Dictionary learning for speech based on a doubly sparse greedy adaptive dictionary algorithm

Maria G. Jafari and Mark D. Plumbley

Abstract—In this paper we present a greedy adaptive dictionary learning algorithm, that sets out to find sparse atoms for speech signals, and in the process yields a sparse signal decomposition. The algorithm learns the dictionary atoms on data frames taken from a speech signal. It iteratively extracts the data frame with minimum sparsity index, and adds this to the dictionary matrix. The contribution of this atom to the data frames is then removed, and the process is repeated.

The algorithm is applied to the problem of speech representation and speech denoising, and its performance is compared to other existing methods. The method is shown to find dictionary atoms that are sparser than their time-domain waveform, and also to result in a sparser speech representation. In the presence of noise, the algorithm is found to have similar performance to the well established principal component analysis.

Finally, we apply a visualization method for the evaluation of the characteristics of the atoms within a dictionary and find that it conveys much more information about the time-frequency properties of the dictionary than looking at the spectrogram alone.

Index Terms—Sparse decomposition, adaptive dictionary, sparse dictionary, dictionary learning, speech analysis, speech denoising, dictionary visualization.

I. INTRODUCTION

Sparse signal representations allow the information within a signal to be conveyed with only a few elementary components, called atoms. For this reason, they have acquired great popularity over the years, and they have been successfully applied to a variety of problems, including the study of the human sensory system \cite{1,2,3}, blind source separation \cite{4,5,6}, and signal denoising \cite{7}. Successful application of a sparse decomposition depends on the dictionary used, and whether it matches the signal features \cite{8}.

Two main methods have emerged to determine a dictionary within a sparse decomposition: dictionary selection and dictionary learning. Dictionary selection entails choosing a pre-existing dictionary, such as the Fourier basis, wavelet basis or modified discrete cosine basis, or even constructing a redundant or overcomplete dictionary by forming a union of bases (for example the Fourier and wavelet bases) so that different properties of the signal can be represented \cite{9}. Dictionary learning, on the other hand, aims at deducing the dictionary from the training data, so that the atoms directly capture the specific features of the signal \cite{7}. Dictionary learning methods are often based on an alternating optimization strategy, in which the dictionary is fixed, and a sparse signal decomposition is found; then the dictionary elements are learned, while the signal representation is fixed.

Early dictionary learning methods by Olshausen and Field \cite{2} and Lewicki and Sejnowski \cite{10} were based on a probabilistic model of the observed data. Lewicki and Sejnowski \cite{10} clarify the relation between sparse coding methods and independent component analysis (ICA), while the connection between dictionary learning in sparse coding, and the vector quantization problem was pointed out by Kreutz-Delgado et al. \cite{11}. The authors also proposed finding sparse representations using variants of the focal undetermined system solver (FOCUSS) \cite{12}, and then updating the dictionary based on these representations. Aharon, Elad and Bruckstein \cite{13} proposed the K-SVD algorithm. It involves a sparse coding stage, based on a pursuit method, followed by an update step, where the dictionary matrix is updated one column at the time, while allowing the expansion coefficients to change \cite{13}. More recently, dictionary learning methods for exact sparse representation based on $\ell_1$ minimization \cite{8,14}, and online learning algorithms \cite{15}, have been proposed.

Generally, the methods described above are computationally expensive algorithms that look for a sparse decomposition, for a variety of signal processing applications. In this paper, we are interested in targeting speech signals, and deriving a dictionary learning algorithm that is computationally fast. The algorithm should be able to learn a dictionary from a short speech signal, so that it can potentially be used in real-time processing applications.

