García Mollá, VM.; San Juan-Sebastian, P.; Virtanen, T.; Vidal Maciá, AM.; Alonso-Jordá, P. (2019). Generalization of the K-SVD algorithm for minimization of ß-divergence. Digital Signal Processing. 92:47-53. https://doi.org/10.1016/j.dsp.2019.05.001

Generalization of the K-SVD Algorithm for Minimization of β-divergence

Victor M. Garcia-Molla[a,1], Pablo San Juan[a], Tuomas Virtanen[b], Antonio M. Vidal[a], Pedro Alonso[a]

[a] *Victor M. Garcia-Molla, Pablo San Juan, Antonio M. Vidal and Pedro Alonso are with Department of Information Systems and Computing, Universitat Politècnica de València, SPAIN e-mail: {vmgarcia,p.sanjuan,amvidal,palonso}@upv.es*

[b] *Tuomas Virtanen is with Department of Signal Processing,Tampere University of Technology, FINLAND e-mail: tuomas.virtanen@tut.fi*

Abstract

In this paper, we propose, describe, and test a modification of the K-SVD algorithm. Given a set of training data, the proposed algorithm computes an overcomplete dictionary by minimizing the β-divergence ($\beta >= 1$) between the data and its representation as linear combinations of atoms of the dictionary, under strict sparsity restrictions. For the special case $\beta = 2$, the proposed algorithm minimizes the Frobenius norm and, therefore, for $\beta = 2$ the proposed algorithm is equivalent to the original K-SVD algorithm. We describe the modifications needed and discuss the possible shortcomings of the new algorithm. The algorithm is tested with random matrices and with an example based on speech separation.

Keywords: K-SVD; Nonnegative K-SVD; beta-divergence; NMF; Matching pursuit algorithms

1. Introduction

Nowadays, the computation of sparse representations of signals of any kind is a very active research field. The generation of overcomplete dictionaries is one of the specific goals in the field. A popular algorithm for computation of

*Corresponding author
Email address: vmgarcia@upv.es (Victor M. Garcia-Molla)

overcomplete dictionaries is the K-SVD algorithm, which was first proposed in [1]. More precisely, given a large family of signals $y_i, i = 1..m$, stored as columns of a large data matrix Y, and given an integer T_0 (maximum number of nonzero elements), the goal of the K-SVD algorithm is to obtain matrices D (the dictionary of signals) and X (the coefficients matrix) such that the Frobenius norm

$$argmin_{D,X}\|Y - DX\|_{fro} \tag{1}$$

is minimized, while the maximum number of nonzero elements in each column of X is T_0. The K-SVD algorithm iteratively computes an overcomplete dictionary, using as tools a sparse pursuit algorithm (such as Orthogonal Matching Pursuit (OMP) [2, 3], Basis Pursuit (BP) [4], etc.) and an iterative algorithm that improves the dictionary using the Singular Value Decomposition (SVD) [5]. Applications of the K-SVD algorithm are discussed in a large number of papers in image denoising, image classification, audio processing, etc. [6, 7, 8, 9]. There also exists a nonnegative K-SVD algorithm (NNKSVD) [10], where the data matrix Y must be nonnegative ($y_{i,j} \geq 0$) and the obtained dictionaries and coefficients matrices must also be nonnegative. As observed in [11], the NNKSVD can be regarded as a different algorithm for computing a Nonnegative Matrix Factorization (NMF) [13, 12].

Both versions of the K-SVD algorithm have been applied successfully to different problems. However, they are based on the minimization of the Frobenius norm. Minimizing the Frobenius norm corresponds to a maximum likelihood estimation in a model where additive Gaussian noise is assumed; however, in other situations, some other noise model and related minimization criterion may be more appropriate. It is well known that in research fields related to sound (music analysis, sound separation, etc.) better results are often obtained working with β-divergence [14]. As mentioned in [15], β-divergence is a family of cost functions that are characterized by a single parameter β. β-divergence has as special cases the Frobenius norm (when β=2), the Kullback-Leibler divergence

(when β=1), or the Itakura-Saito divergence (when β =0). The β parameter controls the statistical properties of the noise in the data being studied. Thus, for example, $\beta = 1$ is equivalent to assuming a Poisson model of noise, while intermediate values of β can be appropriate in other situations. In this paper, we propose a modification of the standard K-SVD algorithm so that the β-divergence is minimized. Given that minimization of β-divergence with $\beta = 2$ is equivalent to minimizing the Frobenius norm, the proposed modification is not really a different algorithm, but rather a generalization of the K-SVD algorithm, which allows the experimentation with values of β that might be more appropriate. The extension to the nonnegative case is also considered. It must be mentioned that the proposed algorithm does not converge for values of β smaller than 1, which is a significant shortcoming.

The proposed algorithm might be useful in any research field where the K-SVD is used, and, by extension, in any field where computation of overcomplete dictionaries is used. There are examples of minimization of β-divergences in image analysis [15], but there are many more in the field of Sound Analysis (voice, music, etc.) [16, 17]. A version of the K-SVD minimizing beta divergences can be an interesting tool for researchers in that field.

The paper has the following structure. First, we describe the standard K-SVD algorithm, its nonnegative version, and the concepts of β-divergence. In the second section, we discuss the adaptation of K-SVD for β-divergences. Finally, we show the results for two test problems.

1.1. Notation

-Given a vector $u \in \Re^{m\times 1}$ and a scalar $\beta \in \Re$, the component-wise exponentiation of the vector u to β is denoted as $u.^{\beta}$ and is the vector with components $u_i^{\beta}, i = 1 \dots m$.

-Given vectors $u \in \Re^{m\times 1}$, $v \in \Re^{m\times 1}$, the component-wise product of the vectors u and v is denoted as $u. * v$ and is the vector with components $u_i v_i, i = 1 \dots m$.

-Given vectors $u \in \Re^{m\times 1}$, $v \in \Re^{m\times 1}$, the component-wise division of the

vectors u and v is denoted as $u./v$ and is the vector with components $u_i/v_i, i = 1 \dots m$.

