park, Algorithms for nonnegative matrix and tensor factorizations: A unified view based on block coordinate descent framework, J. Global Optim. 58 (2) (2014) 285–319.

[17] X. Fu, K. Huang, N.D. Sidiropoulos, W. Ma, Nonnegative matrix factorization for signal and data analytics: Identifiability, algorithms, and applications., IEEE Signal Process. Mag. 36 (2) (2019) 59–80.

[18] N. Gillis, The why and how of nonnegative matrix factorization, Regul. Optim. Kernels Support Vector Mach. 12 (257) (2014).

[19] N. Gillis, Introduction to nonnegative matrix factorization, SIAG/OPT View. News 25 (1) (2017) 7–16.

[20] A. Cichocki, R. Zdunek, A.H. Phan, S. Amari, Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-Way Data Analysis and Blind Source Separation, John Wiley & Sons, 2009.

[21] M. Abdolali, N. Gillis, Simplex-structured matrix factorization: Sparsity-based identifiability and provably correct algorithms, 2020, arXiv preprint arXiv:2007.11446.

[22] L. Miao, H. Qi, Endmember extraction from highly mixed data using minimum volume constrained nonnegative matrix factorization, IEEE Trans. Geosci. Remote Sens. 45 (3) (2007) 765–777.

[23] M.A. Ang, N. Gillis, Volume regularized non-negative matrix factorizations, in: 2018 9th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS), IEEE, 2018, pp. 1–5.

[24] X. Fu, K. Huang, N.D. Sidiropoulos, W. Ma, Nonnegative matrix factorization for signal and data analytics: Identifiability, algorithms, and applications, IEEE Signal Process. Mag. 36 (2) (2019) 59–80.

[25] P.O. Hoyer, Non-negative sparse coding, in: Neural Networks for Signal Processing, 2002. Proceedings of the 2002 12th IEEE Workshop on, IEEE, 2002, pp. 557–565.

[26] M. Mørup, L.K. Hansen, Archetypal analysis for machine learning and data mining, Neurocomputing 80 (2012) 54–63.

[27] P. De Handschutter, N. Gillis, A. Vandaele, X. Siebert, Near-convex archetypal analysis, IEEE Signal Process. Lett. 27 (2019) 81–85.

[28] H. Javadi, A. Montanari, Nonnegative matrix factorization via archetypal analysis, J. Amer. Statist. Assoc. (2019) 1–22.

[29] S.A. Vavasis, On the complexity of nonnegative matrix factorization, SIAM J. Opt. 20 (3) (2009) 1364–1377.

[30] A. Cichocki, R. Zdunek, Multilayer nonnegative matrix factorisation, Electron. Lett. 42 (16) (2006) 947–948.

[31] A. Cichocki, R. Zdunek, Multilayer nonnegative matrix factorization using projected gradient approaches, Int. J. Neural Syst. 17 (06) (2007) 431–446.

[32] G. Trigeorgis, K. Bousmalis, S. Zafeiriou, B. Schuller, A deep semi-NMF model for learning hidden representations, in: International Conference on Machine Learning, 2014, pp. 1692–1700.

[33] C.H. Ding, T. Li, M.I. Jordan, Convex and semi-nonnegative matrix factorizations, IEEE Trans. Pattern Anal. Mach. Intell. 32 (1) (2010) 45–55.

[34] J. Yu, G. Zhou, A. Cichocki, S. Xie, Learning the hierarchical parts of objects by deep non-smooth nonnegative matrix factorization, 2018, arXiv preprint arXiv:1803.07226.

[35] O. Dikmen, Z. Yang, E. Oja, Learning the information divergence, IEEE Trans. Pattern Anal. Mach. Intell. 37 (7) (2014) 1442–1454.

[36] C. Févotte, N. Bertin, J.-L. Durrieu, Nonnegative matrix factorization with the Itakura-Saito divergence: With application to music analysis, Neural Comput. 21 (3) (2009) 793–830.

[37] V. Leplat, N. Gillis, A.M.S. Ang, Blind audio source separation with minimum-volume beta-divergence NMF, IEEE Trans. Signal Process. (2020) 3400–3410.