A. Motivation

The motivation for the work in this paper, was provided by previous results in \cite{16}, where a sparse coding method based on ICA (SC-ICA) \cite{17} was used to adaptively learn a dictionary from audio signals. For speech signals, many of the dictionary atoms were found to be localized in time and frequency. Figure 1 shows some of these atoms, along with their sparsity index, defined as \cite{18}

$$\xi = \frac{\|x\|_1}{\|x\|_2}.$$  

(1)
where \( \| \cdot \|_1 \) and \( \| \cdot \|_2 \) denote the \( \ell_1 \)- and \( \ell_2 \)-norm respectively. The sparsity index measures the sparsity of a signal, and is such that the smaller \( \xi \), the sparser the vector \( x \); for a Gaussian signal, the sparsity index is \( \xi \approx 18 \). Thus, although we were looking for a sparse signal decomposition, we ended up with sparse basis functions. This raises the question: if a sparse decomposition produces sparse atoms, what signal decomposition do we get when we look for sparse atoms?

The work in this paper aims at answering this question, and therefore we look for dictionary atoms that are sparse. This is especially interesting, since we are not following the usual approach of looking for a sparse decomposition, rather we derive a dictionary learning algorithm that learns sparse basis functions.

A major limitation of the method in [16] is the computational cost of the algorithm. Like other sparse coding methods in e.g. [2] or [10], whose development has been hindered by their computational cost [19], the method requires a large data set and is based on a lengthy training stage that results in an extremely expensive computational complexity. For this reason, a final motivating factor in deriving our algorithm is that it must be computationally fast and work on short data segments.

**B. Contributions**

In this paper we consider the formulation of our algorithm from the point of view of minimizing the sparsity index on atoms. We seek the sparsity of the dictionary atoms rather than of the decomposition, and to the authors’ knowledge this perspective has not been considered elsewhere. We show experimentally that our algorithm is “doubly sparse”, yielding both sparse atoms and sparse decompositions.

Further, we propose a stopping rule that automatically selects only a subset of the atoms. This has the potential of making the algorithm even faster, and to aid in denoising applications by using a subset of the atoms within the signal reconstruction. Finally, we consider a dictionary visualization method that conveys more information than simply inspecting the waveform or its frequency domain representation.

**C. Organization of the paper**

The structure of the paper is as follows: the problem that we seek to address is outlined in Section II, and our sparse adaptive dictionary algorithm is introduced in Section III, along with the stopping rule. Experimental results are presented in Section IV, including the investigation of the sparsity of the atoms and speech representation, the application of dictionary visualization, and speech denoising. Conclusions are drawn in Section VII.

**II. PROBLEM STATEMENT**

Given a one-dimensional speech signal \( x(t) \), we divide this into overlapping frames \( x_k \), each of length \( L \) samples, with an overlap of \( M \) samples. Hence, the \( k \)-th frame \( x_k \) is given by

\[
x_k = [x((k-1)(L-M) + 1), \ldots, x(kL - (k-1)M)]^T
\]

where \( k \in \{1, \ldots, K\} \). Then, we construct a new matrix \( X \in \mathbb{R}^{L \times K} \), whose \( k \)-th column is represented by the signal block \( x_k \), and whose \( (l,k) \)-th element is given by

\[
[X]_{l,k} = x(l + (k-1)(L-M))
\]

where \( l \in \{1, \ldots, L\} \), and \( K > L \).

The task is to learn a dictionary \( D \) consisting of \( L \) atoms \( \psi^l \), that is \( D = \{ \psi^l \}_{l=1}^{L} \), providing a sparse representation for the signal blocks \( x_k \). We seek a dictionary and a decomposition of \( x_k \), such that [20]

\[
x_k = \sum_{l=1}^{L} \alpha_k^l \psi^l
\]

where \( \alpha_k^l \) are the expansion coefficients, and

\[
\|\alpha_k\|_0 \ll L.
\]

The \( \ell_0 \)-norm \( \|\alpha_k\|_0 \) counts the number of non-zero entries in the vector \( \alpha_k \), and therefore the expression in equation (5) defines the decomposition as “sparse”, if \( L \) is small. In the remainder of this paper, we use the definition of sparsity in equation (1).

