Neurocomputing 80 (2012) 38–46

Contents lists available at SciVerse ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Sparse nonnegative matrix factorization with ℓ^0-constraints

Robert Peharz *, Franz Pernkopf

Signal Processing and Speech Communication Laboratory, Graz University of Technology, Austria

ARTICLE INFO

Available online 11 November 2011

Keywords:
NMF
Sparse coding
Nonnegative least squares

ABSTRACT

Although nonnegative matrix factorization (NMF) favors a sparse and part-based representation of nonnegative data, there is no guarantee for this behavior. Several authors proposed NMF methods which enforce sparseness by constraining or penalizing the ℓ^1-norm of the factor matrices. On the other hand, little work has been done using a more natural sparseness measure, the ℓ^0-pseudo-norm. In this paper, we propose a framework for approximate NMF which constrains the ℓ^0-norm of the basis matrix, or the coefficient matrix, respectively. For this purpose, techniques for unconstrained NMF can be easily incorporated, such as multiplicative update rules, or the alternating nonnegative least-squares scheme. In experiments we demonstrate the benefits of our methods, which compare to, or outperform existing approaches.

1. Introduction

Nonnegative matrix factorization (NMF) aims to factorize a nonnegative matrix $\mathbf{X}$ into a product of nonnegative matrices $\mathbf{W}$ and $\mathbf{H}$. We can distinguish between *exact* NMF, i.e. $\mathbf{X}=\mathbf{WH}$, and *approximate* NMF, i.e. $\mathbf{X}\approx\mathbf{WH}$. For approximate NMF, which seems more relevant for practical applications, one needs to define a divergence measure between the data $\mathbf{X}$ and its reconstruction $\mathbf{WH}$, such as the Frobenius norm or the generalized Kullback–Leibler divergence [1]. Approaches using more generalized measures, such as Bregman divergences or the α-divergence, can be found in [2,3]. In this paper, we focus on approximate NMF using the Frobenius norm as objective. Let the dimensions of $\mathbf{X}$, $\mathbf{W}$ and $\mathbf{H}$ be $D\times N$, $D\times K$, and $K\times N$, respectively. When the columns of $\mathbf{X}$ are multidimensional measurements of some process, the columns of $\mathbf{W}$ gain the interpretation of basis vectors, while $\mathbf{H}$ contains the corresponding weights. The number of basis vectors (or inner approximation rank) K is typically assumed to be $K\ll\min(D,N)$. Hence, NMF is typically used as compressive technique.

Originally proposed by Paatero and Tapper under the term positive matrix factorization [4], NMF became widely known due to the work of Lee and Seung [5,1]. One reason for its popularity is that the multiplicative update rules proposed in [1] are easy to implement. Furthermore, since these algorithms rely only on matrix multiplication and element-wise multiplication, they are fast on systems with well-tuned linear algebra methods. The main reason for its popularity, however, is that NMF tends to return a sparse and part-based representation of its input data, which makes its application interesting in areas such as computer vision [5], speech and audio processing [6–9], document clustering [10], to name but a few. This naturally occurring sparseness gives NMF a special status compared to other matrix factorization methods such as principal/independent component analysis or k-means clustering.

However, sparsity in NMF occurs as a by-product due to nonnegativity constraints, rather than being a design objective of its own. Various authors proposed modified NMF algorithms which explicitly enforce sparseness. These methods usually penalize [11,12] or constrain [13] the ℓ^1-norm of $\mathbf{H}$ or $\mathbf{W}$, which is known to yield a sparse representation [14,15]. An explanation for the sparseness inducing nature of the ℓ^1-norm is that it can be interpreted as a convex relaxation of the ℓ^0-(pseudo)-norm, i.e. the number of non-zero entries in a vector. Indeed, the ℓ^0-norm is a more intuitive sparseness measure, which allows to specify a certain number of non-zero entries, while similar statements cannot be made via the ℓ^1-norm. Introducing the non-convex ℓ^0-norm as constraint function typically renders a problem NP-hard, requiring exhaustive combinatoric search. However, Vavasis [16] has shown that NMF is NP-hard per se,[1] so we have to accept (most probably) that any algorithm for NMF is suboptimal. Hence, a heuristic method for NMF with ℓ^0-constraints might be just as appropriate and efficient as ℓ^1-sparse NMF.

Little work is concerned with ℓ^0-sparse NMF. The K-SVD algorithm [17] aims to find an overcomplete dictionary for sparse

* Corresponding author.
E-mail address: robert.peharz@tugraz.at (R. Peharz).

[1] He actually has shown that the decision problem, whether an *exact* NMF of a certain rank exists or not, is NP-hard. An optimal algorithm for approximate NMF can be used to solve the decision problem. Hence NP-hardness follows also for approximate NMF.

0925-2312
doi:10.1016/j.neucom.2011.09.024

representation of a set of training signals $\mathbf{X}$. The algorithm minimizes the approximation error between data $\mathbf{X}$ and its reconstruction $\mathbf{WH}$, where ℓ^0-constraints are imposed on the columns of $\mathbf{H}$. The nonnegative version of K-SVD [18] additionally constrains all matrices to be nonnegative. Hence, nonnegative K-SVD can be interpreted as NMF with sparseness constraints on the columns of $\mathbf{H}$. Probabilistic sparse matrix factorization (PSMF) [19] is closely related to K-SVD, however, no nonnegative PSMF was proposed so far. Morup et al. [20] proposed an approximate NMF algorithm which constrains the ℓ^0-norm of the $\mathbf{H}$ columns, using a nonnegative version of least angle regression and selection [21]. For the $\mathbf{W}$ update, they used the normalization-invariant update rule described in [12]. In [22], a method for NMF was described, which penalizes a smoothed ℓ^0-norm of the matrix $\mathbf{H}$. Using this smooth approximation of the ℓ^0-norm, they were able to derive multiplicative update rules, similar as in [1]. More details about prior art concerned with ℓ^0-sparse NMF can be found in Section 2.4.

In this paper, we propose two generic strategies to compute an approximate NMF with ℓ^0-sparseness constraints on the columns of $\mathbf{H}$ or $\mathbf{W}$, respectively. For NMF with constraints on $\mathbf{H}$, the key challenge is to find a good approximate solution for the nonnegative sparse coding problem, which is NP-hard in general [23]. For sparse coding without nonnegativity constraints, a popular approximation algorithm is orthogonal matching pursuit (OMP) [24], due to its simplicity and theoretic guarantees [25,26]. Here, we show a close relation between OMP and the active set algorithm for nonnegative least squares (NNLS) [27], which, to the best of our knowledge, was overlooked in the literature so far. As a consequence, we propose a simple modification of NNLS, called sparse NNLS (sNNLS), which represents a natural integration of nonnegativity constraints in OMP. Furthermore, we propose an algorithm called reverse sparse NNLS (rsNNLS), which uses a reversed matching pursuit principle. This algorithm shows the best performance of all sparse coders in our experiments, and competes with or outperforms nonnegative basis pursuit (NNBP). Note that basis pursuit usually delivers better results than algorithms from the matching pursuit family, while requiring more computational resources [28,25]. For the second stage in our framework, which updates $\mathbf{W}$ and the non-zero coefficients in $\mathbf{H}$, we show that the standard multiplicative update rules [1] can be used without any modification. Also, we propose a sparseness maintaining active set algorithm for NNLS, which allows to apply an alternating least squares scheme for ℓ^0-sparse NMF.

Furthermore, we propose an algorithm for NMF with ℓ^0-constrained columns in $\mathbf{W}$. As far as we know, no method exists for this problem so far. The proposed algorithm follows a similar approach as in ℓ^1-constrained NMF [13], projecting the columns of $\mathbf{W}$ onto the closest vectors with desired sparseness after each update step. In experiments, the algorithm runs much faster than the ℓ^1-constrained method. Furthermore, the results of the proposed method are significantly sparser in terms of the ℓ^0-norm while achieving the same reconstruction quality.