[38] C.H. Ding, T. Li, W. Peng, H. Park, Orthogonal nonnegative matrix t-factorizations for clustering, in: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2006, pp. 126–135.

[39] F. Pompili, N. Gillis, P.-A. Absil, F. Glineur, Two algorithms for orthogonal nonnegative matrix factorization with application to clustering, Neurocomputing 141 (2014) 15–25.

[40] B. Li, G. Zhou, A. Cichocki, Two efficient algorithms for approximately orthogonal nonnegative matrix factorization, IEEE Signal Process. Lett. 22 (7) (2014) 843–846.

[41] B. Lyu, K. Xie, W. Sun, A deep orthogonal non-negative matrix factorization method for learning attribute representations, in: International Conference on Neural Information Processing, Springer, 2017, pp. 443–452.

[42] Y. Qiu, G. Zhou, K. Xie, Deep approximately orthogonal nonnegative matrix factorization for clustering, 2017, arXiv preprint arXiv:1711.07437.

[43] J. Eggert, E. Korner, Sparse coding and NMF, in: 2004 IEEE International Joint Conference on Neural Networks, 4, IEEE, 2004, pp. 2529–2533.

[44] J. Kim, H. Park, Sparse Nonnegative Matrix Factorization for Clustering, echnical Report, Georgia Institute of Technology, 2008.

[45] R. Gribonval, R. Jenatton, F. Bach, Sparse and spurious: dictionary learning with noise and outliers, IEEE Trans. Inform. Theory 61 (11) (2015) 6298–6319.

[46] J.E. Cohen, N. Gillis, Nonnegative low-rank sparse component analysis, in: ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2019, pp. 8226–8230.

[47] Z. Guo, S. Zhang, Sparse deep nonnegative matrix factorization, Big Data Min. Anal. 3 (1) (2019) 13–28.

[48] A. Beck, M. Teboulle, A fast iterative shrinkage-thresholding algorithm for linear inverse problems, SIAM J. Imaging Sci. 2 (1) (2009) 183–202.

[49] N. Gillis, Sparse and unique nonnegative matrix factorization through data preprocessing, J. Mach. Learn. Res. 13 (Nov) (2012) 3349–3386.

[50] S. Lyu, X. Wang, On algorithms for sparse multi-factor NMF, in: Advances in Neural Information Processing Systems, 2013, pp. 602–610.

[51] R. Peharz, F. Pernkopf, Sparse nonnegative matrix factorization with l_0-constraints, Neurocomputing 80 (2012) 38–46.

[52] Y. Qian, S. Jia, J. Zhou, A. Robles-Kelly, Hyperspectral unmixing via $l_{1/2}$ sparsity-constrained nonnegative matrix factorization, IEEE Trans. Geosci. Remote Sens. 49 (11) (2011) 4282–4297.

[53] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.

[54] Z. He, J. Liu, C. Liu, Y. Wang, A. Yin, Y. Huang, Dropout non-negative matrix factorization, Knowl. Inf. Syst. 60 (2) (2019) 781–806.

[55] J. Cavazza, P. Morerio, B. Haeffele, C. Lane, V. Murino, R. Vidal, Dropout as a low-rank regularizer for matrix factorization, in: International Conference on Artificial Intelligence and Statistics, 2018, pp. 435–444.

[56] A. Pascual-Montano, J.M. Carazo, K. Kochi, D. Lehmann, R.D. Pascual-Marqui, Nonsmooth nonnegative matrix factorization (nsNMF), IEEE Trans. Pattern Anal. Mach. Intell. 28 (3) (2006) 403–415.

[57] H.A. Song, S.-Y. Lee, Hierarchical representation using NMF, in: International Conference on Neural Information Processing, Springer, 2013, pp. 466–473.

[58] P. Sharma, V. Abrol, A. Thakur, ASE: Acoustic scene embedding using deep archetypal analysis and GMM, in: Interspeech, 2018, pp. 3299–3303.

[59] S.M. Keller, M. Samarin, M. Wieser, V. Roth, Deep archetypal analysis, in: German Conference on Pattern Recognition, Springer, 2019, pp. 171–185.