The dictionary is learned from the newly constructed matrix \( X \). In the case of our algorithm, we begin with a matrix containing \( K \) columns, and we extract the first \( L \) columns according to the criterion discussed in the next section.
III. Greedy Adaptive Dictionary Algorithm (GAD)

To find a set of sparse dictionary atoms, we consider the sparsity index $\xi$, as defined in equation (1), for each column $x_k$, of $X$, i.e.

$$\xi_k = \frac{\|x_k\|_1}{\|x_k\|_2}.$$  \hspace{1cm} (6)

Recall that the smaller the sparsity index, the sparser the vector $x_k$. Our aim is to sequentially extract new atoms from $X$ to populate the dictionary matrix $D$, and we do this by finding, at each iteration, the column of $X$ with minimum sparsity index

$$\min_k \xi_k.$$ \hspace{1cm} (7)

Practical implementation of the algorithm begins with the definition of a residual matrix $R^l = [r_1^l, \ldots, r_K^l]$, where $r_k^l \in \mathbb{R}^K$ is a residual column vector corresponding to the $k$-th column of $R^l$. The residual matrix changes at each iteration $l$, and is initialized to $X$. The dictionary is then built by selecting the residual vector $r_k^l$ that has lowest sparsity index, as illustrated in Algorithm 1.

\begin{algorithm} 
\caption{Greedy adaptive dictionary (GAD) algorithm} 
\begin{algorithmic} 
1. Initialize: $l = 0, \text{D}^0 = [\ ] \{\text{empty matrix}\}, \text{R}^0 = X$ 
2. \textbf{repeat} 
3. Find residual with lowest $\ell_1$- to $\ell_2$-norm ratio: 
$$k^l = \arg \min_{k \in \mathbb{Z}^+} \left\{ \|r_k^l\|_1 / \|r_k^l\|_2 \right\}$$ 
4. Set the $l$-th atom equal to normalized $r_k^l$: 
$$\psi^l = r_k^l / \|r_k^l\|_2$$ 
5. Add to the dictionary: 
$$\text{D}^l = [\text{D}^{l-1}[\psi^l], \text{I}^l = \text{I}^{l-1} \cup \{k^l\}$$ 
6. Compute the new residual: 
$$r_k^{l+1} = r_k^l - \psi^l (\psi^l, r_k^l)$$ 
7. \textbf{until} “termination” (see Section III-A) 
\end{algorithmic} 
\end{algorithm}

We call our method the greedy adaptive dictionary (GAD) algorithm [21].

Aside from the advantage of producing atoms that are directly relevant to the data, the GAD algorithm results in an orthogonal transform. To see this, consider re-writing the update equation in step 6 in Algorithm 1, as the projection of the current residual $r_k^l$ onto the atom space:

$$r_k^{l+1} = P_{\psi} r_k^l = \left( I - \frac{\psi^l \psi^l^T}{\psi^l \psi^l} \right) r_k^l = r_k^l - \frac{\psi^l (\psi^l, r_k^l)}{\psi^l \psi^l}.$$ \hspace{1cm} (8)

It follows from step 4 in Algorithm 1, that the denominator in the right-hand-side of equation (8) is equal to 1, and therefore the equation corresponds to the residual update in step 6. Orthogonal dictionaries have the advantage being easily invertible, since if the matrix $B$ is orthogonal, then $BB^T = I$, and evaluation of the inverse simply requires the evaluation of the matrix transpose.

We will see later in Section IV-C that although we wish to find a dictionary whose atoms are sparse, and make no assumptions on the signal decomposition, the GAD algorithm results in a reasonably sparse decomposition. Therefore, the algorithm has the effect of being doubly sparse on speech signals, by sparsifying both the atoms and the decomposition.

A. Termination rules

We consider two possible termination rules:

1) The number of atoms $l$ to be extracted is pre-determined, so that up to $L$ atoms are learned. Then, the termination rule is:
- Repeat from step 2, until $l = N$, where $N \leq L$.

2) The reconstruction error at the current iteration $\epsilon^l$ is defined, and the rule is:
- Repeat from step 2 until
$$\epsilon^l = \|\hat{x}^l(t) - x(t)\|_2 \leq \sigma$$ \hspace{1cm} (9)

where $\hat{x}^l(t)$ is the approximation of the speech signal $x(t)$, obtained at the $l$-th iteration from $X^l = D^l (D^l)^T X$, by reversing the framing process; $D^l$ is the dictionary learned so far, as defined in step 5 of Algorithm 1.