Throughout the paper, we use the following notation. Upper-case boldface letter denote matrices. For sets we use Fraktur letters, e.g. $\mathcal{P}$. A lower-case boldface letter denotes a column vector. A lower-case boldface letter with a subscript index denotes a specific column of the matrix denoted by the same upper-case letter, e.g. $\mathbf{x}_i$ is the ith column of matrix $\mathbf{X}$. Lower-case letters with subscripts denote specific entries of a vector or a matrix, e.g. x_i is the ith entry of $\mathbf{x}$ and $x_{i,j}$ is the entry in the ith row and the jth column of $\mathbf{X}$. A matrix symbol subscripted with a set symbol denotes the submatrix consisting of the columns which are indexed by the elements of the set, e.g. $\mathbf{W}_{\mathcal{P}}$ is the sub-matrix of $\mathbf{W}$ containing the columns indexed by $\mathcal{P}$. Similarly, a vector subscripted by a set symbol is the sub-vector containing the entries indexed by the set. With $\|\cdot\|_p, p \geq 1$ we denote the ℓ^p-norm: $\|\mathbf{x}\|_p = (\sum_i |x_i|^p)^{1/p}$. Further, $\|\cdot\|_0$ denotes the ℓ^0-pseudo-norm, i.e. the number of non-zero entries in the argument. The Frobenius norm is defined as $\|\mathbf{X}\|_F = \sqrt{\sum_{i,j} |x_{i,j}|^2}$.

The paper is organized as follows. In Section 2 we discuss NMF techniques related to our work. We present our framework for ℓ^0-sparse NMF in Section 3. Experiments are presented in Section 4 and Section 5 concludes the paper.

2. Related work

Let us formalize NMF as the following optimization problem:

$$\begin{aligned} \underset{\mathbf{W},\mathbf{H}}{\text{minimize}} \quad & \|\mathbf{X}-\mathbf{WH}\|_F \\ \text{subject to} \quad & \mathbf{W} \geq 0, \\ & \mathbf{H} \geq 0, \end{aligned} \tag{1}$$

where $\geq$ denotes the element-wise greater-or-equal operator.

2.1. NMF via multiplicative updates

Lee and Seung [1] showed that $\|\mathbf{X}-\mathbf{WH}\|_F$ is nonincreasing under the multiplicative update rules

$$\mathbf{H} \leftarrow \mathbf{H} \otimes \frac{(\mathbf{W}^T\mathbf{X})}{(\mathbf{W}^T\mathbf{WH})} \tag{2}$$

and

$$\mathbf{W} \leftarrow \mathbf{W} \otimes \frac{(\mathbf{XH}^T)}{(\mathbf{WHH}^T)}, \tag{3}$$

where $\otimes$ and / denote element-wise multiplication and division, respectively. Obviously, these update rules preserve nonnegativity of $\mathbf{W}$ and $\mathbf{H}$, given that $\mathbf{X}$ is element-wise nonnegative.

2.2. NMF via alternating nonnegative least squares

Paatero and Tapper [4] originally suggested to solve (1) alternately for $\mathbf{W}$ and $\mathbf{H}$, i.e. to iterate the following two steps:

$$\mathbf{H} \leftarrow \arg\min_{\mathbf{H}} \|\mathbf{X}-\mathbf{WH}\|_F \quad \text{s.t. } \mathbf{H} \geq 0,$$

$$\mathbf{W} \leftarrow \arg\min_{\mathbf{W}} \|\mathbf{X}-\mathbf{WH}\|_F \quad \text{s.t. } \mathbf{W} \geq 0.$$

Note that $\|\mathbf{X}-\mathbf{WH}\|_F$ is convex in either $\mathbf{W}$ or $\mathbf{H}$, respectively, but non-convex in $\mathbf{W}$ and $\mathbf{H}$ jointly. An optimal solution for each subproblem is found by solving a nonnegatively constrained least-squares problem (NNLS) with multiple right hand sides (i.e. one for each column of $\mathbf{X}$). Consequently, this scheme is called alternating nonnegative least-squares (ANLS). For this purpose, we can use the well known active-set algorithm by Lawson and Hanson [27], which for convenience is shown in Algorithm 1. The symbol † denotes the pseudo-inverse.

Algorithm 1. Active-set NNLS [27].

$\mathcal{Z} = \{1,\ldots,K\}, \mathcal{P} = \emptyset$
$\mathbf{h} = \mathbf{0}$
$\mathbf{r} = \mathbf{x}-\mathbf{Wh}$
$\mathbf{a} = \mathbf{W}^T\mathbf{r}$
while $|\mathcal{Z}| > 0$ **and** $\exists\, i \in \mathcal{Z} : a_i > 0$ **do**
  $i^* = \arg\max \mathbf{a}$
  $\mathcal{Z} \leftarrow \mathcal{Z} \backslash i^*$
  $\mathcal{P} \leftarrow \mathcal{P} \cup i^*$
  $\mathbf{z}_{\mathcal{P}} = \mathbf{W}_{\mathcal{P}}^{\dagger}\mathbf{x}$
  $\mathbf{z}_{\mathcal{Z}} = 0$
  while $\exists\, j \in \mathcal{P} : z_j < 0$ **do**

12: $\alpha = \min_{k \in \mathcal{P}} h_k/(h_k - z_k)$
13: $\mathbf{h} \leftarrow \mathbf{h} + \alpha(\mathbf{z} - \mathbf{h})$
14: $\mathcal{Z} \leftarrow \{i | h_i = 0\}$
15: $\mathcal{P} \leftarrow \{i | h_i > 0\}$
16: $\mathbf{z}_{\mathcal{P}} = \mathbf{W}_{\mathcal{P}}^{\dagger}\mathbf{x}$
17: $\mathbf{z}_{\mathcal{Z}} = 0$
18: **end while**
19: $\mathbf{h} \leftarrow \mathbf{z}$
20: $\mathbf{r} = \mathbf{x} - \mathbf{W}\mathbf{h}$
21: $\mathbf{a} = \mathbf{W}^T\mathbf{r}$
22: **end while**

This algorithm solves NNLS for a single right hand side, i.e. it returns $\arg\min_{\mathbf{h}} \|\mathbf{x} - \mathbf{W}\mathbf{h}\|_2$, s.t. $\mathbf{h} \geq 0$, for an arbitrary vector $\mathbf{x}$. The *active* set $\mathcal{Z}$ and the *in-active* set $\mathcal{P}$ contain disjointly the indices of the $\mathbf{h}$ entries. Entries with indices in $\mathcal{Z}$ are held at zero (i.e. the nonnegativity constraints are *active*), while entries with indices in $\mathcal{P}$ have positive values. In the outer loop of Algorithm 1, indices are moved from $\mathcal{Z}$ to $\mathcal{P}$, until an optimal solution is obtained (cf. [27] for details). The inner loop (steps 11–18) corrects the tentative, possibly infeasible (i.e. negative) solution $\mathbf{z}$ to a feasible one. Note that i^* is guaranteed to be an element of $\mathcal{Z}$ in step 6, since the residual $\mathbf{x} - \mathbf{W}\mathbf{h}$ is orthogonal to every column in $\mathbf{W}_{\mathcal{P}}$, and hence $\mathbf{a}_{\mathcal{P}} \equiv 0$. Lawson and Hanson [27] showed that the outer loop has to terminate eventually, since the residual error strictly decreases in each iteration, which implies that no in-active set $\mathcal{P}$ is considered twice. Unfortunately, there is no polynomial runtime guarantee for NNLS. However, they report that the algorithm typically runs well-behaved.

For ANLS, one can apply Algorithm 1 to all columns of $\mathbf{X}$ independently, in order to solve for $\mathbf{H}$.[2] However, a more efficient variant of NNLS with multiple right hand sides was proposed in [29]. This algorithm executes Algorithm 1 quasi-parallel for all columns in $\mathbf{X}$, and solves step 16 (least-squares) jointly for all columns sharing the same in-active set $\mathcal{P}$. Kim and Park [30] applied this more efficient variant to ANLS-NMF. Alternatively, NNLS can be solved using numerical approaches, such as the projected gradient algorithm proposed in [31]. For the remainder of the paper, we use the notation $\mathbf{h} = \text{NNLS}(\mathbf{x}, \mathbf{W})$ to state that $\mathbf{x}$ is approximated by $\mathbf{W}\mathbf{h}$ in NNLS sense. Similarly, $\mathbf{H} = \text{NNLS}(\mathbf{X}, \mathbf{W})$ denotes the solution of an NNLS problem with multiple right hand sides.