-Given a vector $u \in \Re^{m\times 1}$, $w = [u]_+$ denotes the vector with components $w_i = u_i$ if $u_i \geq 0$; $w_i = 0$ if $u_i < 0$).

All the previous conventions extend naturally to matrices.

-Given a matrix $A \in \Re^{m\times n}$, we will denote the k-th column of A as A_k and the k-th row of A as A_k^T. Given a set of column indices $w = \{i_1, i_2, ..., i_k, 1 \leq i_j \leq n\}$, A^w is the matrix formed with the columns of A whose index is in w.

2. State of the Art

2.1. K-SVD algorithm

The K-SVD algorithm was thoroughly described and tested in [1, 10]. It is built on top of two independent algorithms, a sparse pursuit algorithm and the updating of the dictionary D.

Given a dictionary $D \in \Re^{m\times K}$, a maximum number of nonzero elements T_0 and a data matrix $Y \in \Re^{m\times n}$, the sparse pursuit consists of obtaining a representation of the columns of Y as linear combination of a few columns of D. More precisely, it is the computation of the matrix $X \in \Re^{K\times n}$ such that each column of X has, at most, T_0 nonzero elements and minimizes the Frobenius norm $\|Y - DX\|_{fro}$.

This computation can be done column by column, but, for each column, the computation of the optimal solution is a NP-hard problem [25]. This problem is usually tackled through greedy approaches, such as the OMP algorithm. The solutions obtained may be not optimal, but experience shows that these solutions are quite acceptable.

After the sparse pursuit phase, the dictionary D is updated by proceeding column by column, using the SVD decomposition. Given a subroutine *sparsepurs* that implements the sparse pursuit phase through any appropriate pursuit algorithm, the complete K-SVD algorithm is as follows:

Algorithm 1 K-SVD Algorithm

Input: $Y \in \Re^{m \times n}$ data matrix, $K \in \mathcal{N}^+$ desired number of columns of the dictionary, $D \in \Re^{m \times K}$ initial dictionary, $T_0 \in N^+$ maximum number of nonzero elements

Output: $D \in \Re^{m \times K}$ dictionary, $X \in \Re^{K \times n}$ coefficients matrix

repeat

  Apply sparse pursuit algorithm: X $\leftarrow$ $sparsepurs(Y, D, \mathrm{T}_0)$

  /* loop for update of the columns of the dictionary D */

  for $k = 1$ **to** K **do**

    Find the set of columns of Y that use the atom D_k, i.e. $w_k = \{i | 1 \leq i \leq n, X(k,i) \neq 0\}$. Let the cardinal $|w_k|$ be q.

    Compute $E_k \leftarrow Y - (DX - D_k X_k^T)$

    Obtain E_k^w as E_k restricted to columns with index in w_k, therefore E_k^w has q columns

    Compute SVD of E_k^w, obtaining the first singular value σ_1 and first singular vectors: u_1, v_1^T

    update $D_k \leftarrow u_1$, $X_k^T \leftarrow \sigma_1 v_1^T$

  end for

until convergence

The initial dictionary D can be initialized with random numbers or with any other previously computed dictionary. The convergence of this algorithm is discussed in [1]. The conclusion given there is that, since the sparse pursuit phase is solved only approximately, the convergence cannot be theoretically guaranteed. However, in practice, the algorithm works well and converges to a local minimum.

2.2. Nonnegative K-SVD algorithm

The NNKSVD algorithm was described in [10]. It turns out to be a type of NMF, where sparsity is forced in the sparse pursuit phase. In order to obtain nonnegative results, both phases must be adapted.

The nonnegative sparse pursuit procedure must minimize

$$\|Y - DX\|_{fro}, X \geq 0, \tag{2}$$

where Y and D are fixed, enforcing the desired sparsity. The method proposed in [10] has two phases. The first phase minimizes the Frobenius norm in (2) through a multiplicative algorithm, like the one proposed in [13] (Actually, it is clear that this is equivalent to computing one of the two matrices in a NMF. Therefore, the methods used for NMF could be used for this computation). In the second phase, the sparsity is enforced. To this end, each column x_j of X is processed, and the largest T_0 elements of each column x_j are selected; let I be the indices of these elements, and let $x_{j,I}$ be the corresponding subvector. Then, let D^I be the submatrix of D composed by the columns of D whose index is in I. Then, a small nonnegative least squares problem ($\min \|y_j - D^I x_{j,I}\|$) is solved for each column of X.

The nonnegative updating of the dictionary proposed in [10] is carried out through an iterative method similar to the Power method, which is a method for computing the largest eigenvalue and associated eigenvector of a given matrix [5]. The Power method can also be used to compute the two singular vectors of a matrix associated with the largest singular value, as will be shown below.

2.3. β-Divergence

The β-divergence between vectors x and y is defined as

$$D_\beta(x||y) = \sum_i d_\beta(x_i, y_i) \tag{3}$$

where the function d_β (β-divergence between scalars) is defined in [15] as

$$d_\beta(x,y) = \begin{cases} \frac{1}{\beta(\beta-1)}\left(x^\beta + (\beta-1)y^\beta - \beta x y^{\beta-1}\right) & \beta \in \Re \setminus \{0,1\} \\ x\ log\frac{x}{y} - x + y & \beta = 1 \\ \frac{x}{y} - log\frac{x}{y} - 1 & \beta = 0 \end{cases} \tag{4}$$

The case with $\beta = 1$ is also known as the Kullback-Leibler divergence, while

the case with $\beta = 0$ is also known as the Itakura-Saito divergence. The definition given is valid for vectors, although it can be extended for matrices:

$$D_\beta(X||Y) = \sum_{i,j} d_\beta(x_{i,j}, y_{i,j}) \tag{5}$$

In the case of matrices, minimizing the β-divergence with $\beta = 2$ is equivalent to minimizing the Frobenius norm.