IV. Experiments

We compared the GAD method to PCA [23], SC-ICA [17] and K-SVD [13]. SC-ICA and K-SVD were chosen because they learn data-determined dictionaries, and look for a sparse representation. PCA was chosen because it is a well-established technique, commonly used in speech coding and therefore it sets the benchmark for the speech denoising application.

We used the four algorithms to learn 512 dictionary atoms from a segment of speech lasting 1.25 sec. A short data segment was used because this way the algorithm can be used within real-time speech processing applications. The data was taken from the female speech signal “supernova.wav” by “Corsica_S”, downloaded from The Freesound Project database [22], and downsampled to 16 kHz. We also used the male speech signal “Henry5.mp3” by “acclivity”, downloaded from the same database, and downsampled to 16 kHz.

The K-SVD Matlab Toolbox [24] was used to implement the K-SVD algorithm. K-SVD requires the selection of several parameters. We set the number of iterations to 50, as recommended in [13], and the number of nonzero entries $T_0$ in the coefficient update stage to 10, which we found empirically to give a more accurate, although not as sparse, signal representation than $T_0 = 3$, as used in [13]. The dictionary size was set to 512 and the memory usage to “high”.

The SC-ICA method was used as described in [17], using 50000 iterations.

A. Computational Complexity

In Table I we report the computational times of the algorithms, when learning a dictionary from speech segment of 1.25 sec, and averaged over 100 trials. SC-ICA was also applied to the short data set, since we are not concerned with the output of the algorithm, only its computational load. Two versions of the K-SVD were also compared: the original version which is fully based on Matlab M-code, and the second version, which combines M-code with optimized MEX functions written in C. The experiments were conducted on a Quad-Core Intel Xeon Mac at 2.66 GHz, using Matlab.
Fig. 2. Examples of the frames of the original speech signals, and of the atoms learned with the PCA, SC-ICA, K-SVD and GAD algorithms. The SC-ICA algorithm is applied to a speech segment of length 1.25 sec, and to a longer one of length 196 sec.

The results in the table show that SC-ICA requires over 22 hours on average to learn a dictionary of 512 atoms. The size of the training set does not affect the computational complexity of SC-ICA, because the algorithm randomly selects a $512 \times 512$ matrix of training data at each iteration. When a larger data set is used, the algorithm has more “information” from which to learn the dictionary, while working with a short data segment, the training data is not varied enough for the atoms to be learned to an adequate level [25].

GAD and K-SVD (v2) only need about 2 minutes, and PCA needs as little as 7 sec. However, note how the K-SVD version based exclusively on M-code requires around 1 hour and 45 minutes to learn the dictionary. Therefore, we expect that optimizing the code for GAD will lead to even faster computational complexity.

B. Learned Atoms

We begin by visually inspecting some examples of the atoms learned with the four algorithms, and then considering the
sparsity of the atoms and signal representation. Figure 2(a) shows examples of the overlapping data blocks found in the columns of the matrix \( X \), from which each dictionary is learned, while the remaining plots in the figure show examples of the atoms learned with PCA, SC-ICA, K-SVD and GAD. The sparsity index relating to each atom is also shown.

The atoms extracted with PCA (Fig. 2(b)) are not localized. Comparing them with Figure 2(a), they do not appear to be capturing any particular features of the speech signal. Figure 2(c) and (d) show the atoms obtained with SC-ICA from a short speech segment of 1.25 sec, and a longer data sample, lasting 196 sec, respectively. When the data is short (Figure 2(c)), the resulting atoms are not localized, and the sparsity index is quite high in all cases. This appears to contradict the results shown in Figure 1, which provided motivation for our work. Conversely, the atoms learned from the longer data set (Figure 2(d)) are very localized and reminiscent of wavelet-type functions, similar to the atoms in Figure 1. Following from this result, in the remainder of this paper we will only apply the SC-ICA method to the longer speech signals of 196 sec.