2.3. Sparse NMF

Various extensions have been proposed in order to incorporate sparsity in NMF, where sparseness is typically measured via some function of the ℓ^1-norm. Hoyer [11] proposed an algorithm to minimize the objective $\|\mathbf{X} - \mathbf{W}\mathbf{H}\|_F^2 + \lambda \sum_{ij} |H_{ij}|$, which penalizes the ℓ^1-norm of the coefficient matrix $\mathbf{H}$. Eggert and Koerner [12] used the same objective, but proposed an alternative update which implicitly normalizes the columns of $\mathbf{W}$ to unit length. Furthermore, Hoyer [13] defined the following sparseness function for an arbitrary vector $\mathbf{x}$:

$$\text{sparseness}(\mathbf{x}) = \frac{\sqrt{D} - \|\mathbf{x}\|_1/\|\mathbf{x}\|_2}{\sqrt{D} - 1}, \tag{4}$$

where D is the dimensionality of $\mathbf{x}$. Indeed, sparseness($\mathbf{x}$) is 0, if all entries of $\mathbf{x}$ are non-zero and their absolute values are all equal, and 1 when only one entry is non-zero. For all other $\mathbf{x}$, the function smoothly interpolates between these extreme cases. Hoyer provided an NMF algorithm which constrains the sparseness of the columns of $\mathbf{W}$, the rows of $\mathbf{H}$, or both, to any desired sparseness value according to (4). There are further approaches which aim to achieve a part-based and sparse representation, such as local NMF [32] and non-smooth NMF [33].

[2] To solve for $\mathbf{W}$, one transposes $\mathbf{X}$ and $\mathbf{H}$ and executes the algorithm for each column of $\mathbf{X}^T$.

2.4. Prior art for ℓ^0-sparse NMF

As mentioned in the Introduction, relatively few approaches exist for ℓ^0-sparse NMF. The K-SVD algorithm [17] aims to $\text{minimize}_{\mathbf{W},\mathbf{H}} \|\mathbf{X} - \mathbf{W}\mathbf{H}\|_F$, subject to $\|\mathbf{h}_i\|_0 \leq L, \forall i$, where $L \in \mathbb{N}$ is the maximal number of non-zero coefficients per column of $\mathbf{H}$. Hence, the nonnegative version of K-SVD (NNK-SVD) [18] can be considered as an NMF algorithm with ℓ^0-sparseness constraints on the columns of $\mathbf{H}$. However, the sparse coding stage in nonnegative K-SVD is rather an ad hoc solution, using an approximate version of nonnegative basis pursuit [28]. For the $\mathbf{W}$ update stage, the K-SVD dictionary update is modified, by simply truncating negative values to zero after each iteration of an SVD approximation.

In [20], an algorithm for NMF with ℓ^0-sparseness constraints on the $\mathbf{H}$ columns was proposed. For the sparse coding stage, they used a nonnegative version of least angle regression and selection (LARS) [21], called NLARS. This algorithm returns a so-called solution path of the ℓ^1-regularized objective $\|\mathbf{x} - \mathbf{W}\mathbf{h}\|_F^2 + \lambda\|\mathbf{h}\|_1$ using an active-set algorithm, i.e. it returns several solution vectors $\mathbf{h}$ with varying λ. For a specific column $\mathbf{x}$ out of $\mathbf{X}$, one takes the solution $\mathbf{h}$ with desired ℓ^0-sparseness (when there are several such vectors, one selects the solution with smallest regularization parameter λ). Repeating this for each column of $\mathbf{X}$, one obtains a nonnegative coding matrix with ℓ^0-sparseness on its columns. To update $\mathbf{W}$, the authors used the self-normalizing multiplicative update rule described in [12].

In [22], the objective $\|\mathbf{x} - \mathbf{W}\mathbf{h}\|_F^2 + \alpha \sum_i \sum_j f_\sigma(h_{i,j})$ is considered, where $f_\sigma(h) = \exp(-h^2/2\sigma^2)$. For $\sigma \to 0$, the second term in the objective converges to α times the number of non-zero entries in $\mathbf{H}$. Contrary to the first two approaches, which *constrain* the ℓ^0-norm, this method calculates an NMF which *penalizes* the smoothed ℓ^0-norm, where penalization strength is controlled with the trade-off parameter α. Therefore, the latter approach proceeds similar as the NMF methods which penalize the ℓ^1-norm [11,12]. However, note that a trade-off parameter for a penalization term is generally not easy to choose, while ℓ^0-sparseness *constraints* have an immediate meaning.

3. Sparse NMF with ℓ^0-constraints

We now introduce our methods for NMF with ℓ^0-sparseness constraints on the columns of $\mathbf{W}$ and $\mathbf{H}$, respectively. Formally, we consider the problems

$$\begin{aligned} \underset{\mathbf{W},\mathbf{H}}{\text{minimize}} \quad & \|\mathbf{X} - \mathbf{W}\mathbf{H}\|_F \\ \text{subject to} \quad & \mathbf{W} \geq 0, \mathbf{H} \geq 0, \\ & \|\mathbf{h}_i\|_0 \leq L, \quad \forall i \end{aligned} \tag{5}$$

and

$$\begin{aligned} \underset{\mathbf{W},\mathbf{H}}{\text{minimize}} \quad & \|\mathbf{X} - \mathbf{W}\mathbf{H}\|_F \\ \text{subject to} \quad & \mathbf{W} \geq 0, \mathbf{H} \geq 0, \\ & \|\mathbf{w}_i\|_0 \leq L, \quad \forall i. \end{aligned} \tag{6}$$

We refer as NMFℓ^0-H and NMFℓ^0-W to problems (5) and (6), respectively. Parameter $L \in \mathbb{N}$ is the maximal allowed number of non-zero entries in $\mathbf{w}_i$ or $\mathbf{h}_i$.

For NMFℓ^0-H, the sparseness constraints imply that each column in $\mathbf{X}$ is represented by a conical combination of maximal

L nonnegative basis vectors. When we interpret the columns of $\mathbf{W}$ as *features*, this means that each data sample is represented by maximal L features, where typically $L \ll K$. Nevertheless, if the reconstruction error is small, this implies that the extracted features are important to some extend. Furthermore, as noted in the Introduction, for unconstrained NMF it is typically assumed that $K \ll \min(D,N)$. We do not have to make this restriction for NMFℓ^0-H, and can even choose $K > D$, i.e. we can allow an overcomplete basis matrix $\mathbf{W}$ (however, we require $K < N$).

For NMFℓ^0-W, the sparseness constraints enforce basis vectors with limited support. If for example the columns of $\mathbf{X}$ contain image data, sparseness constraints on $\mathbf{W}$ encourage a part-based representation.

NMF algorithms usually proceed in a two stage iterative manner, i.e. they alternately update $\mathbf{W}$ and $\mathbf{H}$ (cf. Section 2). We apply the same principle to NMFℓ^0-H and NMFℓ^0-W, where we take care that the sparseness constraints are maintained.

3.1. NMFℓ^0-H

A generic alternating update scheme for NMFℓ^0-H is illustrated in Algorithm 2. In the first stage, the sparse coding stage, we aim to solve the nonnegative sparse coding problem. Unfortunately, the sparse coding problem is NP-hard [23], and an approximation is required. We discuss several approaches in Section 3.1.1. In the second stage we aim to enhance the basis matrix $\mathbf{W}$. Here we allow that non-zero values in $\mathbf{H}$ are adapted during this step, but we do *not* allow that zero values become non-zero, i.e. we require that the sparse structure of $\mathbf{H}$ is maintained. We will see in Section 3.1.2 that all NMF techniques discussed so far can be used for this purpose. Dependent on the methods used for each stage, we can derive different algorithms from the generic scheme. Note that NNK-SVD [18] and the ℓ^0-constrained NMF proposed by Morup et al. [20] also follow this framework.