[60] A. Alemi, I. Fischer, J. Dillon, K. Murphy, Deep variational information bottleneck, in: ICLR, 2017.

[61] X. Li, C. Zhao, Z. Shu, Q. Wang, Multilayer concept factorization for data representation, in: 2015 10th International Conference on Computer Science & Education (ICCSE), IEEE, 2015, pp. 486–491.

[62] Y. Zhang, Z. Zhang, Z. Zhang, M. Zhao, L. Zhang, Z. Zha, M. Wang, Deep self-representative concept factorization network for representation learning, in: Proceedings of the 2020 SIAM International Conference on Data Mining, SIAM, 2020, pp. 361–369.

[63] Z. Zhang, Y. Zhang, M. Xu, L. Zhang, Y. Yang, S. Yan, A survey on concept factorization: From shallow to deep representation learning, Inf. Process. Manage. 58 (3) (2021) 102534.

[64] Y. Meng, R. Shang, F. Shang, L. Jiao, S. Yang, R. Stolkin, Semi-supervised graph regularized deep NMF with bi-orthogonal constraints for data representation, IEEE Trans. Neural Netw. Learn. Syst. (2019).

[65] N.D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E.E. Papalexakis, C. Faloutsos, Tensor decomposition for signal processing and machine learning, IEEE Trans. Signal Process. 65 (13) (2017) 3551–3582.

[66] X. Bi, A. Qu, X. Shen, Multilayer tensor factorization with applications to recommender systems, Ann. Statist. 46 (6B) (2018) 3308–3333.

[67] J. Casebeer, M. Colomb, P. Smaragdis, Deep tensor factorization for spatially-aware scene decomposition, in: 2019 IEEE Workshop on Applications of Signal Processing To Audio and Acoustics (WASPAA), IEEE, 2019, pp. 180–184.

[68] P. Smaragdis, S. Venkataramani, A neural network alternative to non-negative audio models, in: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2017, pp. 86–90.

[69] C. Jia, M. Shao, Y. Fu, Sparse canonical temporal alignment with deep tensor decomposition for action recognition, IEEE Trans. Image Process. 26 (2) (2016) 738–750.

[70] S. Oymak, M. Soltanolkotabi, End-to-end learning of a convolutional neural network via deep tensor decomposition, 2018, arXiv preprint arXiv:1805.06523.

[71] I. Domanov, L.D. Lathauwer, Generic uniqueness conditions for the canonical polyadic decomposition and INDSCAL, SIAM J. Matrix Anal. Appl. 36 (4) (2015) 1567–1589.

[72] S. Ravishankar, Y. Bresler, Learning sparsifying transforms, IEEE Trans. Signal Process. 61 (5) (2012) 1072–1086.

[73] J. Maggu, A. Majumdar, Unsupervised deep transform learning, in: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2018, pp. 6782–6786.

[74] N. Gillis, Successive nonnegative projection algorithm for robust non-negative blind source separation, SIAM J. Imaging Sci. 7 (2) (2014) 1420–1450.

[75] D.D. Lee, H.S. Seung, Algorithms for non-negative matrix factorization, in: Advances in Neural Information Processing Systems, 2001, pp. 556–562.

[76] J. Ahn, S. Choi, J. Oh, A multiplicative up-propagation algorithm, in: Proceedings of the Twenty-First International Conference on Machine Learning, ACM, 2004, p. 3.

[77] C. Lin, Projected gradient methods for nonnegative matrix factorization, Neural Comput. 19 (10) (2007) 2756–2779.

[78] Y.E. Nesterov, A method for solving the convex programming problem with convergence rate O(1/k^2), in: Dokl. Akad. Nauk Sssr, 269, 1983, pp. 543–547.

[79] K. Huang, N.D. Sidiropoulos, A.P. Liavas, A flexible and efficient algorithmic framework for constrained matrix and tensor factorization, IEEE Trans. Signal Process. 64 (19) (2016) 5052–5065.

[80] Y. Zhou, L. Xu, A deep structure-enforced nonnegative matrix factorization for data representation, in: Chinese Conference on Pattern Recognition and Computer Vision (PRCV), Springer, 2018, pp. 340–350.