3. The Proposed K-SVD to use β-divergences

The goal of this paper is to propose a K-SVD algorithm that minimizes the β-divergence between Y and DX, instead of the Frobenius norm. We will denote this algorithm as β-K-SVD algorithm. Like the original K-SVD algorithm, a nonnegative version can be considered. Actually, the nonnegative version is more important from a practical point of view than the standard version. Indeed, most applications where β-divergences are used include non-negativity constraints. All of the steps of the algorithms that should be modified to obtain a nonnegative version will be marked.

In order to obtain such an algorithm, both phases of the K-SVD must be adapted to minimize β-divergences.

3.1. Adaptation of the updating of the dictionary

We will start by considering the second part, the updating of the dictionary. This part is the main novelty of this paper. First, it is important to note that, despite the name of the algorithm, the full SVD decomposition (which is quite an expensive algorithm that is executed inside the loop and is executed many times) is actually not needed.

Indeed, as can be seen in Algorithm 1, only the singular vectors u_1, v_1 associated to the largest singular value σ_1 of the E_k^w matrix are used (in line 11). The full SVD algorithm computes all of the singular values and all of the singular vectors, which is clearly unnecessary; there are simplified versions (SVDS

command in Matlab) that are more appropriate, since they only compute a few singular values and the associated singular vectors.

Furthermore, the singular value σ_1 does not have to be calculated separately. For the K-SVD algorithm, it is enough to compute two vectors, u and v, such that u is normalized ($\|u\|_2 = 1$) and the Frobenius norm of the difference between the outer product of u and v and the residual matrix E_k^w is minimized ($min\|E_k^w - uv^T\|_{fro}$). This is equivalent to finding the best rank-one approximation to E_k^w. These facts have been previously recognized in several papers [19, 20]. Therefore, fast implementations of the K-SVD do not use a full SVD, using instead incomplete SVD, or, even better, an appropriate version of the Power method, which is a very fast method for obtaining the desired u and v vectors. This algorithm is shown as Algorithm 2.

Algorithm 2 Power Algorithm for computation of the best rank-one approximation of a matrix A, i.e., computation of the vectors u, v such that $min\|A - uv^T\|_{fro}$ is minimized

Input: $A \in \Re^{m \times n}$
Output: $u \in \Re^{m \times 1}, v \in \Re^{1 \times n}$
$u \leftarrow random\ vector \in \Re^{m \times 1}$
$u \leftarrow \frac{u}{\|u\|_2}$
repeat
    $v \leftarrow A^t u$
    $u \leftarrow Av$
    $u \leftarrow \frac{u}{\|u\|_2}$
until convergence

This procedure is equivalent to finding the largest eigenvalue (and associated eigenvectors) of the symmetric and positive definite matrix $A^T A$. All of the eigenvalues of this matrix are real and positive. Even in the case when the maximum eigenvalue of $A^T A$ has multiplicity larger than 1, this algorithm returns vectors that minimize $\|A - uv^T\|_{fro}$.

Our goal is to obtain a similar algorithm, but for minimizing $D_\beta(A, uv^T)$. Such an algorithm is derived by obtaining the gradient of $D_\beta(A, uv^T)$ with respect to u and with respect to v, and equating to 0. For example, deriving $D_\beta(A, uv^T)$ with respect to u_i:

$$\frac{\partial}{\partial u_i} \sum_{i,j} d_\beta(A_{i,j}, u_i v_j) =$$
$$\frac{\partial}{\partial u_i} \sum_{i,j} \left(\frac{A_{i,j}}{\beta(\beta-1)} + \frac{(u_i v_j)^\beta}{\beta} - \frac{A_{i,j}(u_i v_j)^{\beta-1}}{\beta-1} \right) =$$
$$\sum_j \left(u_i^{\beta-1} v_j^\beta - A_{i,j} u_i^{\beta-2} v_j^{\beta-1} \right) \tag{6}$$

by equating the result in eq. 6 to zero and simplifying, we obtain an expression in the form of a matrix vector product that is similar to those in the power method:

$$u = \frac{A v.^{\beta-1}}{(v^T).^{\beta-1} v} \tag{7}$$

and, through a similar process, a similar expression is obtained for v:

$$v = \frac{A^T u.^{\beta-1}}{(u^T) u.^{\beta-1}} \tag{8}$$

These equations can be used in a power-like algorithm (listed as Algorithm 3) that we call *power_beta*. The nonnegative version is trivially obtained, using the alternative expressions shown as comments.

Algorithm 3 Power algorithm (*power_beta*) for computation of vectors u, v such that the β-divergence between the input matrix A and uv^T is minimized, and such that $\|u\|_2 = 1$

1: **Input**: $A \in \Re^{m \times n}, \beta \in \Re$
2: **Output**: $u \in \Re^{m \times 1}, v \in \Re^{1 \times n}$
3: $u \leftarrow random\ vector \in \Re^{m \times 1}$ /*Nonnegative version $u > 0$ */
4: $u \leftarrow \frac{u}{\|u\|_2}$
5: **repeat**
6: $\quad u_{aux} \leftarrow u.^{\beta-1}$
7: $\quad v \leftarrow A^t u_{aux}$ /*Nonnegative version $v = [A^t u_{aux}]_+$ */
8: $\quad v \leftarrow \frac{v}{u^t u_{aux}}$
9: $\quad v_{aux} \leftarrow v.^{\beta-1}$
10: $\quad u \leftarrow A v_{aux}$ /* Nonnegative version $u = [A v_{aux}]_+$*/
11: $\quad u \leftarrow \frac{u}{(v^t).^{\beta-1} v_{aux}}$
12: $\quad u \leftarrow \frac{u}{\|u\|_2}$
13: **until** convergence

The applicability of this algorithm is restricted if $\beta < 1$. In this case, the algorithm would fail if the matrix A contains any negative entry (or entry close to zero). For values of $\beta \geq 1$, the algorithm usually converges without problems. If $\beta = 2$, Algorithm 3 minimizes the Frobenius norm between A and uv^t and therefore is equivalent to Algorithm 2. In all of the cases tested, Algorithm 3 (for $\beta = 2$) returns the same approximation uv^T as Algorithm 2. From a mathematical point of view, the normalization in line 12 of Algorithm 3 is not strictly necessary, but it avoids numerical problems.