Algorithm 2. NMFℓ^0-H.

1: Initialize $\mathbf{W}$ randomly
2: **for** $i = 1$: numlter **do**
3: **Nonnegative Sparse Coding:** Sparsely encode data $\mathbf{X}$, using fixed basis matrix $\mathbf{W}$, resulting in a sparse, nonnegative matrix $\mathbf{H}$.
4: **Basis Matrix Update:** Enhance basis matrix $\mathbf{W}$ and coding matrix $\mathbf{H}$, maintaining the sparse structure of $\mathbf{H}$.
5: **end for**

3.1.1. Nonnegative sparse coding

The nonnegative sparse coding problem is formulated as

$$\begin{aligned} \underset{\mathbf{H}}{\text{minimize}} \quad & \|\mathbf{X}-\mathbf{W}\mathbf{H}\|_F \\ \text{subject to} \quad & \mathbf{H} \geq 0, \\ & \|\mathbf{h}_i\|_0 \leq L, \quad \forall i. \end{aligned} \tag{7}$$

Without loss of generality, we assume that the columns of $\mathbf{W}$ are normalized to unit length. A well known and simple sparse coding technique *without* nonnegativity constraints is orthogonal matching pursuit (OMP) [24], which is shown in Algorithm 3. OMP is a popular technique, due to its simplicity, its low computational complexity, and its theoretical optimality bounds [25,26].

Algorithm 3. Orthogonal matching pursuit (OMP).

1: $\mathbf{r} \leftarrow \mathbf{x}$
2: $\mathbf{h} = \mathbf{0}$
3: $\mathcal{P} = \emptyset$
4: **for** l = 1:L **do**
5: $\mathbf{a} = \mathbf{W}^T\mathbf{r}$
6: $i^* = \arg\max|\mathbf{a}|$
7: $\mathcal{P} \leftarrow \mathcal{P} \cup \{i^*\}$
8: $\mathbf{h}_{\mathcal{P}} \leftarrow \mathbf{W}_{\mathcal{P}}^{\dagger}\mathbf{x}$
9: $\mathbf{r} \leftarrow \mathbf{x} - \mathbf{W}_{\mathcal{P}}\mathbf{h}_{\mathcal{P}}$
10: **end for**

In each iteration, OMP selects the basis vector which reduces the residual $\mathbf{r}$ most (steps 6 and 7). After each selection step, OMP projects the data vector into the space spanned by the basis vectors selected so far (step 8).

Several authors proposed nonnegative variants of OMP [26,34,35], where all of them replace step 6 with $i^* = \arg\max \mathbf{a}$, i.e. the absolute value function is removed in order to select a basis vector with a *positive* coefficient. The second point, where nonnegativity can be violated, is in step 8, the projection step. Bruckstein et al. [26] used NNLS [27] (Algorithm 1) instead of ordinary least squares, in order to maintain nonnegativity. Since this variant is very slow, we used the multiplicative NMF update rule for $\mathbf{H}$ (cf. (2)), in order to approximate NNLS [35]. Yang et al. [34] left step 8 unchanged, which violates nonnegativity in general.

However, there is a more natural approach for nonnegative OMP: Note that the body of the outer loop of the active-set algorithm for NNLS (Algorithm 1) and OMP (Algorithm 3) perform *identical* computations, except that OMP selects $i^* = \arg\max|\mathbf{a}|$, while NNLS selects $i^* = \arg\max \mathbf{a}$, exactly as in the nonnegative OMP variants proposed so far. NNLS additionally contains the inner loop (11–18) to correct a possibly negative solution. Hence, a straightforward modification for nonnegative sparse coding, which we call sNNLS (which equally well can be called nonnegative OMP), is simply to stop NNLS, as soon as L basis vectors have been selected (i.e. as soon as $|\mathcal{P}| = L$ in Algorithm 1). Note that NNLS can also terminate before it selects L basis vectors. In this case, however, the solution is guaranteed to be optimal [27]. Hence, sNNLS guarantees to find either a nonnegative solution with *exactly* L positive coefficients, or an *optimal* nonnegative solution with less than L positive coefficients. Note further that OMP and NNLS have been proposed in distinct communities, namely sparse representation versus nonnegatively constrained least squares. It seems that this connection has been missed so far. The computational complexity of sNNLS is upper bounded by the original NNLS algorithm [27] (see Section 2.2), since it only contains an additional stopping criterion ($|\mathcal{P}| = L$).

We further propose a variant of matching pursuit, which does not only apply to the nonnegative framework, but generally to the matching pursuit principle. We call this variant *reverse* matching pursuit: instead of *adding* basis vectors to an initially empty set, we *remove* basis vectors from an optimal, non-sparse solution. For nonnegative sparse coding, this method is illustrated in Algorithm 4, to which we refer as *reverse sparse NNLS* (rsNNLS). The algorithm starts with an optimal, non-sparse solution in step 1. While the ℓ^0-norm of the solution is greater than L, the smallest entry in the solution vector is set to zero and its index is moved from the in-active set $\mathcal{P}$ to the active set $\mathcal{Z}$ (steps 4–7). Subsequently, the data vector is approximated in NNLS sense by the remaining basis vectors in $\mathcal{P}$, using steps 9–19 of Algorithm 1 (inner correction loop), where possibly additional basis vectors are removed from $\mathcal{P}$. For simplicity, Algorithm 4 is shown for a single data vector $\mathbf{x}$ and corresponding output vector $\mathbf{h}$. An implementation for data matrices $\mathbf{X}$, using the fast combinatorial NNLS algorithm [29], is straightforward. Similar as for sNNLS, the computational complexity of rsNNLS is not worse than for active-set NNLS [27] (Section 2.2), since in each iteration one index is irreversibly removed from $\mathcal{P}$. The NNLS algorithm (step 1)

needs at least the same number of iterations to build the non-sparse solution. Hence, rsNNLS needs at most twice as many operations as NNLS.

Algorithm 4. Reverse sparse NNLS (rsNNLS).

1: $\mathbf{h} = \text{NNLS}(\mathbf{x},\mathbf{W})$
2: $\mathcal{Z} = \{i|h_i = 0\}, \mathcal{P} = \{i|h_i > 0\}$
3: **while** $\|\mathbf{h}\|_0 > L$ **do**
4: $j = \arg\min_{i \in \mathcal{P}} h_i$
5: $h_j \leftarrow 0$
6: $\mathcal{Z} \leftarrow \mathcal{Z} \cup \{j\}$
7: $\mathcal{P} \leftarrow \mathcal{P} \backslash \{j\}$
8: Perform steps 9–19 of Algorithm 1.
9: **end while**

The second main approach for sparse coding without non-negativity constraints is *basis pursuit* (BP) [28], which relaxes the ℓ^0-norm with the convex ℓ^1-norm. Similar as for matching pursuit, there exist theoretical optimality guarantees and error bounds for BP [36,15]. Typically, BP produces better results than OMP [28,25]. *Nonnegative* BP (NNBP) can be formulated as

$$\begin{aligned} \underset{\mathbf{h}}{\text{minimize}} \quad & \|\mathbf{x}-\mathbf{W}\mathbf{h}\|_2 + \lambda\|\mathbf{h}\|_1 \\ \text{subject to} \quad & \mathbf{h} \geq 0, \end{aligned} \tag{8}$$

where λ controls the trade-off between reconstruction quality and sparseness. Alternatively, we can formulate NNBP as

$$\begin{aligned} \underset{\mathbf{h}}{\text{minimize}} \quad & \|\mathbf{h}\|_1 \\ \text{subject to} \quad & \|\mathbf{x}-\mathbf{W}\mathbf{h}\|_2 \leq \epsilon, \\ & \mathbf{h} \geq 0, \end{aligned} \tag{9}$$

where ϵ is the desired maximal reconstruction error. Formulation (9) is more convenient than (8), since the parameter ϵ is more intuitive and easier to choose than λ. Both (8) and (9) are convex optimization problems and can readily be solved [37]. To obtain an ℓ^0-sparse solution from a solution of (8) or (9), we select the L basis vectors with the largest coefficients in $\mathbf{h}$ and calculate new, optimal coefficients for these basis vectors, using NNLS. All other coefficients are set to zero. The whole procedure is repeated for each column in $\mathbf{X}$, in order to obtain a coding matrix $\mathbf{H}$ for problem (7). NNK-SVD follows this approach for nonnegative sparse coding, using the algorithm described in [11] as approximation for problem (8).