[81] S. Arora, N. Cohen, W. Hu, Y. Luo, Implicit regularization in deep matrix factorization, in: Advances in Neural Information Processing Systems, 2019, pp. 7411–7422.

[82] J. Fan, J. Cheng, Matrix completion by deep matrix factorization, Neural Netw. 98 (2018) 34–41.

[83] Q. Wang, M. Sun, L. Zhan, P. Thompson, S. Ji, J. Zhou, Multi-modality disease modeling via collective deep matrix factorization, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017, pp. 1155–1164.

[84] J. Le Roux, J.R. Hershey, F. Weninger, Deep NMF for speech separation, in: 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2015, pp. 66–70.

[85] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (8) (2009) 30–37.

[86] J.M. Bioucas-Dias, A. Plaza, N. Dobigeon, M. Parente, Q. Du, P. Gader, J. Chanussot, Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches, IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 5 (2) (2012) 354–379.

[87] W.-K. Ma, J.M. Bioucas-Dias, T.-H. Chan, N. Gillis, P. Gader, A.J. Plaza, A. Ambikapathi, C.-Y. Chi, A signal processing perspective on hyperspectral unmixing: Insights from remote sensing, IEEE Signal Process. Mag. 31 (1) (2013) 67–81.

[88] F. Zhu, Spectral unmixing datasets with ground truths, 2017, arXiv preprint arXiv:1708.05125.

[89] Data - rslab, (Accessed on 09/09/2020), https://rslab.ut.ac.ir/data.

[90] A. Mongia, N. Jhamb, E. Chouzenoux, A. Majumdar, Deep latent factor model for collaborative filtering, Signal Process. 169 (2020) 107366.

[91] H. Xue, X. Dai, J. Zhang, S. Huang, J. Chen, Deep matrix factorization models for recommender systems, in: IJCAI, 2017, pp. 3203–3209.

[92] B. Yi, X. Shen, H. Liu, Z. Zhang, W. Zhang, S. Liu, N. Xiong, Deep matrix factorization with implicit feedback embedding for recommendation system, IEEE Trans. Ind. Inf. (2019).

[93] Y. Yang, H. Wang, Multi-view clustering: A survey, Big Data Min. Anal. 1 (2) (2018) 83–107.

[94] H. Zhao, Z. Ding, Y. Fu, Multi-view clustering via deep matrix factorization, in: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, 2017, pp. 2921–2927.

[95] B. Cui, H. Yu, T. Zhang, S. Li, Self-weighted multi-view clustering with deep matrix factorization, in: Asian Conference on Machine Learning, 2019, pp. 567–582.

[96] S. Wei, J. Wang, G. Yu, C. Domeniconi, X. Zhang, Multi-view multiple clusterings using deep matrix factorization., in: AAAI, 2020, pp. 6348–6355.

[97] C. Xu, Z. Guan, W. Zhao, Y. Niu, Q. Wang, Z. Wang, Deep multi-view concept learning., in: IJCAI, 2018, pp. 2898–2904.

[98] S. Huang, Z. Kang, Z. Xu, Auto-weighted multi-view clustering via deep matrix decomposition, Pattern Recognit. 97 (2020) 107015.

[99] Y. Xiong, Y. Xu, X. Shu, Cross-view hashing via supervised deep discrete matrix factorization, Pattern Recognit. 103 (2020) 107270.

[100] J. Yang, J. Leskovec, Overlapping community detection at scale: a non-negative matrix factorization approach, in: Proceedings of the Sixth ACM International Conference on Web Search and Data Mining, 2013, pp. 587–596.

[101] F. Ye, C. Chen, Z. Zheng, Deep autoencoder-like nonnegative matrix factorization for community detection, in: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, ACM, 2018, pp. 1393–1402.