Algorithms 2 and 3 are clearly related to the Hierarchical Alternating Least Squares method (HALS), [12, 22], which is one of the better methods for computation of Nonnegative Matrix factorization. The HALS algorithm can work with the Frobenius norm or with β-divergence. Actually, the above derivation of the formulas for minimization of $D_\beta(A, uv^T)$ is virtually identical to the derivation of the formulas of HALS with β-divergence in [12] (although the use of these

formulas is different in this paper and in [12]).

3.2. Adaptation of the sparse pursuit phase to β-divergence

Now we will consider the sparse pursuit phase, which has been tackled in several papers [11, 10], especially in the nonnegative case. The simplest choices for the sparse pursuit algorithm are, either to use an adapted OMP [11] or to use a two-phase approach similar to the one proposed in [10], for the nonnegative case. For this paper, we have chosen the latter option.

For the first stage (minimization of the β-divergence between Y and DX, with D fixed), we initially used a multiplicative algorithm adapted to β-divergence (Algorithm 4), proposed in [12, 18] for the computation of the NMF for β-divergence, but which is used here for computing only the coefficient matrix X. Multiplicative algorithms like Algorithm 4 are derived from the Karush-Kuhn-Tucker conditions (see [12]). Different versions have been derived for the minimization of different metrics (Frobenius norm, β-divergence, α-divergence, etc. [12])and are the most popular family of algorithms for computing the NMF.

Algorithm 4 Multiplicative algorithm for computation of matrix X such that the β-divergence between the data matrix Y and $D \cdot X$ is minimized

Input: $Y \in \Re^{m \times n}, D \in \Re^{m \times K}, \beta \in \Re$
Output: $X \in \Re^{K \times n}$
$X \leftarrow random\ matrix \in \Re^{K \times n}$ /*Nonnegative version $X > 0$ */
$E \leftarrow Y - D\ X$
repeat
  $F1 \leftarrow D^T (DX).^{\beta-2}. * Y$
  $G1 \leftarrow D^T (DX).^{\beta-1}$
  $X \leftarrow X. * (F1./G1)$ /*Nonnegative version $X \leftarrow X. * [F1./G1]_+$ */
until convergence

However, we noticed that the β-divergence HALS algorithm (restricted to the coefficients matrix X, Algorithm 5) is usually faster than the multiplicative

method 4 (save for the special case $\beta = 1$, to be discussed below).

Algorithm 5 HALS-like Algorithm for computation of matrix X such that the β-divergence between the data matrix Y and DX is minimized

Input: $Y \in \Re^{m \times n}, D \in \Re^{m \times K}, \beta \in \Re$
Output: $X \in \Re^{K \times n}$
$X \leftarrow random\ matrix \in \Re^{K \times n}$ /*Nonnegative version $X > 0$ */
$E \leftarrow Y - DX$
repeat
  for $j = 1$ **to** K **do**
    $Y^{(j)} \leftarrow E + D_j X_j^T$
    $X_j^T \leftarrow \frac{Y^{(j)^T}(D_j.^{\beta-1})}{D_j^T(D_j.^{\beta-1})}$ /*Nonneg. version $X_j^T \leftarrow \frac{\left[Y^{(j)^T}(D_j.^{\beta-1})\right]_+}{D_j^T(D_j.^{\beta-1})}$ */
    $E \leftarrow Y^{(j)} - D_j X_j^T$
  end for
until convergence

The sparsity-enforcing phase starts like the one proposed in [10], column by column and selecting the T_0 largest coefficients of each column X_j and its indices I. However, the resulting subproblems are no longer least squares problems. The new subproblems can be formulated as: for each column j, minimize the β-divergence between Y_j and $D^I X_j^I$. In this case, we chose to use a β-divergence multiplicative method, with only 5 iterations, which seems to have worked reasonably well in all of the cases tested.

Our proposed sparse pursuit algorithm for β-divergence can be seen as Algorithm 6.

Algorithm 6 *Sparse_pursuit_beta* algorithm for computation of matrix X such that the β-divergence between the data matrix Y and DX is minimized and the maximum number of nonzero elements in each column of X is T_0

Input: $Y \in \Re^{m \times n}, D \in \Re^{m \times K}, T_0 \in \mathcal{N}, \beta \in \Re$

Output: $X \in \Re^{K \times n}$

$Z \leftarrow random\ matrix \in \Re^{K \times n}$

/* First phase, minimization of β-divergence through a HALS-like algorithm*/

$E \leftarrow Y - DZ$

repeat

  for $j = 1$ **to** K **do**

    $Y^{(j)} \leftarrow E + D_j\ Z_j^T$

    $Z_j^T \leftarrow \frac{{Y^{(j)}}^T (D_j.^{\beta-1})}{D_j^T (D_j.^{\beta-1})}$ /*Nonnegative version $Z_j^T \leftarrow \frac{\left[{Y^{(j)}}^T (D_j.^{\beta-1})\right]_+}{D_j^T (D_j.^{\beta-1})}$ */

    $E \leftarrow Y^{(j)} - D_j Z_j^T$

  end for

until convergence

/* Second phase, sparsity enforcing */

$X \leftarrow zero\ matrix \in \Re^{K \times n}$

for $j = 1$ **to** n **do**

  Let I be the set of the indices of the T_0 largest elements in Z_j; let Z_j^I be the subvector composed by the elements of the column Z_j with index in I.

  Let D^I be the submatrix composed by the columns of D whose indices are in I.

  $X_j^I \leftarrow Z_j^I$ /*The only nonzero elements in X_j are in the indices I. */

  /*multiplicative algorithm for β-divergence*/

  for $iter = 1$ **to** 5 **do**

    $F1 \leftarrow (D^I)^T (D^I X_j^I).^{\beta-2} .* Y_j$

    $G1 \leftarrow (D^I)^T (D^I X_j^I).^{\beta-1}$

    $X_j^I \leftarrow X_j^I .* (F1./G1)$ /* Nonneg. version $X_j^I = X_j^I .* [F1./G1]_+$ */

  end for

end for

Using the *power_beta* algorithm for the updating phase and the sparse pursuit algorithm for β-divergence, the proposed β-K-SVD algorithm is the one shown in Algorithm 7.