3.1.2. Enhancing the basis matrix

Once we have obtained a sparse encoding matrix $\mathbf{H}$, we aim to enhance the basis matrix $\mathbf{W}$ (step 4 in Algorithm 2). We also allow the non-zero values in $\mathbf{H}$ to vary, but require that the coding scheme is maintained, i.e. that the "zeros" in $\mathbf{H}$ have to remain zero during the enhancement stage.

For this purpose, we can use a technique for unconstrained NMF as discussed in Section 2. Note that the multiplicative update rules (2) and (3) can be applied without any modification, since they consist of *element-wise* multiplications of the old factors with some update term. Hence, if an entry in $\mathbf{H}$ is zero before the update, it remains zero after the update. Nevertheless, the multiplicative updates do not increase $\|\mathbf{X}-\mathbf{W}\mathbf{H}\|_F$, and typically reduce the objective. Therefore, step 4 in Algorithm 2 can be implemented by executing (2) and (3) for several iterations.

We can also perform an update according to ANLS. Since there are no constraints on the basis matrix, we can proceed exactly as in Section 2.2 in order to update $\mathbf{W}$. To update $\mathbf{H}$, we have to make some minor modifications of the fast combinatorial NNLS algorithm [29], in order to maintain the sparse structure of $\mathbf{H}$. Let $\overline{\mathcal{Z}}$ be the set of indices depicting the zero-entries in $\mathbf{H}$ after the sparse coding stage. We have to take care that these entries remain zero during the active set algorithm, i.e. we simply do not allow that an entry, whose index is in $\overline{\mathcal{Z}}$, is moved to the in-active set. The convergence criterion of the algorithm (cf. [27]) has to be evaluated considering only entries whose indices are not in $\overline{\mathcal{Z}}$. Similarly, we can modify numerical approaches for NNLS such as the projected gradient algorithm [31]. Generally, the $\mathbf{W}$ and $\mathbf{H}$ matrices of the previous iteration should be used as initial solution for each ANLS iteration, which significantly enhances the overall speed of NMFℓ^0-H.

3.2. NMFℓ^0-W

We now address problem NMFℓ^0-W (6), where we again follow a two stage iterative approach, as illustrated in Algorithm 5. We first calculate an optimal, unconstrained solution for the basis matrix $\mathbf{W}$ (with fixed $\mathbf{H}$) in step 3. Next, we project the basis vectors onto the closest nonnegative vector in Euclidean space, satisfying the desired ℓ^0-constraints. This step is easy, since we simply have to delete all entries except the L largest ones. Step 7 enhances $\mathbf{H}$, where also the non-zero entries of $\mathbf{W}$ can be adapted, but maintaining the sparse structure of $\mathbf{W}$. As in Section 3.1.2, we can use the multiplicative update rules due to their sparseness maintaining property. Alternatively, we can also use the ANLS scheme, similar as in Section 3.1.2. Our framework is inspired by Hoyer's ℓ^1-sparse NMF [13], which uses gradient descent to minimize the Frobenius norm. After each gradient step, the algorithm projects the basis vectors onto the closest vector with desired ℓ^1-sparseness (cf. (4)).

Algorithm 5. NMFℓ^0-W.

1: Initialize $\mathbf{H}$ randomly
2: **for** $i = 1$: numIter **do**
3: $\mathbf{W}^T = \text{NNLS}(\mathbf{X}^T,\mathbf{H}^T)$
4: **for** $j = 1 : K$ **do**
5: Set $D-L$ smallest values in $\mathbf{w}_j$ to zero
6: **end for**
7: **Coding Matrix Update:** Enhance the coding matrix $\mathbf{H}$ and basis matrix $\mathbf{W}$, maintaining the sparse structure of $\mathbf{W}$.
8: **end for**

4. Experiments

4.1. Nonnegative sparse coding

In this section we empirically compare the nonnegative sparse coding techniques discussed in this paper. To be able to evaluate the quality of the sparse coders, we created synthetic sparse data as follows. We considered random overcomplete basis matrices with $D=100$ dimensions and containing $K \in \{200, 400, 800\}$ basis vectors, respectively. For each K, we generated 10 random basis matrices using isotropic Gaussian noise; then the sign was discarded and each vector was normalized to unit length. Further, for each basis matrix, we generated "true" $K \times 100$ sparse encoding matrices $\mathbf{H}$ with sparseness factors $L \in \{5, 10, \ldots, 50\}$, i.e. we varied the sparseness from $L=5$ (very sparse) to $L=50$ (rather dense). The values of the non-zero entries in $\mathbf{H}$ were the absolute value of samples from a Gaussian distribution with standard deviation 10. The sparse synthetic data $\mathbf{X}$ is generated using $\mathbf{X} = \mathbf{W}\mathbf{H}$. We executed NMP [35], NNBP, sNNLS, rsNNLS and NLARS[3] [20] on each data set. For NNBP we used formulation (9), where ϵ was chosen such that an SNR of 120 dB was achieved, and we used an interior point solver [38] for

[3] We used the MATLAB implementation available under http://www2.imm.dtu.dk/pubdb/views/edoc_download.php/5523/zip/imm5523.zip.