The K-SVD atoms (Fig. 2(e)) exhibit some structure that generally seems to correspond to that of the original data blocks. The atoms obtained with the GAD algorithm are illustrated in Figure 2(f). Those atoms extracted earlier, shown on the first two lines, are quite similar to the original data, and are also the sparsest atoms, as indicated by the low sparsity index. Atoms extracted later, shown on the last two lines in the figure, capture mostly “noise”-like characteristics, or less meaningful features of the signal.

1) Applying K-SVD and GAD to a larger data set: To compare directly with SC-ICA, even when the whole 196 sec signal was used, we attempted to learn dictionaries with GAD and K-SVD from the whole longer speech. However, having obtained a matrix \( X \) of size 512 × 261557, neither GAD nor K-SVD was able to perform even one iteration, both resulting in an “Out of memory” error in Matlab.

C. Sparsity of Atoms and Representation

We have seen in Figure 2 how the GAD algorithm yields atoms that are initially quite sparse and then become more “noise”-like. To investigate this further, 100 segments were taken from the original speech data, each lasting 1.25 sec. PCA, K-SVD and GAD were used to learn dictionaries from each segment. The sparsity index \( \xi_k \) for each atom was then evaluated, and the average across the 100 trials was taken. SC-ICA atoms were learned from a speech sample lasting 196 sec during a single trial.

Figure 3 shows the atom sparsity index for the framed speech data in the columns of \( X \), and for the atoms learned with PCA, K-SVD and GAD. Recall that a sparse atom is characterized by a low sparsity index. The plot shows that the atoms learned by GAD in the beginning are the sparsest, and after around 200 atoms have been extracted, the sparsity index is close to its maximum value. The behavior observed here is in agreement with what was observed in Figure 2. It also shows that the atoms obtained with the other algorithms are not as sparse as those extracted by GAD, except for the SC-ICA atoms. The original data blocks that are considered in the figure correspond to the columns in \( X \) that are extracted by GAD, and therefore they are the sparsest within the speech segment.

To compare the sparsity of the SC-ICA atoms to the results

<table>
<thead>
<tr>
<th>Method</th>
<th>Average Computation Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA</td>
<td>7</td>
</tr>
<tr>
<td>SC-ICA</td>
<td>80729</td>
</tr>
<tr>
<td>K-SVD (Matlab only)</td>
<td>6710</td>
</tr>
<tr>
<td>K-SVD (v2)</td>
<td>163</td>
</tr>
<tr>
<td>GAD</td>
<td>167</td>
</tr>
</tbody>
</table>

**Table I** Comparing the computational complexity for the PCA, SC-ICA, K-SVD and GAD algorithms. The table shows the average computational time for each algorithm, obtained over 100 trials.

![Image](image-url)
obtained with the other algorithms, we obtain a single value by averaging the atom sparsity index not only across the 100 trials but also across all 512 atoms (the average is only over the 512 atoms in the SC-ICA case).

The results are shown in the first column of Table II, and they validate our expectations: GAD yields atoms that are sparser than the original signal blocks, and than all the algorithms except for SC-ICA. However, when we used the termination rule in equation (9) (shown in Table II as GAD-TR), with $\sigma = 5 \times 10^{-3}$, the average sparsity index for the GAD atoms decreased from 16.2 to 12.6. On average, GAD-TR was found to learn less than 110 atoms, which from Figure 3 can be seen to correspond to those atoms that are sparsest. The algorithms perform in a similar way on the male speech.