Algorithm 7 β-K-SVD Algorithm

Input: $Y \in \Re^{m \times n}, D \in \Re^{m \times K}, T_0, \beta \in \Re$
Output: $D \in \Re^{m \times K}, X \in \Re^{K \times n}$
repeat
  Apply sparse pursuit algorithm for β-divergence: W $\leftarrow$ sparse_pursuit_beta(Y, D,T$_0$,β)
  /* loop for update of the columns of the dictionary D */
  for $k = 1$ **to** K **do**
    Find the set of columns of Y that use the atom D_k, i.e. $w_k = \{i | 1 \leq i \leq n, X(k,i) \neq 0\}$. Let the cardinal $|w_k|$ be q.
    Compute $E_k \leftarrow Y - (DX - D_x X_k)$
    Obtain E_k^w as E_k restricted to columns with index in I_k, therefore E_k^w has q columns
    [u,v] $\leftarrow$power_beta(E$_k^w, \beta$)
    update $D_k \leftarrow u$, $X_k^T \leftarrow v$
  end for
until convergence

The nonnegative version is obtained by carrying out the changes suggested in *sparse_purs_beta* and in *power_beta*. When Comparing Algorithms 1 and 7, the only differences are in the sparse pursuit algorithm and in the *power_beta* algorithm. As mentioned above, the *power_beta* minimizes the Frobenius norm between A and uv^t if $\beta = 2$. Furthermore, the sparse pursuit algorithm is composed of standard minimization procedures for beta-divergences, which minimize the Frobenius norm in the case $\beta = 2$. Therefore, Algorithm 7 is equivalent to Algorithm 1 when $\beta = 2$.

3.3. Special cases

The proposed algorithm works properly when $\beta > 1$. However, when $\beta < 1$, the proposed algorithm does not work. The reason is that, in the updating phase, the submatrices E_k^w (obtained as the difference of two matrices) can easily have negative numbers. When the *power_beta* algorithm is applied to a matrix with negative numbers, sooner or later the power of a negative number will be computed (or the power of a number very close to zero, in the nonnegative version); then, either complex numbers or huge numbers appear, and the computation fails.

The Itakura-Saito case ($\beta = 0$) can be formulated without powers; however, the algorithm still fails. The reason for this can be traced again to the appearance of negative numbers in the submatrices E_k^w.

The KullBack Leibler case ($\beta = 1$) is of great practical importance, but it has some special features. First, it can be observed that when $\beta = 1$, the problem of finding the vectors u and v that minimize the KL divergence between a matrix A and the product uv^T is greatly simplified. Actually, the optimal vectors have simple expressions: u is a multiple of the vector obtained with the sums of the rows of A ($u = \omega \mathbf{s}, s_i = \sum_j A_{i,j}$) for some scalar ω, and v is a multiple of the vector obtained with the sums of the columns of A ($v = \omega \mathbf{s}, s_i = \sum_i A_{i,j}$) for some scalar ω. These optimal vectors are obtained straight away, after just one iteration of the *power_beta* method.

This has a positive influence on the computational cost of the updating of the dictionary. However, there is also a drawback. This drawback appears when a HALS-like procedure is applied to try to obtain matrices $U \in \Re^{m \times K}$ and $V \in \Re^{K \times n}$, $K > 1$, such that the divergence between A and UV^T is minimized. When a HALS-like procedure is applied to minimize the divergence between A and UV^T, with $\beta = 1$, the procedure stagnates after the first iteration and cannot reduce the β-divergence beyond the value obtained in the first iteration. The same phenomenon occurs if, with $\beta = 1$, one of the two matrices (U or V^T) is already known and we only seek the other matrix.

Nevertheless, it is easy to check that a multiplicative algorithm for the same

purpose (when $K > 1$) can usually reduce the β-divergence more than the value obtained in the first iteration of HALS. Therefore, our implementation of the sparse pursuit phase, in the case with $\beta = 1$, has been modified so that the multiplicative algorithm (Algorithm 4) is used in the first stage of Algorithm 6, instead of the HALS-like Algorithm 5.

In the updating of the dictionary, the number of inner iterations in Algorithm 3 is set to 1 when $\beta = 1$. As mentioned above, any additional iteration does not offer any improvement. In practice, and despite not having theoretical proof of convergence, Algorithm 7 converges reasonably well for $\beta \geq 1$.

4. Numerical Experiments

The goal of the paper is to propose a new algorithm that can be used to obtain overcomplete dictionaries that are are different from those obtained through other methods. Our belief is that the performance of this method (in efficiency and/or in accuracy) depends on many different parameters that must be chosen by the user (size of the dictionaries, desired number of nonzero elements, β parameter chosen, etc.), and, above all, on the problem being analyzed.

In order to give evidence of convergence of the algorithm, we have carried out experiments with random matrices, varying either the β parameter or the number of nonzero elements T_0. For the sake of completeness, we have also applied the nonnegative version of the β-K-SVD algorithm to a voice separation problem previously described in other papers [23].

4.1. Experiments with random matrices

We designed a simple experiment, in order to give empirical proof of convergence. We generated data matrices with 300 rows and 1000 columns, with elements drawn from a standard normal distribution ("randn" function of Matlab), setting all the negative numbers to 0. $K = 50$ was selected as the number of columns of the dictionary. The nonnegative β-K-SVD algorithm was tested with four different values of $\beta \in \{1, 1.5, 2, 2.5\}$ and 10 values for the maximum