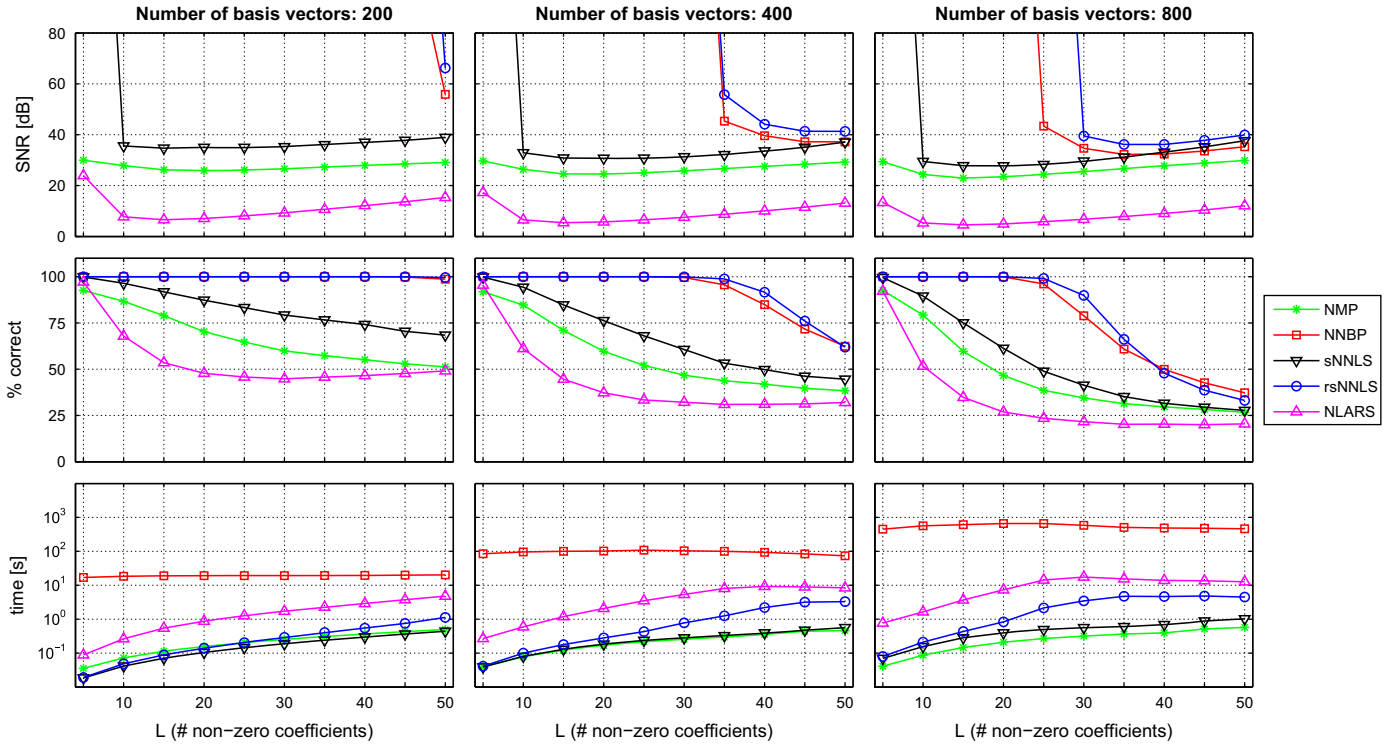

Fig. 1. Results of sparse coders, applied to sparse synthetic data. All results are shown as a function of the number of non-zero coefficients L (sparseness factor), and are averaged over 10 independent data sets. First row: reconstruction quality in terms of SNR. Markers outside the plot area correspond to perfect reconstruction (>120 dB). Second row: percentage of correctly identified basis vectors. Third row: runtime on logarithmic scale. Abbreviations: nonnegative matching pursuit (NMP) [35], nonnegative basis pursuit (NNBP) [28,18], sparse nonnegative least squares (sNNLS) (this paper), reverse sparse nonnegative least squares (rsNNLS) (this paper), nonnegative least angle regression and selection (NLARS) [20].

optimization. All algorithms were executed on a quad-core processor (3.2 GHz, 12 GB memory).

Fig. 1 shows the performance of the sparse coders in terms of reconstruction quality ($SNR = 10\log_{10}\|\mathbf{X}\|_F^2/\|\mathbf{X}-\mathbf{WH}\|_F^2$ dB), percentage of correctly identified basis vectors, and runtime, averaged over the 10 independent data sets per combination of K and L, where the SNR was averaged in the linear domain (i.e. arithmetic mean). Curve parts outside the plot area correspond to SNR values larger than 120 dB (which we considered as perfect reconstruction). NLARS shows clearly the worst performance, while requiring the most computational time after NNBP. sNNLS performs consistently better than NMP, while being approximately as efficient in terms of computational requirements. The proposed rsNNLS clearly shows the best performance, except that it selects slightly less correct basis vectors than NNBP for $K=800$ and $L>40$. Note that NNBP requires far the most computational time,[4] and that the effort for NNBP depends heavily on the number of basis vectors K. rsNNLS is orders of magnitude faster than NNBP.

We repeated the experiment, where we added positive, uniformly distributed noise to the synthetic sparse data, such that SNR $=10$ dB. Note that the choice of ϵ for NNBP is now more difficult: A too small ϵ renders problem (9) infeasible, while a too large ϵ delivers poor results. Therefore, for each column in $\mathbf{X}$, we initially selected ϵ according to SNR $=19$ dB. For the case where problem (9) turned out to be infeasible, we relaxed the SNR constraint by -3 dB until a feasible problem was obtained. The results of the experiment with noisy data is shown in Fig. 2. As expected, all sparse coders achieve a lower reconstruction quality and identify less correct basis vectors than in the noise-free case. The proposed rsNNLS shows the best performance for noisy data.

4.2. NMFℓ^0-H applied to spectrogram data

In this section, we compare methods for the update stage in NMFℓ^0-H (Algorithm 2). As data we used a magnitude spectrogram of 2 min of speech from the database by Cooke et al. [39]. The speech was sampled at 8 kHz and a 512 point FFT with an overlap of 256 samples was used. The data matrix finally had dimensions 257×3749. We executed NMFℓ^0-H for 200 iterations, where rsNNLS was used as sparse coder. We compared the update method from NNK-SVD [18], the multiplicative update rules [1] (Section 3.1.2), and ANLS using the sparseness maintaining active-set algorithm (Section 3.1.2). Note that these methods are difficult to compare, since one update iteration of ANLS fully optimizes first $\mathbf{W}$, then $\mathbf{H}$, while one iteration of NNK-SVD or the multiplicative rules only *reduce* the objective, which is significantly faster. Therefore, we proceeded as follows: we executed 10 (inner) ANLS iterations per (outer) NMFℓ^0-H iteration, and recorded the required time. Then we executed the versions using NNK-SVD and the multiplicative updates, respectively, where both were allowed to update $\mathbf{W}$ and $\mathbf{H}$ for the same amount of time as ANLS in each outer iteration. Since NNK-SVD updates the columns of $\mathbf{W}$ separately [18], we let NNK-SVD update each column for a kth fraction of the ANLS time.

For the number of basis vectors K and sparseness value L, we used each combination of $K\in\{100,250,500\}$ and $L\in\{5,10,20\}$. Since the error depends strongly on K and L, we calculated the root mean square error (RMSE) of each update method *relative* to the error of the ANLS method, and averaged over K and L: $\mathrm{RMSE}_{rel}(i) = \frac{1}{9}\sum_{K,L}\|\mathbf{X}-\mathbf{W}(K,L,i)\mathbf{H}(K,L,i)\|_F/\|\mathbf{X}-\mathbf{W}_{\mathrm{ANLS}}(K,L,i)\mathbf{H}_{\mathrm{ANLS}}(K,\ L,i)\|_F$, where

[4] For NNBP we used a C++ implementation [38], while all other algorithms were implemented in MATLAB. We also tried an implementation using the MATLAB optimization toolbox for NNBP, which was slower by a factor of 3–4.