Next, we seek to determine how sparse is the representation obtained with the GAD method. We do this by considering the transform coefficients obtained with all methods, for each block, and across the 100 speech segments taken from the speech signal, each lasting 1.25 sec. The sparsity index of the transform coefficients is found each time. We then average across the 100 segments, and across all block, to obtain the atom sparsity index of the transform coefficients obtained with all methods, for each method, and for both the female and male speech signals, as shown in the second column of Table II. This also includes values for the sparsity index for the original signal blocks in $X$. The lowest representation sparsity index value is obtained with K-SVD, thanks to the strong sparsity constraint imposed by the algorithm on the signal decomposition. This entails limiting the number of non-zero elements in the signal representation to a small number (we use $T_0 = 10$). The signal transformed with the GAD algorithm is sparser than in the time domain, and than the coefficients obtained with SC-ICA and PCA when all atoms are used in the signal reconstruction, for both signals. Moreover, the representation becomes even sparser when GAD is used with the termination rule.

Thus, although we set out to find a dictionary whose atoms are sparse, and make no assumptions on the signal representation, we end up with a sparse decomposition. Hence, the algorithm has the effect of being doubly sparse on speech by sparsifying both the atoms and the decomposition.

### D. Representation Accuracy

The accuracy of the signal approximation given by each algorithm can be assessed with the reconstruction error $\epsilon$, as defined in equation (9), after the dictionary has been learned

$$
\epsilon = \|\hat{x}(t) - x(t)\|_2
$$

where $\hat{x}(t)$ is the signal approximation obtained from $X = D^T (D^T)^\dagger X$, and $(D^T)^\dagger$ is the right pseudo-inverse of $D^T$. This is plotted in Figure 4 for each algorithm, as the number of atoms omitted in the signal reconstruction goes from 0 to 462 (or, the total number of atoms used goes from 512 down to 50). K-SVD has a non-zero reconstruction error even when all atoms are included in the signal approximation, because the transform is not complete, and therefore it does not result in an exact reconstruction.

In general, the results show that all algorithms perform quite well when few atoms are omitted in the reconstruction. As more and more atoms are omitted, the reconstruction error...
increases. PCA performs best, because the transform arranges the signal components so that most energy is concentrated in a small number of components, corresponding to those extracted earlier. The GAD transform also gives good signal approximations as more atoms are excluded from the reconstruction, although its performance seems to worsen as the number of omitted atoms becomes more than 300 (or less than 200 atoms are used in the reconstruction). This corresponds to the number, identified in Figure 3, below which the GAD atoms are sparest, and above which the sparsity index reaches its maximum value. SC-ICA and K-SVD yield signal approximations that suffer most from the reduction in the number of atoms, with SC-ICA performance suddenly worsening as the number of excluded atoms goes from about 300 to over 400.

The behavior of GAD suggests that it might be suitable for denoising applications. Hence, we will consider this problem later in this paper.

E. Dictionary Visualization

The frequency spectra of the original signal and learned dictionaries, are plotted in Figure 5, where they are ordered according to their centre frequency. The plots show that the original signal has little energy above 4kHz, while the SC-ICA atoms seem to have energy that is more or less equal across all frequencies and across time. The energy of PCA and GAD atoms is also more spread across the time-frequency plane, while K-SVD yields atoms whose energy is focused in the lower frequencies. These plots convey little more information, and at times they are hard to interpret also because the printed plot is not as clear as the one on the computer screen.

For an alternative visualization, we apply a method for visualizing the dictionary that was first proposed in [17], and subsequently in [26]. It entails plotting an axis-aligned ellipse for each atom that corresponds to its position and spread in time and frequency.

To do this, each squared atom $(\psi^i)^2$ and its squared Fourier transform $(\Psi^i)^2$ are treated as a Laplacian probability distribution:

$$f(g|\mu, b) = \frac{1}{2b} e^{-\frac{|g-\mu|}{2b}}$$

The median $\mu_{1/2}(g)$ and absolute deviation from that median

$$d_i = \frac{1}{L} \sum_{i=1}^{L} |g_i - \mu_{1/2}(g)|$$

are then evaluated from $(\psi^i)^2$ and $(\Psi^i)^2$. For the time domain waveform, the median $\mu_{1/2}(g)$ and deviation $d_i$ correspond, respectively, to the location of the atom and its spread on the time axis. For the frequency domain case, the two quantities, $\mu_{1/2}(g)$ and $d_i$, represent the centre frequency of the atom and its bandwidth. Then, for each basis vector, an ellipse is drawn on the time-frequency plane [17], centered at $(\mu_{1/2}^t(g), \mu_{1/2}^f(g))$ and with radia $d_i^t$ and $d_i^f$. The energy of each atom relative to the one with the highest energy is encoded by the grey scale of the elliptical plots.