[102] R. Rajabi, H. Ghassemian, Spectral unmixing of hyperspectral imagery using multilayer NMF, IEEE Geosci. Remote Sens. Lett. 12 (1) (2014) 38–42.
[103] L. Tong, J. Yu, C. Xiao, B. Qian, Hyperspectral unmixing via deep matrix factorization, Int. J. Wavelets Multiresolut. Inf. Process. 15 (06) (2017) 1750058.
[104] X. Feng, H. Li, J. Li, Q. Du, A. Plaza, W.J. Emery, Hyperspectral unmixing using sparsity-constrained deep nonnegative matrix factorization with total variation, IEEE Trans. Geosci. Remote Sens. 56 (10) (2018) 6245–6257.
[105] L.I. Rudin, S. Osher, E. Fatemi, Nonlinear total variation based noise removal algorithms, Physica D 60 (1–4) (1992) 259–268.
[106] G. Zhao, C. Zhao, X. Jia, Multilayer unmixing for hyperspectral imagery with fast kernel archetypal analysis, IEEE Geosci. Remote Sens. Lett. 13 (10) (2016) 1532–1536.
[107] F. Gao, X. Liu, J. Dong, G. Zhong, M. Jian, Change detection in SAR images based on deep semi-NMF and SVD networks, Remote Sens. 9 (5) (2017) 435.
[108] H. Li, G. Yang, W. Yang, Q. Du, W.J. Emery, Deep nonsmooth nonnegative matrix factorization network factorization network with semi-supervised learning for SAR image change detection, ISPRS J. Photogramm. Remote Sens. 160 (2020) 167–179.
[109] P. Sharma, V. Abrol, A.K. Sao, Deep sparse representation based features for speech recognition, IEEE/ACM Trans. Audio Speech Lang. Process. 25 (11) (2017) 2162–2175.
[110] S. Davis, P. Mermelstein, Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences, IEEE Trans. Acoust. Speech Signal Process. 28 (4) (1980) 357–366.
[111] C. Hsu, J. Chien, T. Chi, Layered nonnegative matrix factorization for speech separation, in: 16th Annual Conference of the International Speech Communication Association (Interspeech 2015), Vols 1-5, 2015, pp. 628–632.
[112] A. Thakur, V. Abrol, P. Sharma, P. Rajan, Deep convex representations: Feature representations for bioacoustics classification, in: Interspeech, 2018, pp. 2127–2131.
[113] A. Thakur, P. Rajan, Deep archetypal analysis based intermediate matching kernel for bioacoustic classification, IEEE J. Sel. Top. Sign. Proces. 13 (2) (2019) 298–309.
[114] C. Ding, T. Li, W. Peng, On the equivalence between non-negative matrix factorization and probabilistic latent semantic indexing, Comput. Statist. Data Anal. 52 (8) (2008) 3913–3927.
[115] S. Arora, R. Ge, Y. Halpern, D. Mimno, A. Moitra, D. Sontag, Y. Wu, M. Zhu, A practical algorithm for topic modeling with provable guarantees, in: International Conference on Machine Learning, 2013, pp. 280–288.
[116] N. Dobigeon, J. Tourneret, C. Richard, J.C.M. Bermudez, S. McLaughlin, A.O. Hero, Nonlinear unmixing of hyperspectral images: Models and algorithms, IEEE Signal Process. Mag. 31 (1) (2013) 82–94.
[117] T.N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, B. Ramabhadran, Low-rank matrix factorization for deep neural network training with high-dimensional output targets, in: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, 2013, pp. 6655–6659.
[118] Y. Zhang, E. Chuangsuwanich, J. Glass, Extracting deep neural network bottleneck features using low-rank matrix factorization, in: 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2014, pp. 185–189.
[119] T.G. Kang, K. Kwon, J.W. Shin, N.S. Kim, NMF-based target source separation using deep neural network, IEEE Signal Process. Lett. 22 (2) (2014) 229–233.
[120] S. Ozkan, B. Kaya, G.B. Akar, Endnet: Sparse autoencoder network for endmember extraction and hyperspectral unmixing, IEEE Trans. Geosci. Remote Sens. (99) (2018) 1–15.
[121] A. Ng, Sparse autoencoder, CS294A Lecture Notes 72 (2011) (2011) 1–19.
[122] A. Lemme, R.F. Reinhart, J.J. Steil, Online learning and generalization of parts-based image representations by non-negative sparse autoencoders, Neural Netw. 33 (2012) 194–203.