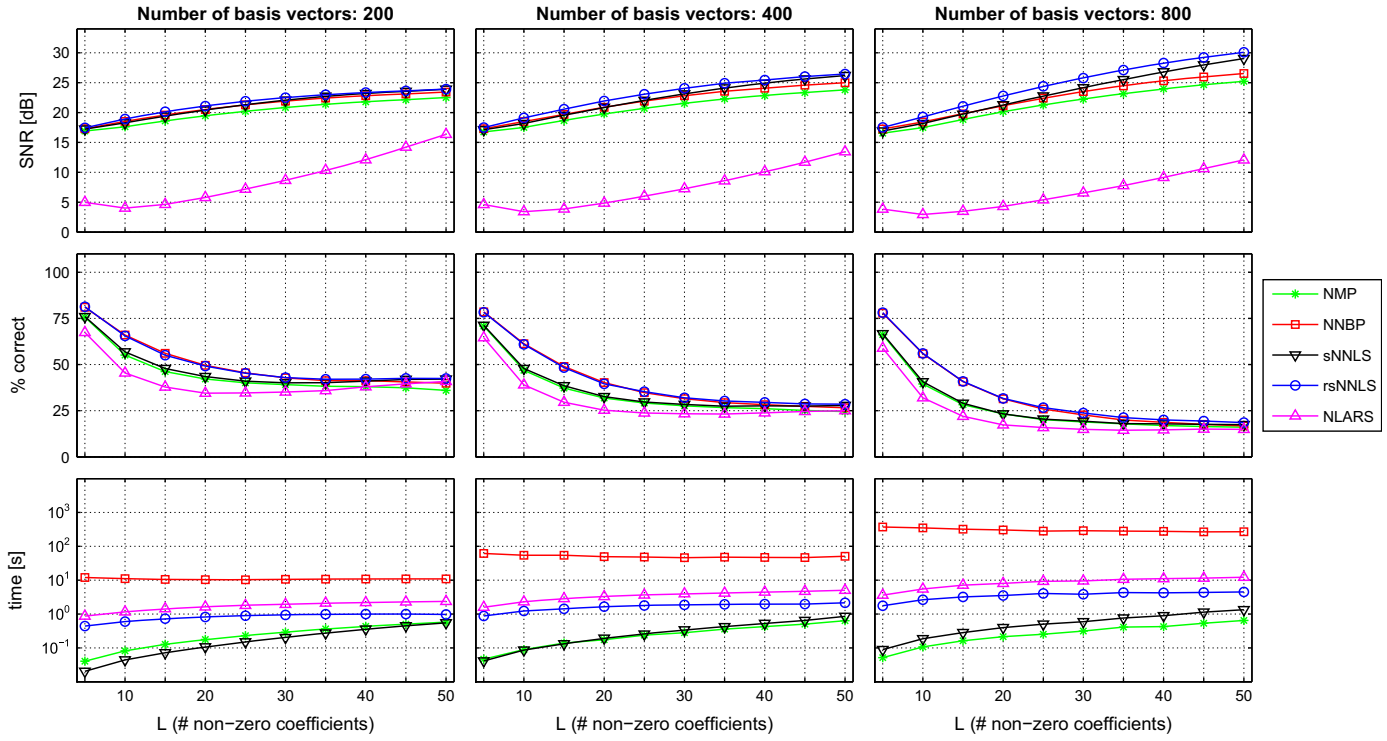

Fig. 2. Results of sparse coders, applied to sparse synthetic data, contaminated with noise (SNR = 10 dB). All results are shown as a function of L and are averaged over 10 independent data sets. First row: reconstruction quality in terms of SNR. Second row: percentage of correctly identified basis vectors. Third row: runtime on logarithmic scale. Abbreviations: nonnegative matching pursuit (NMP) [35], nonnegative basis pursuit (NNBP) [28,18], sparse nonnegative least squares (sNNLS) (this paper), reverse sparse nonnegative least squares (rsNNLS) (this paper), nonnegative least angle regression and selection (NLARS) [20].

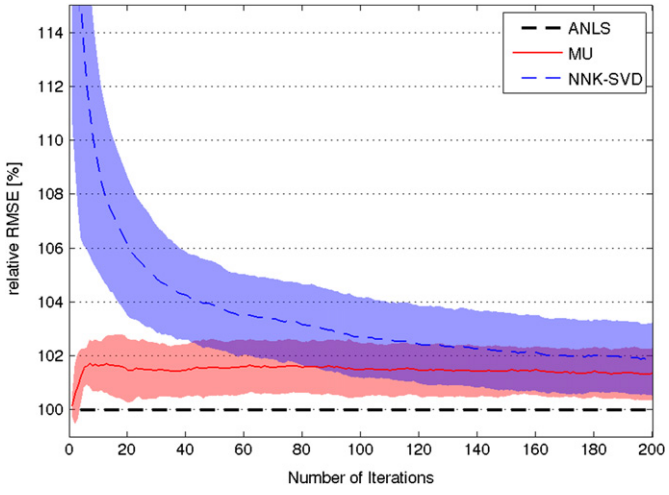

Fig. 3. Averaged relative RMSE (compared to ANLS update) of multiplicative update rules (MU) and nonnegative K-SVD (NNK-SVD) over number of NMFℓ^0-H iterations. The shaded bars correspond to standard deviation.

i denotes the iteration number of NMFℓ^0-H, $\mathbf{W}(K,L,i)$ and $\mathbf{H}(K,L,i)$ are the factor matrices of the method under test in iteration i, for parameters K and L. Similarly, $\mathbf{W}_{\text{ANLS}}(K,L,i)$ and $\mathbf{H}_{\text{ANLS}}(K,L,i)$ are the factor matrices of the ANLS update method. Fig. 3 shows the averaged relative RMSE for each update method, as a function of i. The ANLS approach shows the best performance. The multiplicative update rules are a constant factor worse than ANLS, but perform better than NNK-SVD. After 200 iterations, the multiplicative update rules and NNK-SVD achieve an average error which is approximately 1.35% and 1.9% worse than the ANLS error, respectively. NNK-SVD converges slower than the other update methods in the first 50 iterations.

4.3. NMFℓ^0-*W applied to face images*

Lee and Seung [5] reported that NMF returns a part-based representation, when applied to a database of face images. However, as noted by Hoyer [13], this effect does not occur when the face images are not aligned, like in the ORL database [40]. In this case, the representation happens to be rather global and holistic. In order to enforce a part-based representation, Hoyer constrained the basis vectors to be sparse. Similarly, we applied NMFℓ^0-W to the ORL database, where we constrained the basis vectors to have sparseness values L corresponding to 33%, 25% and 10% of the total pixel number per image (denoted as sparseness classes a, b and c, respectively) As in [13], we trained 25 basis vectors per sparseness value. We executed the algorithm for 30 iterations, using 10 ANLS coding matrix updates per iteration. In order to compare our results to ℓ^1-sparse NMF [13], we calculated the average ℓ^1-sparseness (using (4)) of our basis vectors. We executed ℓ^1-sparse NMF on the same data, where we required the basis vectors to have the same ℓ^1-sparseness as our NMFℓ^0-W basis vectors. We executed the algorithm for 2500 iterations, which were necessary for convergence. Fig. 4 shows the resulting basis vectors returned by NMFℓ^0-W and ℓ^1-sparse NMF, where dark pixels indicate high values and white pixels indicate low values.

We see that the results are qualitatively similar, and that the representation becomes a more local one, when sparseness is increased (from left to right). We repeated this experiment 10 times and calculated the ℓ^0-sparseness (in % of non-zero pixels), the $SNR = 10\log_{10}\|\mathbf{X}\|_F^2/\|\mathbf{X}-\mathbf{WH}\|_F^2$ dB and measured the runtime

Fig. 4. Top: basis images trained by NMFℓ^0-W. Bottom: basis images trained by ℓ^1-sparse NMF. Sparseness: 33% (a), 25% (b), 10% (c), 52.4% (d), 43% (e), 18.6% (f) of non-zero pixels.

Table 1
Comparison of ℓ^0-sparseness (in percent of non-zero pixels), ℓ^1-sparseness (cf. (4)), reconstruction quality in terms of SNR, and runtime for ℓ^1-sparse NMF [13] and NMFℓ^0-W. SNR* denotes the SNR value for ℓ^1-NMF, when the same ℓ^0-sparseness as for NMFℓ^0-W is enforced.

Method	ℓ^0 (%)	ℓ^1	SNR (dB)	SNR* (dB)	Time (s)
ℓ^1-NMF	52.4	0.55	15.09	14.55	2440
NMFℓ^0-W	33	0.55	15.07	–	186
ℓ^1-NMF	43	0.6	14.96	14.31	2495
NMFℓ^0-W	25	0.6	14.94	–	164
ℓ^1-NMF	18.6	0.73	14.31	13.52	2598
NMFℓ^0-W	10	0.73	14.33	–	50

for both methods. The averaged results are shown in Table 1, where the *SNR* was averaged in the linear domain.

We see that the two methods achieve de facto the same reconstruction quality, while NMFℓ^0-W uses a significantly lower number of non-zero pixels. One might ask if the larger portion of non-zeros in ℓ^1-NMF basis vectors stems from over-counting entries which are extremely small, yet numerically non-zero. Therefore, Table 1 additionally contains a column showing SNR* for ℓ^1-NMF, which denotes the SNR value when the target ℓ^0-sparseness is enforced, i.e. all but the 33%, 25%, 10% of pixels with largest values are set to zero. We see that when a strict ℓ^0-constraint is required, NMFℓ^0-W achieves a significantly better SNR (at least 0.5 dB in this example). Furthermore, NMFℓ^0-W is more than an order of magnitude faster than ℓ^1-sparse NMF in this setup.