The results are shown in Figure 6. The plot in Figure 6(a) shows the results from the original signal divided into overlapping data frames. The ellipses for the PCA atoms (Figure 6(b)) are concentrated in the centre of the time scale, while the SC-ICA atoms (Figure 6(c)) evenly tile the time-frequency plane, and are well localized, except for the low frequencies. This is consistent with the wavelet-like basis functions learned by SC-ICA from speech (see [17] for more details). Visualization of the plot for the K-SVD atom (Figure 6(d)) shows that this is the only algorithm that yields atoms whose elliptical plots are similar to the original signal. Several of the elliptical plots for the GAD atoms (Figure 6(e)) are focused in the middle region, although less than the PCA atoms. However, unlike PCA, the GAD atoms cover more of the time-frequency plane, and ellipses are clearly present in the high and low frequencies, similarly to the original signal.

We can further investigate the behavior of the GAD algorithm by considering the atoms it learned according to when they were extracted in the learning process. In Figure 7, the ellipses for the GAD atoms are plotted for the atoms extracted early on (atoms 1 to 150), and later on in the learning process (atoms 151 to 512). These regions were
selected by inspecting the average sparsity index in Figure 3, and they are expected to correspond to those atoms that are sparsest (atoms 1 to 150), followed by the atoms that correspond mostly to “noise”-like features (atoms 151 to 512). As well as being the sparsest, the early atoms are more similar to the speech signal, because the algorithm will not have had time to make a big impact on the training data. The plots in Figure 7(a) show that the early atoms indeed have time-frequency characteristics that are similar to the speech signal. The plots in Figure 7(b) are the result of the learning process.

Figures 8 and 9 show that similar results are obtained for the male speech signal. Here, the distinction between the first 150 atoms that are extracted, and those learned subsequently, is even more accentuated. Figure 9 shows that all elliptical plots above 4kHz is related to atoms extracted later in the process, while the initial 150 atoms result in a plot that is similar to the original signal.

V. APPLICATION TO SPEECH DENOISING

The term denoising refers to the removal of noise from a signal, and it was first used to describe the application of methods based on the wavelet transform for noise reduction [27]. Sparse transforms have been found to be among the most successful methods for denoising [13], and dictionary learning methods have been used for this application [28].

Table III shows the tolerance of the PCA, SC-ICA, K-SVD and GAD algorithms to a noise level changing from 20 dB to -10 dB, as the number of atoms in the reconstruction is reduced from 512 to 50. This is evaluated with the improvement in signal to noise ratio (ISNR):

$$ISNR = 10 \log \frac{E\{(x(t) - x_n(t))^2\}}{E\{x(t) - \hat{x}(t))^2\}}$$

where $x(t)$ is the original signal, $x_n(t)$ is the observed distorted (noisy) signal, and $\hat{x}(t)$ is the source approximated by the transform. As the signal approximation becomes closer to the original source, ISNR increases.

When all atoms are used in the reconstruction, the complete transforms PCA, SC-ICA and GAD, yield an ISNR of 0 dB, while K-SVD gives a non-zero ISNR, since the approximation...
is not exact. Generally, K-SVD has been shown to perform well for tasks such as image denoising [7], and the results in Table III show that this is also true for speech: the algorithm yields the highest ISNR values across all experiments. For the remaining algorithms, when the noise is low (10 dB), reducing the number of atoms in the reconstruction leads to distortion in the signal approximation, at times yielding negative results in ISNR (this is the case for SC-ICA). As the level of noise increases, the high ISNR values for PCA and GAD indicate that there are benefits in reducing the number of atoms used in the signal approximation. It is well-known that PCA can reduce the level of noise present, but it decomposes the space into signal and noise subspaces, and the results in Table III show that the performance of GAD is similar. Conversely, SC-ICA appears to perform best in environments where the noise dominates.