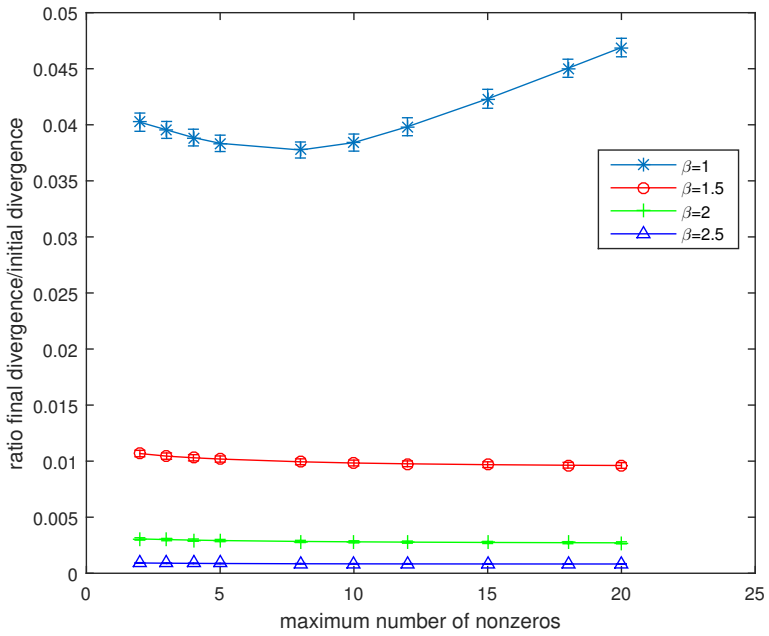

Figure 1: Ratios of final to initial β-divergences after 15 iterations for different numbers of nonzero elements and different values of β.

number of nonzero elements $T_0 \in \{2, 3, 4, 5, 8, 10, 12, 15, 18, 20\}$. For each combination of values of the parameters, we generated 10 data matrices Y_i. We also generated (with random positive numbers) initial dictionaries D_i^{ini} and initial coefficient matrices X_i^{ini}. The nonnegative β-K-SVD algorithm was applied to the data matrices Y_i, using as initial dictionaries D_i^{ini} and initial coefficient matrices X_i^{ini}, and obtaining as s result the final dictionaries D_i^{fin} and the coefficient matrices X_i^{fin}. A fixed number of iterations (15) was used in all of the cases. The average ratio between initial β-divergences ($D_\beta(Y_i||D_i^{ini}X_i^{ini})$ and the final β-divergences ($D_\beta(Y_i||D_i^{fin}X_i^{fin})$ was computed. All of the ratios were below 0.05, indicating convergence in all the cases. Figure 1 displays the average ratios, with error bars for standard deviation (although the standard deviation is quite small and can only be noticed for the case $\beta = 1$).

The initial and final average β-divergences for all of the experiments for each

β can be seen in Table 1. These results show that, while there is convergence in all of the cases tested, the convergence is better (beta divergence is reduced more) for cases with larger β: the worst case is with $\beta = 1$. This seems consistent with the algorithm failing to converge for $\beta < 1$.

β	1	1.5	2	2.5
Avg. initial β-divergences	2.6e+06	5.8e+06	1.4e+07	3.9e+07
Avg. final β-divergences	1.0e+05	5.8e+04	4.1e+04	3.4e+04

Table 1: Average initial and final β-divergences

4.2. Voice separation problem

An experiment of voice separation was carried out using as acoustic data the subset of the GRID corpus [26] that was used as the training set in the Speech Separation Challenge [27].

In this problem, the algorithm should compute the weights matrix X to approximate the mixture matrix Y (created by mixing two speech signals) taking into account the dictionary matrix D which contains dictionaries of both speakers from the original speech signals. The goal is to separate the mixed signal into two individual signals, one for each speaker.

For these experiments, 100 signals were generated mixing 2 random speakers for each signal from a pool of 34 speakers. Each signal is represented by a magnitude spectrogram matrix Y that was obtained by using the short-time Fourier transform with o columns (observations) and $f = 751$ rows (features). The number of observations o ranges between 94 and 177, with an average of 129.73.

The size of the dictionaries (number of columns) to be generated was chosen as 500. For each experiment, 6 dictionaries were generated for each speaker, varying the parameter β from 1 (Kullback-Leibler) up to 2.25, with step 0.25. In this problem, the optimal number of nonzero elements (from the point of view of the final signal-to-distortion ratio, SDR) was obtained experimentally

as 3, which was the number of nonzero elements used in the generation of the dictionaries. Of course, this number will be different for other problems.

Dictionaries $D_{i,j}$ were computed for all speakers $i = 1, .., 34$ and all values of β $j = 1, .., 6$. If the signal of a given experiment e was the signal y^e mixed from speakers a and b, for each method j the dictionaries $D_{a,j}$ and $D_{b,j}$ were appended; $D_j^e = [D_{a,j}, D_{b,j}]$, obtaining a dictionary of size 1000.

Then, the goal is to obtain the vector x that minimizes the β-divergence between $D_j^e \cdot x$ and the mixed signal y^e. The vector x is, again, obtained through a multiplicative algorithm to minimize β-divergence, with the same parameter β that was used for the generation of the dictionaries. The upper part of the vector x can be used to recover the signal from speaker a, and the lower part can be used to recover the signal from speaker b (See paper [23] for details).

The recovered signals can be used to compute the SDR (as described in [23]) between the separated and the original signal.

The average SDRs in dBs obtained are displayed in Fig.2. Clearly, the best result is obtained with $\beta = 1$. These results agree with other similar papers, that show that algorithms that minimize the Kullback-Leibler divergence ($\beta = 1$) provide better results in problems of this kind than algorithms that minimize Frobenius norm. In this case, this also means that the β-K-SVD algorithm can be tuned to improve the accuracy obtained with the standard K-SVD.

This tuning is possible in any situation where standard K-SVD can be used because, as was shown above, the standard K-SVD is actually a specific case (with $\beta = 2$) of the β-K-SVD algorithm.

5. Future Work

We have several active lines of research that are related to the β-K-SVD algorithm. Most of them are oriented towards obtaining faster versions of the β-K-SVD algorithm through GPU implementations. We are also looking for algorithmic improvements that could be applied to the β-K-SVD algorithm, such as the ones proposed [28, 29] for the standard K-SVD.