5. Conclusion

We proposed a generic alternating update scheme for ℓ^0-sparse NMF, which naturally incorporates existing approaches for unconstrained NMF, such as the classic multiplicative update rules or the ANLS scheme. For the key problem in NMFℓ^0-H, the nonnegative sparse coding problem, we proposed sNNLS, whose interpretation is twofold: it can be regarded as sparse nonnegative least-squares or as nonnegative orthogonal matching pursuit. From the matching-pursuit perspective, sNNLS represents a natural implementation of nonnegativity constraints in OMP. We further proposed the simple but astonishingly well performing rsNNLS, which competes with or outperforms nonnegative basis pursuit. Generally, ℓ^0-sparse NMF is a powerful technique with a large number of potential applications. NMFℓ^0-H aims to find a (possibly overcomplete) representation using sparsely activated basis vectors. NMFℓ^0-W encourages a part-based representation, which might be particularly interesting for applications in computer vision.

Acknowledgment

This work was supported by the Austrian Science Fund (Project number P22488-N23). The authors like to thank Sebastian Tschiatschek for his support concerning the Ipopt library, and his useful comments concerning this paper.

References

[1] D.D. Lee, H.S. Seung, Algorithms for non-negative matrix factorization, in: NIPS, 2001, pp. 556–562.
[2] I.S. Dhillon, S. Sra, Generalized nonnegative matrix approximations with Bregman divergences, in: NIPS, 2005, pp. 283–290.
[3] A. Cichocki, H. Lee, Y.D. Kim, S. Choi, Non-negative matrix factorization with α-divergence, Pattern Recognition Lett. 29 (9) (2008) 1433–1440.
[4] P. Paatero, U. Tapper, Positive matrix factorization: a non-negative factor model with optimal utilization of error estimates of data values, Environmetrics 5 (1994) 111–126.
[5] D.D. Lee, H.S. Seung, Learning the parts of objects by nonnegative matrix factorization, Nature 401 (1999) 788–791.
[6] P. Smaragdis, J. Brown, Non-negative matrix factorization for polyphonic music transcription, in: IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 2003, pp. 177–180.
[7] T. Virtanen, Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria, IEEE Trans. Audio Speech Lang. Process. 15 (3) (2007) 1066–1074.
[8] 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.
[9] R. Peharz, M. Stark, F. Pernkopf, A factorial sparse coder model for single channel source separation, in: Proceedings of Interspeech, 2010, pp. 386–389.
[10] W. Xu, X. Liu, Y. Gong, Document clustering based on non-negative matrix factorization, in: Proceedings of ACM SIGIR'03, 2003, pp. 267–273.
[11] P.O. Hoyer, Non-negative sparse coding, in: Proceedings of Neural Networks for Signal Processing, 2002, pp. 557–565.
[12] J. Eggert, E. Koerner, Sparse coding and NMF, in: International Joint Conference on Neural Networks, 2004, pp. 2529–2533.
[13] P.O. Hoyer, Non-negative matrix factorization with sparseness constraints, J. Mach. Learn. Res. 5 (2004) 1457–1469.
[14] D. Donoho, M. Elad, Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ^1 minimization, Proc. Natl. Acad. Sci. 100 (5) (2003) 2197–2202.
[15] J. Tropp, Just relax: convex programming methods for identifying sparse signals, IEEE Trans. Inf. Theory 51 (2006) 1030–1051.
[16] S. Vavasis, On the complexity of nonnegative matrix factorization, SIAM J. Matrix Anal. Appl. J. Optim. 20 (3) (2009) 1364–1377.
[17] M. Aharon, M. Elad, A. Bruckstein, K-SVD: an algorithm for designing overcomplete dictionaries for sparse representation, IEEE Trans. Signal Process. 54 (2006) 4311–4322.
[18] M. Aharon, M. Elad, A. Bruckstein, K-SVD and its non-negative variant for dictionary design. in: Proceedings of the SPIE Conference, Curvelet, Directional, and Sparse Representations II, vol. 5914, 2005, pp. 11.1–11.13.
[19] D. Dueck, B. Frey, Probabilistic Sparse Matrix Factorization, Technical Report PSI TR 2004-023, Department of Computer Science, University of Toronto, 2004.
[20] M. Morup, K. Madsen, L. Hansen, Approximate L_0 constrained non-negative matrix and tensor factorization, in: Proceedings of ISCAS, 2008, pp. 1328–1331.
[21] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani, Least angle regression, Ann. Stat. 32 (2004) 407–499.
[22] Z. Yang, X. Chen, G. Zhou, S. Xie, Spectral unmixing using nonnegative matrix factorization with smoothed L0 norm constraint, in: Proceedings of SPIE, 2009.
[23] G. Davis, S. Mallat, M. Avellaneda, Adaptive greedy approximations, J. Constr. Approx. 13 (1997) 57–98.

[24] Y.C. Pati, R. Rezaiifar, P.S. Krishnaprasad, Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition, in: Proceedings of 27th Asilomar Conference on Signals, Systems and Computers, 1993, pp. 40–44.
[25] J. Tropp, Greed is good: algorithmic results for sparse approximation, IEEE Trans. Inf. Theory 50 (10) (2004) 2231–2242.
[26] A.M. Bruckstein, M. Elad, M. Zibulevsky, On the uniqueness of nonnegative sparse solutions to underdetermined systems of equations, IEEE Trans. Inf. Theory 54 (2008) 4813–4820.
[27] C. Lawson, R. Hanson, Solving Least Squares Problems, Prentice-Hall, 1974.
[28] S. Chen, D. Donoho, M. Saunders, Atomic decomposition by basis pursuit, SIAM J. Sci. Comput. 20 (1) (1998) 33–61.
[29] M. Van Benthem, M. Keenan, Fast algorithm for the solution of large-scale non-negativity-constrained least-squares problems, J. Chemometrics 18 (2004) 441–450.
[30] H. Kim, H. Park, Non-negative matrix factorization based on alternating non-negativity constrained least squares and active set method, SIAM J. Matrix Anal. Appl. 30 (2008) 713–730.
[31] C.J. Lin, Projected gradient methods for nonnegative matrix factorization, Neural Comput. 19 (2007) 2756–2779.
[32] S. Li, X. Hou, H. Zhang, Q. Cheng, Learning spatially localized, parts-based representation, in: Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), 2001.
[33] A. Pascual-Montano, J. Carazo, K. Kochi, D. Lehmann, R. Pascual-Marqui, Nonsmooth nonnegative matrix factorization (NSNMF), in: Transactions on Pattern Analysis and Machine Intelligence, 2006.
[34] A.Y. Yang, S. Maji, K. Hong, P. Yan, S.S. Sastry, Distributed compression and fusion of nonnegative sparse signals for multiple-view object recognition, in: International Conference on Information Fusion, 2009.
[35] P. Peharz, M. Stark, F. Pernkopf, Sparse nonnegative matrix factorization using ℓ^0-constraints, in: Proceedings of MLSP, 2010, pp. 83–88.
[36] D.L. Donoho, J. Tanner, Sparse nonnegative solutions of underdetermined linear equations by linear programming, Proc. Natl. Acad. Sci. 102 (27) (2005) 9446–9451.
[37] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, 2004.
[38] A. Wächter, L.T. Biegler, On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming, Math. Programming 106 (2006) 25–57.
[39] M.P. Cooke, J. Barker, S.P. Cunningham, X. Shao, An audio-visual corpus for speech perception and automatic speech recognition, J. Am. Stat. Assoc. 120 (2006) 2421–2424.
[40] F.S. Samaria, A. Harter, Parameterisation of a stochastic model for human face identification, in: Proceedings of the 2nd IEEE Workshop on Applications of Computer Vision, 1994, pp. 138–142.

Robert Peharz received his MSc degree in 2010. He currently pursues his PhD studies at the SPSC Lab, Graz University of Technology. His research interests include nonnegative matrix factorization, sparse coding, machine learning and graphical models, with applications to signal processing, audio engineering and computer vision.

Franz Pernkopf received his PhD degree in 2002. In 2002 he received the Erwin Schrödinger Fellowship. From 2004 to 2006, he was Research Associate in the Department of Electrical Engineering at the University of Washington. He received the young investigator award of the province of Styria in 2010. Currently, he is Associate Professor at the SPSC Lab, Graz University of Technology, leading the Intelligent Systems research group. His research interests include machine learning, discriminative learning, graphical models, with applications to image- and speech processing.