[123] E. Hosseini-Asl, J.M. Zurada, O. Nasraoui, Deep learning of part-based representation of data using sparse autoencoders with nonnegativity constraints, IEEE Trans. Neural Netw. Learn. Syst. 27 (12) (2016) 2486–2498.
[124] J. Flenner, B. Hunter, A deep non-negative matrix factorization neural network, Semant. Sch. (2017).
[125] S. Tariyal, A. Majumdar, R. Singh, M. Vatsa, Deep dictionary learning, IEEE Access 4 (2016) 10096–10109.
[126] D. van Dijk, D.B. Burkhardt, M. Amodio, A. Tong, G. Wolf, S. Krishnaswamy, Finding archetypal spaces using neural networks, in: 2019 IEEE International Conference on Big Data, IEEE, 2019, pp. 2634–2643.
[127] C. Bauckhage, K. Kersting, F. Hoppe, C. Thurau, Archetypal analysis as an autoencoder, in: Workshop New Challenges in Neural Computation, 2015, p. 8.
[128] M. Razaviyayn, M. Hong, Z.-Q. Luo, A unified convergence analysis of block successive minimization methods for nonsmooth optimization, SIAM J. Optim. 23 (2) (2013) 1126–1153.
[129] R. Sun, D. Li, S. Liang, T. Ding, R. Srikant, The global landscape of neural networks: An overview, IEEE Signal Process. Mag. 37 (5) (2020) 95–108.
[130] T. Laurent, J. Brecht, Deep linear networks with arbitrary loss: All local minima are global, in: International Conference on Machine Learning, PMLR, 2018, pp. 2902–2907.
[131] S. Arora, N. Golowich, N. Cohen, W. Hu, A convergence analysis of gradient descent for deep linear neural networks, in: 7th International Conference on Learning Representations, ICLR 2019, 2019.
[132] P.L. Bartlett, D.P. Helmbold, P.M. Long, Gradient descent with identity initialization efficiently learns positive-definite linear transformations by deep residual networks, Neural Comput. 31 (3) (2019) 477–502.
[133] S. Arora, N. Cohen, E. Hazan, On the optimization of deep networks: Implicit acceleration by overparameterization, in: International Conference on Machine Learning, 2018, pp. 244–253.
[134] S. Du, W. Hu, Width provably matters in optimization for deep linear neural networks, in: International Conference on Machine Learning, 2019, pp. 1655–1664.
[135] O. Shamir, Exponential convergence time of gradient descent for one-dimensional deep linear neural networks, in: Conference on Learning Theory, 2019, pp. 2691–2713.
[136] S. Gunasekar, B.E. Woodworth, S. Bhojanapalli, B. Neyshabur, N. Srebro, Implicit regularization in matrix factorization, in: Advances in Neural Information Processing Systems, 2017, pp. 6151–6159.
[137] K. Huang, N.D. Sidiropoulos, A. Swami, Non-negative matrix factorization revisited: Uniqueness and algorithm for symmetric decomposition, IEEE Trans. Signal Process. 62 (1) (2013) 211–224.
[138] F. Malgouyres, J. Landsberg, On the identifiability and stable recovery of deep/multi-layer structured matrix factorization, in: 2016 IEEE Information Theory Workshop (ITW), IEEE, 2016, pp. 315–319.
[139] F. Malgouyres, J. Landsberg, Multilinear compressive sensing and an application to convolutional linear networks, SIAM J. Math. Data Sci. 1 (3) (2019) 446–475.
[140] G.W. Stewart, J.-G. Sun, Matrix Perturbation Theory, Academic Press, New York, 1990.
[141] O. Seddati, S. Dupont, S. Mahmoudi, M. Parian, Towards good practices for image retrieval based on CNN features, in: Proceedings of the IEEE International Conference on Computer Vision Workshops, 2017, pp. 1246–1255.
[142] P. Smaragdis, Non-negative matrix factor deconvolution; extraction of multiple sound sources from monophonic inputs, in: International Conference on Independent Component Analysis and Signal Separation, Springer, 2004, pp. 494–499.
[143] J. Wang, X.-L. Zhang, Deep NMF topic modeling, 2021, arXiv preprint arXiv:2102.12998.