VI. DISCUSSION

GAD is a computationally fast algorithm that finds atoms having characteristics similar to those of the SC-ICA atoms, namely, that are sparse and that cover a relatively large area of the time-frequency plane. It yields a signal decomposition that is as sparse as that obtained with SC-ICA, even though, unlike the SC-ICA method, GAD does not make any assumptions on the decomposition. It results in better signal reconstruction than SC-ICA and it has good tolerance to noise and does not exhibit distortion when noise reduction is performed at low noise levels.

We are particularly interested in applying the GAD method to other audio signals, such as music, and potentially other signal processing areas, such as image processing and biomedical signal processing. Biomedical applications typically give rise to large data sets, for instance, in microarray experiments the expression values of thousands of genes are generated. Therefore, in this case the algorithm would have to be extended to deal with large data sets. Finally, we are considering the application of this approach to the problem of source separation.

VII. CONCLUSIONS

In this paper we have presented a greedy adaptive dictionary learning algorithm, that finds new dictionary elements that are sparse. The algorithm constructs a user-defined dictionary, whose atoms encode local properties of the signal, and which has been shown to result in an orthogonal dictionary.

The algorithm has been shown to be a doubly sparse method for speech signals, as it yields sparse atoms and a sparse signal representation. Its performance was compared to that of SC-ICA, PCA and K-SVD methods, and it was found to give good signal approximations, even as the number of atoms in the reconstructions decreases considerably.

A dictionary visualization method was used to investigate the properties of the algorithms in the time-frequency domain, and it was found to convey more information than inspecting the atom waveform, or looking at the spectrogram of the dictionary. It was also observed that the algorithm has good tolerance to noise in speech denoising applications.

### Table III

<table>
<thead>
<tr>
<th>Noise Level</th>
<th>Method</th>
<th>Number of Atoms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>512 400 300 200 100 50</td>
</tr>
<tr>
<td>10 dB</td>
<td>PCA</td>
<td>0.00 0.52 1.32 2.61 4.74 5.69</td>
</tr>
<tr>
<td></td>
<td>SC-ICA</td>
<td>0.00 0.12 -2.38 -6.46 -8.74 -9.16</td>
</tr>
<tr>
<td></td>
<td>K-SVD</td>
<td>5.10 6.01 6.83 7.45 7.00 5.85</td>
</tr>
<tr>
<td></td>
<td>GAD</td>
<td>0.00 1.40 2.97 4.71 5.10 2.53</td>
</tr>
<tr>
<td>0 dB</td>
<td>PCA</td>
<td>0.00 0.50 1.30 2.69 5.42 8.33</td>
</tr>
<tr>
<td></td>
<td>SC-ICA</td>
<td>0.00 0.46 2.00 2.09 1.46 1.26</td>
</tr>
<tr>
<td></td>
<td>K-SVD</td>
<td>4.89 5.98 7.10 8.52 10.17 10.97</td>
</tr>
<tr>
<td></td>
<td>GAD</td>
<td>0.00 1.50 3.29 5.65 7.20 7.27</td>
</tr>
<tr>
<td>-10 dB</td>
<td>PCA</td>
<td>0.00 0.49 1.28 2.65 5.34 8.28</td>
</tr>
<tr>
<td></td>
<td>SC-ICA</td>
<td>0.00 0.60 3.31 8.26 10.25 10.47</td>
</tr>
<tr>
<td></td>
<td>K-SVD</td>
<td>4.70 5.75 6.96 8.53 10.64 12.07</td>
</tr>
<tr>
<td></td>
<td>GAD</td>
<td>0.00 1.47 3.27 5.80 8.86 10.21</td>
</tr>
</tbody>
</table>

**ISNR for the GAD, SC-ICA, PCA and K-SVD algorithms. All ISNR values are expressed in decibels (dB).**

### REFERENCES