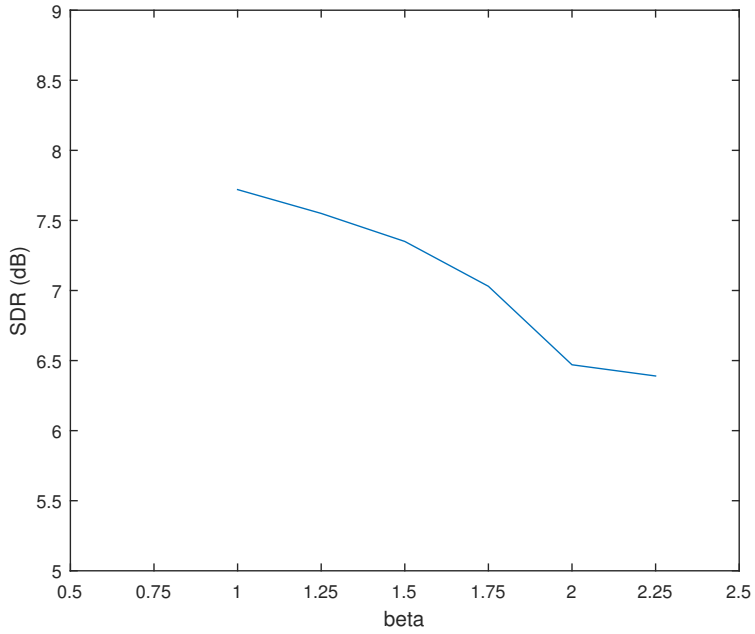

Figure 2: Signal to Distortion Ratio (dB) obtained in the experiment with varying β.

6. Conclusion

The standard K-SVD algorithm is a technique for obtaining overcomplete dictionaries with a desired, predefined degree of sparsity. However, it has only been developed for the case in which the distance minimization is carried out using the Frobenius norm of the difference between the signal matrix and the estimated matrix. It is well known that the choice of the measure of divergence between the signal matrix and the estimated matrix affects the solution obtained. We have extended this algorithm to the case where β-divergence is used as a measure of the separation between observed and estimated signals. This has involved reformulating the steps that are used in the K-SVD algorithm so that the β-divergence is minimized, instead of the Frobenius norm. In this way, an algorithm has been developed that is capable of addressing similar problems to those of K-SVD in fields where β-divergence is preferable to distances based on the Frobenius norm.

An especially interesting case is that of audio processing where the Gaussian noise model is less suitable than a noise model based on the Poisson distribution. This is why the algorithms that are based on β-divergence are more suitable for this field.

The developed algorithm has been tested on a real audio problem, verifying that the generated dictionaries are similar to those obtained by other methods. Similarly to those methods the algorithm works better in the $\beta = 1$ case.

In short, we propose a new tool that addresses the coding of signals and the computation of dictionaries by means of a reformulation of the K-SVD algorithm that minimizes the β-divergence instead of minimizing the Frobenius norm. This is especially useful for fields where the additive Gaussian noise model is not suitable.

7. Acknowledgements

This work has been partially supported by the EU together with the Spanish Government through TEC2015-67387-C4-1-R (MINECO/FEDER) and by Pro-

grama de FPU del Ministerio de Educacion, Cultura y Deporte FPU13/03828 (Spain).

8. Vitae

Victor M. Garcia-Molla received his Degree in Mathematics at the Universidad Complutense in Madrid (Spain), his M.Sc. in Industrial Mathematics at Strathclyde University (Glasgow, UK), and his PhD at the Universitat Politecnica de Valencia (Spain) in 1998. He is an Associate Professor at the Universitat Politecnica de Valencia, and his main research interests are high- performance computing and numerical methods for signal processing.

Pablo San Juan was born in 1990 in Valencia, Spain. He received a B.S degree in Computer Science and a B.S. degree in Telecommunications under the specialty of telematics from the University of Valencia (UV), Spain in 2013. He then received an M.Sc. degree in Parallel and Distributed Computing from the Universitat Politecnica de Valencia (UPV), Spain in 2014. Now he is about to finish his PhD in Computer Science at the Universitat Politecnica de Valencia (UPV). His current research interest are HPC systems, parallel computing, and nonnegative decompositions (NMF, NNLS).

Tuomas Virtanen is an Academy Research Fellow and Associate Professor (tenure track) at the Department of Signal Processing, Tampere University of Technology (TUT), Finland, where he is leading the Audio Research Group. He received the M.Sc. and Doctor of Science degrees in information technology from TUT in 2001 and 2006, respectively. He has been developing methods for single-channel sound source separation using non-negative matrix factorization based techniques, and noise-robust speech recognition, music content analysis, and audio event detection. In addition to the above topics, his research interests include content analysis of audio signals in general and machine learning. He has authored more than 100 scientific publications on the above topics.

Antonio M. Vidal received his Ph.D. degree in Computer Science from the Universitat Politecnica de Valencia, Spain, in 1990. Since 1992 he has been

at the Universitat Politecnica de Valencia, Spain, where he is currently a Full Professor in the Department of Computer Science. His main areas of interest include parallel computing with applications in numerical linear algebra and signal processing. In this field, he has published more than 100 articles in journals and international conference proceedings, has coordinated and participated in several research projects and has supervised 14 Ph.D. Theses.

Pedro Alonso (1968, Valencia, Spain) received the Engineering Degree in Computer Science in 1994 and the PhD Degree in 2003 from the Universitat Politecnica de Valncia (UPV), Spain. His dissertation was on the design of parallel algorithms for structured matrices with application in several fields of digital signal analysis. He is an Asociate Professor of the Department of Information Systems and Computation of the UPV since 1996 and is also a member of the High Performance Networking and Computing Research Group. His areas of interest are High Performance Computing, numerical algorithms, parallel computing and heterogeneous parallel computing, and hardware accelerators.

9. Bibliography

[1] M. Aharon, M. Elad and A. Bruckstein, "KSVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation", *IEEE T. Signal Proces.*, vol. 54, no. 11, pp. 4311–4322, 2006.

[2] S. Mallat, and Z. Zhang, "Matching Pursuits with Time-Frequency Dictionaries". *IEEE T. Signal Proces.* pp. 3397-3415, 1993, doi:10.1109/78.258082.

[3] F. Bergeaud, and S. Mallat, "Matching pursuit of images", in *Proc. International Conference on Image Processing.* pp. 53-56, 1995, doi:10.1109/ICIP.1995.529037.

[4] S. S. Chen, D.L. Donoho, and M. A. Saunders, "Atomic Decomposition by Basis Pursuit", *SIAM J. Sci. Comput.*, vol. 20 no.1, pp. 33-61, 2001, doi:/10.1137/S1064827596304010.

[5] G.H. Golub, and C.F. Van Loan. "Matrix Computations". The Johns Hopkins University Press, Baltimore, MD, USA, fourth edition, 2013.

[6] Y. Liang, Y. Tian, and M. Li, "Parallel transformation of K-SVD solar image denoising algorithm", in *Proc. Second International Conference on Photonics and Optical Engineering* 1025614 (2017) doi:10.1117/12.2256495

[7] Y. Li, F. Li, B. Bai, and Q. Shen, "Image fusion via nonlocal sparse K-SVD dictionary learning", *Applied Optics* Vol. 55, Issue 7, pp. 1814-1823 (2016) doi: 10.1364/AO.55.001814

[8] D V.R Mohan, I Rambabu, B Harish, "Denoising and SAR Image Classification with K-SVD Algorithm", .International Journal of Engineering and Technology, 7 (3.3) (2018) 36-40

[9] lSyed Zubair, Fei Yan, Wenwu Wang, "Dictionary learning based sparse coefficients for audio classification with max and average pooling" Digital Signal Processing Volume 23, Issue 3, May 2013, pp. 960-970 doi: 10.1016/j.dsp.2013.01.004

[10] M. Aharon, M. Elad and A. Bruckstein, "K-SVD and its non-negative variant for dictionary design" in *Proceedings of the SPIE Conference, Curvelet, Directional, and Sparse Representations II*, pp. 11.1-11.13. vol 5914, 2005.

[11] R. Peharz and F. Pernkopf, "Sparse nonnegative matrix factorization with 0-constraints", *Neurocomputing*, vol. 80 no. 1, (2012), pp. 38-46, doi: 10.1016/j.neucom.2011.09.024.

[12] A. Cichocki. R. Zdunek, A.H. Phan, and S.Amari. "Nonnegative Matrix and Tensor Factorization". John Wiley & Sons, Ltd, (2009)

[13] D. D. Lee, and H. S. Seung, "Algorithms for non-negative matrix factorization", *Adv. Neur. In.* vol. 13, pp. 556–562, (2001)

[14] A. Basu, I. R. Harris, N. L. Hjort, and M. C. Jones, "Robust and efficient estimation by minimising a density power divergence", *Biometrika*, vol. 85 no. 3, pp. 549-559, (1998)

[15] C. Fevotte, and J.Idier, "Algorithms for nonnegative matrix factorization with the β-divergence", *Neural Comput.* vol. 23 no.9, pp.2421–2456, 2011.

[16] D. Fitzgerald, M. Cranitch, E. Coyle, "On the use of the beta divergence for musical source separation" . IET Irish Signals and Systems Conference (ISSC 2009) pp. 1–6. doi:10.1049/cp.2009.1711. 2009

[17] Canadas-Quesada, F. Vera-Candeas, P., Carabias, J.J., Cabanas, P., Munoz, D. "Constrained non-negative matrix factorization for score-informed piano music restoration". *Digit Signal Process.* 50. doi:10.1016/j.dsp.2016.01.004. 2016

[18] M. Nakano, H. Kameoka, J. Le Roux, Y. Kitano, N. Ono, and S. Sagayama. "Convergence-guaranteed multiplicative algorithms for non-negative matrix factorization with beta-divergence". In Proc. IEEE International Workshop on Machine Learning for Signal Processing (MLSP2010), Sep. 2010.

[19] R. Rubinstein, M. Zibulevsky, and M. Elad, "Efficient Implementation of the K-SVD Algorithm using Batch Orthogonal Matching Pursuit, Technical Report - CS Technion (April 2008)

[20] D. Bartuschat, A. Borsdorf, and H. Kostler, "A parallel K-SVD implementation for CT image denoising", Friedrich-Alexander-University of Erlangen-Nuremberg", Department of Computer Science 10th (System-Simulation),2009

[21] C. L. Lawson, and R. J. Hanson, "Solving Least Squares Problems", Prentice Hall, 1974.

[22] N. Gillis, and F. Glineur, "Accelerated Multiplicative Updates and Hierarchical ALS Algorithms for Nonnegative Matrix Factorization", *Neural Comput.* vol. 24, no.4, pp. 1085–1105, 2012.

[23] T. Virtanen, J. F. Gemmeke, and B. Raj, "Active-Set Newton Algorithm for Overcomplete Non-Negative Representations of Audio",

IEEE T. Audio Speech, vol. 21 no. 11, pp. 2277–2289, 2013, doi:10.1109/TASL.2013.2263144.

[24] Matlab, The Mathworks Inc., MATLAB R14 Natick MA,2004.

[25] B. K. Natarajan, "Sparse Approximate Solutions to Linear Systems", *SIAM J. Comput.*, Volume 24 Issue 2, pp. 227–234,April 1995.

[26] M. P. Cooke, J. Barker, S. P. Cunningham, and X. Shao, An audiovisual corpus for speech perception and automatic speech recognition, Journal of the Acoustical Society of America, vol. 120, no. 5, 2006.

[27] M. Cooke, J. R. Hershey, and S. J. Rennie, Monaural speech separation and recognition challenge, Computer Speech and Language, vol. 24, no. 1, 2010.

[28] M. Sadeghi, M. Babaie-Zadeh and C. Jutten, Learning Overcomplete Dictionaries Based on Atom-by-Atom Updating, IEEE Trans. Signal Processing, vol. 62, no. 4, pp. 883-891, 2014.

[29] M. Sadeghi, M. Babaie-Zadeh and C. Jutten, Dictionary Learning for Sparse Representation: A Novel Approach, IEEE Signal Processing Letters, vol. 20, no. 12, pp. 1195-1198, 2013.